UNIVERSIDAD NACIONAL DE COLOMBIA

Estudio de la reducción del sobreajuste en arquitecturas de redes neuronales residuales ResNet en un escenario de clasificación de patrones

Manuela Viviana Chacón Chamorro

Universidad Nacional de Colombia
Facultad de ciencias exactas y naturales
Manizales, Colombia
2023

Estudio de la reducción del sobreajuste en arquitecturas de redes neuronales residuales ResNet en un escenario de clasificación de patrones

Manuela Viviana Chacón Chamorro

Trabajo de grado presentado como requisito parcial para optar al título de:
Magister en Ciencias - Matemática Aplicada

Director:
Ph.D. Juan Carlos Riaño Rojas
Co-director:
Ph.D. Fernando Andrés Gallego Restrepo

Grupo de Investigación:
PCM Computational applications

Universidad Nacional de Colombia
Facultad de ciencias exactas y naturales
Manizales, Colombia
2023

A mi familia,
Gracias por darle sentido, dirección y magnitud
a cada acción de mi vida.

A quienes me hacen sentir en familia,
Gracias por ser mis puntos de equilibrio en este
universo caótico.

Agradecimientos

Agradezco a la Universidad Nacional de Colombia sede Manizales, a la Facultad de Ciencias Exactas y Naturales y al grupo de investigación PCM Computational Applications. También expreso mis agradecimientos al proyecto "Creating Diverse Scientific Communities in the Americas JUUNTOS: Joining UNAL Manizales and UTRGV in the Development of STEM Collaboration", código 55327, por financiar y apoyar la divulgación de resultados de esta investigación. Agradezco de manera especial al docente Juan Carlos Riaño Rojas, quien dirigió este trabajo con dedicación y paciencia. Muchas gracias por motivarme siempre, por encontrar la manera adecuada de transmitir sus conocimientos y por contribuir e impactar en mi formación académica. Así mismo, agradezco la co-asesoría del docente Fernando Gallego Restrepo, sus ideas y discusiones contribuyeron al desarrollo de esta investigación. Agradezco también a quienes me escucharon y debatieron algunas ideas, sus aportes y el interés demostrado fueron realmente significativos. Finalmente, expreso mis agradecimientos a todas las personas que me apoyan en el proceso de cumplir mis metas. Gracias.

Resumen

Estudio de la reducción del sobreajuste en arquitecturas de redes neuronales residuales ResNet en un escenario de clasificación de patrones

Las redes neuronales artificiales son una técnica de aprendizaje automático inspirada en el funcionamiento biológico de las neuronas, actualmente soportan gran parte de la denominada Inteligencia Artificial. Pese a su notable evolución estos algoritmos presentan el problema de sobreajuste, "memorización de los datos de entrenamiento", lo cual disminuye la capacidad de generalización. En este trabajo se estudió el sobreajuste en un escenario de clasificación de patrones y se determinó un método para resolver el problema. Este estudio se realizó para la arquitectura de neuronal residual (ResNet) y se sustentó en el análisis de las propiedades matemáticas de la función que representa esta estructura, en particular, la continuidad de Lipschitz. La validación del método se realizó comparando su desempeño con las técnicas convencionales de reducción de sobreajuste: la regularización L1, L2 y *Dropout*. Variando la profundidad de la red se realizaron dos experimentos de clasificación con los conjuntos de datos *Digits* y *Fashion* de MNIST. También se efectuaron pruebas en arquitecturas definidas para 3 conjuntos de datos convencionales y 3 de datos sintéticos. Adicionalmente, se realizaron dos experimentos que incluyeron imágenes adversarias. El método desarrollado presenta un desempeño destacable logrando: comportamiento similar en las curvas de aprendizaje para entrenamiento y prueba, menor variabilidad del modelo al cambiar el conjunto de entrenamiento, reducción de la cota de Lipschitz, tolerancia a las pruebas adversarias. En síntesis, el método propuesto resultó idóneo en la reducción del sobreajuste en las arquitecturas residuales de los experimentos y tolera de manera sobresaliente ataques adversarios.

Palabras clave: continuidad Lipschitz, generalización, ODENet, regularización, Redes neuronales, ResNet, sobreajuste.

Abstract

Study of Overfitting Reduction in Residual Neural Network Architectures (ResNet) in a Pattern Classification Scenario

Artificial neural networks are a technique of machine learning inspired by the biological functioning of neurons, currently supporting a significant portion of the so-called Artificial Intelligence. Despite their notable evolution, these algorithms present the problem of overfitting, "training data memorization", which reduces the capacity of generalization. In this work, overfitting in a pattern classification scenario was studied and a method to solve the problem was determined. This study was carried out for the Residual Neural Network architecture (ResNet) and was based on the analysis of the mathematical properties of the function that represents this structure, in particular, the Lipschitz continuity. The method was validated by comparing its performance with conventional overfitting reduction techniques: L1, L2 and Dropout regularization. Varying the depth of the network, two classification experiments were performed with the data sets Digits and Fashion MNIST. Tests were also performed on architectures defined for 3 conventional data sets and 3 synthetic data sets. Additionally, two experiments were conducted that included adversarial images. The developed method posed remarkable performance achieving: similar behavior in the learning curves for train and test set, less variability of the model when changing the train set, reduction of the Lipschitz bound and adversarial test tolerance. In summary, the method is suitable to reduce overfitting in residual architectures of the experiments and it tolerates adversary attacks in an outstanding way.

Keywords: Generalization, Lipschitz continuity, Neural Networks, ODENet, overfitting, regularization, ResNet.

Contenido

Lista de Figuras

Lista de Tablas

Lista de símbolos

Símbolos con notación matemática

Símbolo	Definición
$\mathbb{R}$	Conjunto de los números reales
$\mathbb{Z}$	Conjunto de los números enteros
$\mathcal{X}$	Espacio de los datos de manera que $\mathcal{X} \subseteq \mathbb{R}^n$
$\mathcal{Y}$	Espacio de las etiquetas de manera que $\mathcal{Y} \subseteq \mathbb{R}^p$
$\mathbf{W}$	Matriz de pesos. En general las letras mayúsculas con este formato de fuente representaran estructuras matriciales
$\mathbf{X}$	Matriz de observaciones
$\mathbf{x}$	Vector de datos, $\mathbf{x} \in \mathbb{R}^n$ Las letras minúsculas con este formato de fuente representaran estructuras vectoriales. Los vectores se asumen de tipo columna.
$\|\| \cdot \|\|_F$	Denotará la norma de Frobenious.
$\|\| \cdot \|\|_2$	Denotará la norma de espectral en matrices y la norma euclídea en vectores.
∇	Gradiente

Símbolos con letras latinas

Símbolo	Definición
C	Número de clases en un problema de clasificación multiclase

Símbolo	Definición
K	Número entero que representa la cantidad de observaciones
K_f	Número de pliegues en validación cruzada
K_L	Constante de Lipschitz
L	Número entero que representa la cantidad de capas de una red neuronal.
LB	Cota superior de la constante de Lipschitz
L1	Regularización L1 en el contexto de las redes neuronales
L2	Regularización L2 (*weight decay*) en el contexto de las redes neuronales
l	Función de perdida definida como $l(\hat{y}, y) : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$
N_b	Número de lotes
N_e	Número de épocas
q y q'	Número entero que representa la cantidad de neuronas en alguna capa.
s	Número entero que representa la cantidad de capas residuales en una red neuronal
T	Operación transpuesta
y	Etiqueta para algún dato de entrada $\mathbf{x}$
X	Población de los datos
$\hat{y}$	Etiqueta estimada por la red para algún dato de entrada $\mathbf{x}$

Símbolos con letras griegas

Símbolo	Definición
Γ	Factor de perturbación usado en la generación de imágenes adversarias
γ	Factor de perturbación usado en la generación de imágenes adversarias
ϵ	Cantidad arbitrariamente pequeña mayor que cero
η	Tasa de aprendizaje

Símbolo	Definición
θ	Parámetros de la red neuronal, constituye una dupla que resume todas las matrices de pesos de la red y los valores del bias $\{\mathbf{W}, \mathbf{b}\}$
$\theta_\star$	Parámetros óptimos de la red neuronal
$\Theta_\star$	Parámetros óptimos de la red que generalizan a toda la población
λ	Parámetro de regularización, también denotará los valores propios de una matriz, su uso en este trabajo depende del contexto
$\rho(\cdot)$	Radio espectral de una matriz
σ	Función de activación de las neuronas. Se considera una función real, su versión vectorial se denotara como $\bar{\sigma}$
$\Omega(\cdot)$	Función de regularización

Subíndices

Símbolo	Definición
i	Índice para denotar la observación i-esima de un conjunto de datos
j	Índice para denotar la componente j-esima de un vector. Puede variar dependiendo el contexto.

Superíndices

Símbolo	Definición
n	Número entero que representa la dimensión de las observaciones.
p	Número entero que representa la dimensión de las etiquetas.
(j)	El superíndice entre paréntesis denotará el índice de la capa de una estructura neuronal.

Símbolo	Definición
(ad)	Identifica un dato o imagen adversaria.

Introducción

Las tecnologías para la clasificación de patrones exhiben un amplio desarrollo investigativo y sus potenciales aplicaciones se encuentran en diferentes áreas, todas de interés mundial. Estas tecnologías se soportan por los denominados clasificadores, algoritmos matemáticos que permiten separar en clases un conjunto de datos. Los clasificadores se construyen teórica y computacionalmente desde diferentes enfoques, por ejemplo, se distinguen aquellos que se soportan en inferencias estadísticas como es el caso de los clasificadores Bayesianos. Otras técnicas ampliamente utilizadas son los algoritmos de partición de datos, entre estos se destaca el algoritmo K-NN (*k-nearest neighbors*), el cual emplea la distancia euclídea para asociar el elemento a clasificar a la clase de los datos más cercanos.

En los clasificadores también se pueden encontrar técnicas de clasificación jerárquicas, su funcionamiento radica en emplear uniones que preserven la estructura de las relaciones entre las observaciones de un conjunto de datos. Otra categoría de algoritmos de clasificación es el grupo de los métodos basados en funciones Kernel. En esta categoría se encuentra el clasificador SVM (*Support Vector Machine*), su funcionamiento implica utilizar una función que transforma los datos no separables linealmente, es decir, aquellos que no se pueden dividir a través de hiper-planos, a una dimensión superior donde sí lo sean.

Como se puede notar son diversos los métodos de clasificación, entre estos se destacan las redes neuronales artificiales. Este método corresponde al tipo de algoritmos de aprendizaje automático inspirados en la estructura y el funcionamiento del cerebro humano [1]. Desde su desarrollo la técnica fue creciendo en popularidad, esto puede atribuirse a que se desempeña de manera óptima cuando se enfrenta a problemas de datos que no son linealmente separables. Además, las redes neuronales artificiales se pueden adaptar a diversos conjuntos de datos y tienen capacidad para trabajar en escenarios multivariables y con gran cantidad de observaciones. Actualmente las redes neuronales soportan, en gran medida, la denominada Inteligencia Artificial.

Las redes neuronales artificiales son un algoritmo conexionista y esto permite construir diversos modelos variando la topología de la red [1]. Actualmente, se cuenta con arquitecturas de red complejas tal es el caso de las redes neuronales convolucionales CNN (*Convolutional Neural Network*) [2, 3], las redes generativas antagónicas GAN (*Generative Adversarial Net-*

works) [4] y las redes neuronales residuales ResNet (*Residual Neural Networks*) [5]. Todas estas diversas arquitecturas han sido empleadas en innumerables aplicaciones. Por ejemplo, aplicaciones al sector energético [6, 7], soluciones para problemáticas en agricultura [3], soporte al diagnóstico médico [8], entre otras.

A pesar de su adecuado funcionamiento y las diversas aplicaciones de las redes neuronales artificiales, aún no es posible construir de forma analítica una frontera óptima de separación de clases. La frontera debería ser capaz de dividir las clases sin memorizar los datos particulares de un experimento y sin ser tan simple que presente un error de clasificación alto, estos problemas se denominan sobreajuste y subajuste, respectivamente [9, 10, 11]. En clasificadores sobreajustados, la respuesta del algorimo ante observaciones diferentes a las del conjunto de entrenamiento genera un desempeño pobre, esta incapacidad de generalización puede repercutir negativamente en los resultados esperados y vulnera la confiabilidad del sistema de clasificación.

La problemática del sobreajuste ha sido trabajada en la literatura a través de diversos métodos, sin embargo y en general, la metodología empleada continúa dependiendo de técnicas de prueba-error o de métodos que se particularizan a una aplicación. El problema mencionado se ha atribuido a diversos factores, por nombrar algunos: ruido en los datos de entrenamiento, desproporción entre el número de muestras usadas para entrenar y validar, sobre entrenamiento, inadecuada relación entra la complejidad del modelo y la estructura de los datos, ineficiente selección de características y deficiente sintonización de hiper-parámetros [9, 11, 12]. La metodología prueba-error y las diversas fuentes de origen del problema dificultan generar una estrategia de solución que se adapte a diferentes escenarios de clasificación, actualmente el problema continua abierto.

Por otra parte, en los últimos años un nuevo enfoque para el análisis, entrenamiento y creación de nuevas arquitecturas de red se ha popularizado. Este marco teórico propone que ciertas arquitecturas de redes neuronales, particularmente las redes residuales ResNet, son equivalentes a la discretización de una ecuación diferencial [13, 14, 15]. Lo anterior implica que la red tendrá un equivalente a un sistema continuo y el problema de entrenamiento se traslada a un problema de control óptimo [16, 17, 18, 19]. Este enfoque brinda unas herramientas matemáticas novedosas que no se habían tenido en cuenta y que pueden ser usadas para contribuir al problema de sobreajuste [20, 21, 22].

Otras investigaciones han propuesto analizar las redes neuronales como funciones matemáticas y estudiar sus propiedades para interpretar y manipular su comportamiento. En esta área de estudios se destaca el análisis de la propiedad de continuidad de Lipschitz para estructuras neuronales [23, 24, 25, 26]. En la literatura se señala que esta propiedad tiene relación con

la capacidad de generalización del clasificador, es decir, esta ligada a su tolerancia al sobreajuste [23, 24, 25, 26]. Además, se menciona que generar estructuras con un cierto control de la propiedad de continuidad Lipschitz es adecuado para contrarrestar posibles ataques adversarios [27].

En ese sentido, bajo el enfoque de la representación de las redes como funciones matemáticas, el presente trabajo tiene como objetivo general estudiar un método para disminuir el sobreajuste. Lo anterior se realiza para un entorno de clasificación de patrones en un escenarios de datos no separables linealmente. El método se desarrolla para la arquitectura de red neuronal residual ResNet.

Para lograr el objetivo planteado se propone como primer objetivo específico, describir el funcionamiento computacional y matemático de la red neuronal residual ResNet. El segundo objetivo es estudiar las condiciones que originan el problema del sobreajuste. A partir de estos resultados, el tercer objetivo busca adaptar un método para disminuir el sobreajuste en la red residual. Finalmente, se ha trazado como último objetivo específico estimar la eficiencia del método en un escenario de clasificación de patrones.

El presente documento se encuentra dividido en capítulos, el planteamiento del problema de investigación se expone en el Capítulo 1, en este apartado se encuentra la descripción del área problemática, la formulación de la pregunta de investigación, la justificación y los objetivos. En el Capítulo 2 se presenta la contextualización teórica, incluyendo la formulación del problema de aprendizaje en redes convencionales y particularizado a la estructura ResNet. Al finalizar este capitulo se detalla el análisis del problema de sobreajuste y se incluyen simulaciones computacionales que replican el fenómeno. Estos capítulos engloban los primeros dos objetivos específicos, con sus resultados se impulsa la propuesta del método para la reducción del sobreajuste y su posterior validación.

Siguiendo los objetivos específicos, en el Capítulo 3, se adapta un método para resolver la problemática del sobreajuste, en este apartado se incluye el análisis teórico y los resultados de experimentos computacionales para validar el método propuesto. En el Capítulo 4 se exponen las conclusiones y trabajo futuro, en este capitulo también se incluye una sección adicional (Sección 4.2) en la cual se resume la perspectiva de la arquitectura ResNet desde el análisis de un sistema continuo. Para esta sección se consideró incluir el modelo matemático, el proceso de entrenamiento y las condiciones que derivan estructuras sobreajustadas. Al finalizar el documento se encuentran los anexos que incluyen demostraciones, propiedades, definiciones y teoremas relevantes.

1. Descripción del problema

1.1. Área problemática

Las redes neuronales artificiales son clasificadores que realizan la separación entre clases emulando el comportamiento de las redes neuronales biológicas [28]. Inspirado en el proceso de aprendizaje biológico, las redes artificiales son una arquitectura computacional conexionista. Las unidades básicas que conforma la red se conocen como neuronas, estas se ubican en capas y se conectan entre sí con las neuronas de la siguiente capa.

El proceso de aprendizaje, o entrenamiento, consiste en estimar los pesos de las conexiones de la red a partir de un conjunto de datos previamente etiquetados, típicamente conocido como conjunto de entrenamiento. Matemáticamente, el entrenamiento puede ser observado como un problema de optimización [29]. Los pesos se encuentran aplicando estrategias que generalmente emplean metodologías que involucran el gradiente de alguna función relacionada con el error de clasificación. Se espera que una vez establecidos los pesos de las conexiones el modelo computacional de la red pueda clasificar un nuevo dato del cual se desconoce su clase.

El modelo más simple de red neuronal se conoce como Perceptrón [30], esta arquitectura, compuesta únicamente por una capa de entrada y una de salida, realiza la separación entre clases empleando fronteras lineales. Por ejemplo, para un conjunto de datos con dos características $\mathbb{R}^2$ la frontera será una recta, para el caso de tres características $\mathbb{R}^3$ se emplea un plano, y de forma general para un espacio de características n-dimensional $\mathbb{R}^n$ se emplea un hiper-plano de dimensión $\mathbb{R}^{n-1}$. Lo anterior es realizable siempre y cuando el conjunto de datos sea linealmente separable, característica que no siempre se cumple en datos de aplicaciones reales. Esta situación motivó la aparición de redes con más capas, denominadas capas ocultas, esta modificación de la arquitectura permite construir fronteras de separación no lineales.

Las arquitecturas con varias capas ocultas se conocen como Redes Neuronales Profundas (*Deep Neural Network*) [31]. Actualmente dentro de este modelo, se encuentran redes más complejas como es el caso de las redes neuronales convolucionales CNN (*Convolutional neural network*), arquitecturas con capas ocultas que incluyen operaciones de convolución, ampliamente empleadas en tareas de clasificación de imágenes [2]. Otro ejemplo, son las redes

generativas antagónicas GAN (*Generative Adversarial Networks*) [4] y las redes neuronales residuales ResNet (*Residual Neural Networks*) [32], estas últimas son la arquitectura de estudio para esta propuesta. Las arquitecturas de redes continúan en estudio, proponiendo diferentes tipos y variaciones acordes a las aplicaciones de diferentes campos.

Las neuronas de las capas ocultas se componen por una unidad sumadora y una función que limita su activación. Estas neuronas reciben los valores de la salida de las neuronas de las capas anteriores, ponderados por el peso de la conexión, los ingresan a la unidad sumadora y les aplican la función de activación para propagar esta salida a la siguiente capa. El proceso se repite hasta la capa de salida, la cual también se compone por neuronas con unidades sumadoras y una función de activación [29].

Al enfrentarse a un problema de clasificación y elegir una red neuronal artificial como algoritmo para resolverlo, el investigador deberá establecer la arquitectura de la red [33]. Este proceso implica definir de antemano parámetros como el número de capas de la red, el número de neuronas en cada capa, la función de activación, el método de optimización, entre otros. Los aspectos a sintonizar y la arquitectura de la red establecerán la complejidad del modelo. El proceso para elegir los valores se conoce como sintonización de hiper-parámetros y se relaciona directamente con la selección del modelo.

En general, las redes neuronales artificiales son adecuadas y ampliamente usadas para labores de clasificación (y regresión), incluso cuando se enfrentan a datos que no son linealmente separables. No obstante, este clasificador tiene varias problemáticas que afectan su eficiencia, robustez y confiabilidad, entre las principales se encuentra el sobreajuste frecuentemente conocido como *overfitting* y el subajuste o *underfitting* [9, 11].

El sobreajuste sucede cuando la red neuronal se sobre-entrena con ciertos datos, a tal nivel que un dato con una pequeña variación diferente a la del conjunto de entrenamiento, pero de la misma clase, arrojará un mayor error y puede clasificarse en la clase incorrecta. En estos casos el sistema "memoriza" los datos de entrenamiento y no funciona bien con datos sutilmente diferentes a estos, es decir, se pierde la capacidad de generalización. Situación análoga ocurre con el subajuste o *underfitting*, en este contexto el entrenamiento es tan pobre que la frontera es muy simple y el error de clasificación entre clases es alto [11, 9, 20].

Actualmente, los estudios y la literatura en el contexto de las redes neuronales artificiales son amplios, pero existen menos estudios enfocados en analizar y optimizar su desempeño, comparados con la gran cantidad de publicaciones de las diferentes aplicaciones. Adicionalmente, se encuentran diferentes herramientas computacionales y librerías, como *Statistics and Machine Learning Toolbox* [34] de Matlab, *Neuralnet* [35] en R o *TensorFlow* [36] en

Python, que permiten al usuario una rápida implementación de cualquier arquitectura de red neuronal. Si bien estas herramientas son útiles y simplifican la implementación computacional, varios autores se limitan a usarlas sin analizar el problema que desean resolver, considerándose como cajas negras (*black box*), en las cuales se ingresan datos y en poco tiempo se logra un clasificador que aparentemente tiene un buen desempeño. Sin embargo, el poco análisis del problema, la inadecuada validación y la sintonización heurística de los hiper-parámetros de la red, entre otras malas prácticas, genera fenómenos de sobreajuste o subajuste [9].

Para solucionar la problemática mencionada los estudios se han enfocado desde diversas perspectivas, una de ellas es la correcta selección de características. En este ámbito se han realizado propuestas que optimizan los resultados de la red al combinarla con métodos que seleccionan de la totalidad de características las que mayor relevancia tienen para la clasificación, o con algoritmos que reducen la dimensión de los datos a través de transformaciones [37, 38, 9]. Otros estudios que se han realizado corresponden a determinar los hiper-parámetros adecuados que mejoren el desempeño del clasificador. Por ejemplo, en el artículo [10] se responsabiliza al método de optimización basado en gradiente de generar sobreajuste, como solución se establece un nuevo método de aprendizaje que mantiene la relación entre la precisión y el sobreajuste. También se puede encontrar en la literatura métodos que emplean algoritmos heurísticos para la sintonización de hiper parámetros en las arquitecturas de redes, algunas de las técnicas utilizadas son: algoritmos genéticos [39], optimización por enjambre de partículas (PSO) [40], optimización por colonias de hormigas, entre otros [41, 42].

A nivel nacional, generalmente, el enfoque se concentra en el estudio de aplicaciones a problemas concretos de clasificación o en sistemas de predicción y regresión de datos a través de algoritmos de aprendizaje. Las investigaciones desarrolladas abordan desde diferentes tópicos aplicados, por ejemplo en salud [43], en predicción de series de tiempo [44], toma de decisiones financieras [45, 46], entre otros. Pese a los pocos estudios orientados a estudiar el problema del sobreajuste y la compresión matemática de las redes neuronales, se han encontrado algunas investigaciones que se relacionan con el tema, por ejemplo, se destaca un trabajo de tesis en el cual se presentó un modelo denominado Red Estocástica Generativa de aprendizaje supervisado basado en funciones kernel [47].

En importante señalar que la mayoría de las contribuciones para solventar el problema descrito, se soportan en métodos heurísticos, en particular aquellos métodos trabajados para la sintonización de los hiper-parámetros de la red neuronal. Actualmente el uso de clasificadores depende de múltiples parámetros y para su construcción y validación se emplea, en general, la metodología prueba-error. Además, la causa del problema se atribuye a diferentes factores

y no existe una forma estándar para abordar y solucionar parcial o totalmente la problemática. En este sentido, son válidas todas las contribuciones para mejorar el rendimiento de los clasificadores, y en particular el desempeño de las redes.

Por otra parte, en los últimos años se ha destacado un nuevo enfoque para el análisis, entrenamiento y creación de nuevas arquitecturas de red. Esta novedosa perspectiva teórica propone que ciertas arquitecturas de redes neuronales, particularmente las redes residuales ResNet, pueden ser vistas como la discretización de una ecuación diferencial [13, 48]. En consecuencia, la red tendrá un equivalente a un sistema continuo y el problema de entrenamiento se traslada a un problema de control óptimo [17, 18, 19, 49] o un problema de control simultaneo [50, 51]. Este enfoque está en continua exploración y brinda un marco teórico novedoso y flexible que puede ser empelado como escenario para atender el problema de sobreajuste y subajuste [17, 20].

Otro enfoque motivado desde el análisis matemático sugiere que los modelos robustos a pequeñas variaciones de los datos de entrada pueden generalizar de forma adecuada, es decir, presentan menor sobreajuste. La sensibilidad a las variaciones en la entrada ha sido caracteriza por la propiedad de continuidad de Lipschitz de la red neuronal [23]. La idea general del planteamiento de las investigaciones en esta área es acotar la constante de Lipschitz de la red o forzar que la red tenga un valor deseado de la constante [25, 24]. Los estudios de este tipo presentan algunas dificultades, por ejemplo, encontrar el valor preciso de la constante de Lipschitz de la función que caracteriza una determinada arquitectura de red es un problema NP-Hard, esta dificultad ha motivado algoritmos para un calculo que se aproxime al valor buscado [52, 53]. Por otra parte, la modificación teórica que sugieren este tipo de estudios implica la ejecución de algoritmos computacionales complejos, lo que afectaría el tiempo de entrenamiento de los modelos y la capacidad de extender los métodos a arquitectura y datos de alta dimensión.

En relación a lo mencionado, es evidente que existe una problemática a resolver en los clasificadores y concretamente en el caso de las redes neuronales artificiales. Si bien la literatura ha considerado el problema del sobreajuste y subajuste y lo ha intentado resolver a través de diferentes métodos, el área aún continúa abierta a diferentes aportes. Además, los nuevos enfoques teóricos, el análisis de las redes neuronales residuales como sistemas continuos y la continuidad de Lipschitz, son una oportunidad para realizar aportes teóricos y computacionales novedosos que contribuyan al problema expuesto.

1.2. Formulación del problema

¿Cómo reducir el sobreajuste (*overfitting*) en arquitecturas de redes neuronales residuales (ResNet) para un escenario de clasificación de patrones?

1.3. Justificación

Las redes neuronales artificiales son una herramienta matemática y computacional que ha permitido solucionar varios problemas prácticos de diferentes áreas, para ilustrar algunos ejemplos se pueden consideran las siguientes aplicaciones: gestión energética y redes eléctricas inteligentes *Smart Grid* [54, 55, 56, 57], agricultura 4.0 [3, 58, 59], soporte al diagnóstico médico [8, 60, 61], entre otros. Como puede notarse las redes neuronales artificiales abarcan una amplia gama de aplicaciones en diferentes áreas. De forma general se puede decir que los clasificadores, entre los que se encuentran las redes neuronales, tienen aplicaciones diversas y relevantes en diversas áreas de conocimiento. Es por esto que ocupan un lugar importante en la producción científica, actualmente se cuenta con líneas de investigación activas dedicadas exclusivamente al estudio de los algoritmos de clasificación y en particular a las redes neuronales.

En el caso de Colombia, el Plan Nacional de CTeI para el desarrollo del sector TIC [62] plantea 10 programas para dar solución a los retos que se enfrenta el país. En el primer programa denominado "Computación Centrada en las Personas" se prioriza el tema de procesamiento de imágenes y en el segundo programa "Sistemas Inteligentes" se hace énfasis en las temáticas de *machine learning*, reconocimiento de patrones e inteligencia artificial, tópicos en los cuales se engloba esta propuesta. Además, las diversas aplicaciones de los clasificadores responden a las necesidades expuestas en el Plan de Desarrollo Nacional 2018-2022 "Pacto por Colombia" [63] y se contemplan dentro de la linea transversal la cual prioriza la transformación digital del país. Estas aplicaciones también se articulan con la Política Nacional de Ciencia e Innovación para el Desarrollo Sostenible - Libro Verde 2030 [64], la Agenda 2030 y los Objetivos de Desarrollo Sostenible, políticas que exponen las necesidades actuales y señalan una ruta de acción para el desarrollo científico. En concreto, y para ilustrar dos ejemplos de áreas en las cuales son relevantes los estudios de clasificación y que responden a las necesidades y proyecciones del país, se resalta el sector salud y el sector agropecuario y agroindustrial.

Son amplias las aplicaciones en salud que pueden cubrirse desde la clasificación de datos y que resultan de interés nacional y mundial. Estudios en soporte al diagnóstico médico en enfermedades de ligamentos y tendones, clasificación de tumores, diagnóstico oportuno de cáncer, son solo algunos ejemplos que se pueden citar de las amplias aplicaciones de gran relevancia para la comunidad médica [65, 66]. En el caso del sector agropecuario y agroindustrial, al-

tamente priorizado en la política pública, la clasificación de datos puede solucionar distintas problemáticas. Dentro de los retos que se pueden resolver a través de esta técnica se encuentra la correcta fertilización de cultivos, la distinción de plantas con presencia de maleza [67], la acertada determinación de áreas para el riego [68], la clasificación de estados fisiológicos y fitopatológicos de una planta [69], la predicción de productividad de una cosecha, entre otras.

La variedad de aplicaciones pertinentes para la región permiten reconocer que los algoritmos de clasificación son de gran importancia y de interés nacional y global. Estas herramientas matemáticas y computacionales atienden a la solución de las problemáticas nacionales actuales y son tecnologías a la vanguardia de las tendencias de innovación mundial. En este sentido, los estudios que aporten a esta línea de investigación son también significativos y en particular, las investigaciones orientadas a mejorar el desempeño del clasificador, como en el caso de este trabajo, son trascendentales y de interés no solo para la comunidad científica, sino para la sociedad en general.

Por otra parte, las investigaciones de aplicaciones que emplean las redes neuronales artificiales son múltiples en comparación con sus estudios teóricos. De estos últimos, algunos se han enfocado en resolver el problema de sobreajuste, pero las técnicas mayormente desarrolladas corresponden a métodos heurísticos que se condicionan y particularizan a la aplicación que se desea atender. A diferencia de estos métodos, en este trabajo se pretende realizar un estudio con un enfoque distinto; se propone analizar las propiedades matemáticas subyacentes en la función que representa la red neuronal residual. La perspectiva propuesta brinda un enfoque teórico–matemático para finalmente lograr determinar un método que contribuya parcialmente a resolver el problema mencionado, pero no se fundamenten netamente en una metodología prueba-error.

El resultado de esta investigación podrá ser utilizado en estudios posteriores que requieran una clasificación robusta de diferentes tipos de datos. Además, con el estudio de la problemática se darán las pautas para continuar con investigaciones equivalentes en arquitecturas de redes neuronales que sean similares a las redes residuales. Se espera que se pueda adaptar, en cierta medida, los resultados a diversos clasificadores que así lo permitan y contribuir a incrementar la documentación en el área de estudio.

Es importante resaltar que para esta investigación se contó con los recursos bibliográficos y de infraestructura computacional dispuestos por la Universidad Nacional de Colombia – Sede Manizales, el apoyo del grupo PCM *Computational Applications* el cual ha logrado una clasificación en categoría A1 y prioriza dentro de sus líneas de investigación los tópicos en los cuales se enmarca el presente trabajo. El trabajo se desarrolló con la asesoría del docente Juan Carlos Riaño Rojas, quien se dedica ampliamente a la investigación de procesamiento

de señales e imágenes y los tópicos relacionados con el aprendizaje de máquina. Además, se contó con el apoyo en el rol de co-asesor del docente Fernando Andrés Gallego Restrepo, quien se destaca por sus amplios conocimientos en las áreas de Análisis Matemático y Ecuaciones Diferenciales Parciales.

1.4. Objetivos

1.4.1. Objetivo general

Determinar un método para reducir el sobreajuste en arquitecturas de redes neuronales residuales ResNet en un escenario de clasificación de patrones.

1.4.2. Objetivos específicos

- Describir el funcionamiento matemático y computacional de la red neuronal residual ResNet en un escenario de clasificación de patrones.
- Estudiar las condiciones que originan sobreajuste en la red neuronal residual ResNet.
- Adaptar un método que reduzca el sobreajuste en la arquitectura de red neuronal residual ResNet, teniendo en cuenta de las condiciones que originan este problema.
- Evaluar la eficiencia del método en relación a las técnicas convencionales en un escenario de clasificación de patrones.

2. Marco Teórico

En el año 1958 el investigador Frank Rosenblatt introdujo la primera idea de una red neuronal artificial. Rosenblatt inspirado por investigaciones sobre el funcionamiento del cerebro humano propuso la arquitectura artificial más sencilla denominada **Perceptrón** [30], esta estructura emula el comportamiento de una neurona biológica. La solución aportada en este trabajo permitió establecer un clasificador binario que podía separar correctamente datos a través de fronteras lineales. Se considera que el Perceptrón es el inicio de las diferentes arquitecturas de red neuronal construidas hasta la fecha.

En el Perceptrón, la capa de entrada recibe un vector de características que representa matemáticamente a un individuo del conjunto de entrenamiento[1]. Estas entradas son ponderadas por los pesos de la conexión y se transmiten a una unidad sumadora y con una cierta función de activación. La función de activación finalmente genera la salida de la red. Esta estructura es un discriminante lineal el cual puede realizar la separación entre dos clases. En la Figura **2-1** se presenta un diagrama de la arquitectura de esta configuración.
Matemáticamente el problema de clasificación dada la estructura del Perceptrón se puede expresar de la siguiente manera. Considere el conjunto de las entradas $\mathcal{X} \subseteq \mathbb{R}^n$, con n arbitrario según el problema, y las etiquetas $\mathcal{Y}$, usualmente corresponden al conjunto binario $\{0, 1\}$. Entonces, el Perceptrón es una función $f : \mathcal{X} \to \mathcal{Y}$ que tiene la forma descrita en la ecuación 2-1.

$$\hat{y} = f\left(\sum_{j=1}^{n} w_j x_j + b\right) \tag{2-1}$$

La Ecuación 2-1 se puede expresar de forma vectorial como $\hat{y} = f(\mathbf{w}^T\mathbf{x}+b)$, en esta expresión $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{w} \in \mathbb{R}^n$ y $b \in \mathbb{R}$. El parámetro b se denomina *bias*, la función f es conocida como la activación de la neurona, los pesos de las conexiones emulan la ponderación de las dendritas en un neurona biológica, la unidad sumadora es similar a su soma y la salida es el axón [30].

[1] En clasificación de imágenes se pueden extraer características representativas de la señal o usar como vector de entrada directamente los píxeles de la imagen, esta última técnica es el fundamento del *Deep learning*

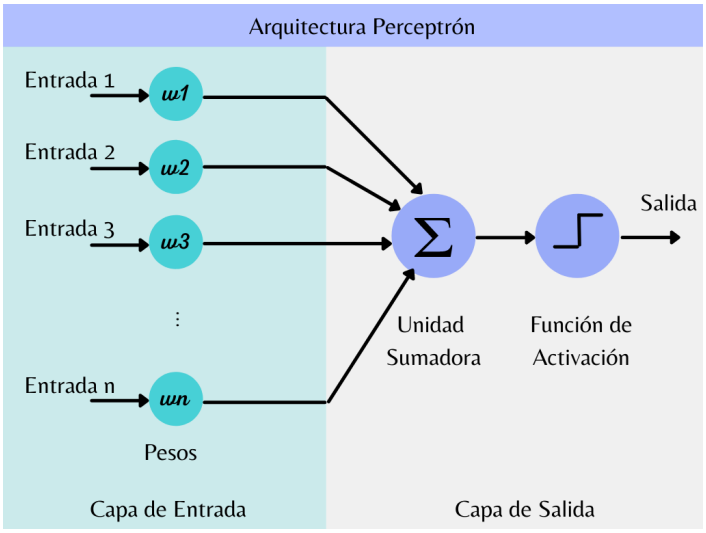


Figura 2-1.: Diagrama de la arquitectura Perceptrón. Fuente: elaborado por el autor. (En adelante todas las figuras que no mencionen fuente se atribuyen a la elaboración del autor).

Para el problema de aprendizaje considere que se tienen K duplas de la forma $\{\mathbf{x}_i, y_i\}$, con $\mathbf{x}_i \in \mathcal{X}$ y $y \in \mathcal{Y}$, $\forall i = 1 : K$. Es decir dado un vector $\mathbf{x}_i$ característico del individuo i se conoce su etiqueta y_i, entonces con los K individuos se pretende encontrar los valores de los pesos $\mathbf{w}$ óptimos de manera que la Ecuación 2-1 permita generalizar ante datos $\mathbf{x}$ de los cuales se desconoce su clasificación. En la Sección 2.1 se describe el problema de aprendizaje de las redes neuronales para una estructura general.

A partir de los estudios realizados por Rosenblatt y la creación de la arquitectura del Perceptrón las investigaciones en torno a su teoría continuaron extendiéndose. En el año 1969 los autores Minsky y Papert, encontraron que el Perceptrón no funcionaba de manera adecuada si los datos no eran linealmente separables [70]. El ejemplo más sencillo de una estructura con estas condiciones es imitar la función *XOR*. Para resolver el problema y generalizar la estructura de red neuronal artificial a cualquier conjunto de datos, los autores proponen incluir más capas completamente conectadas. Con esta nueva perspectiva se logra establecer fronteras de separación no lineales.

La idea de incluir más capas y funciones de activación no lineales motivó el **Teorema de Aproximación Universal**, una primera versión fue realizada en el año 1989 [71] y se restringía al uso de funciones de activación sigmoideas. Este teorema establece que una red neuronal compuesta por una capa oculta y un número finito de neuronas, es un aproximador universal entre funciones continuas en un subconjunto compacto de $\mathbb{R}^n$, si se realizan algunas suposiciones sobre la función de activación. Esto implica que una arquitectura de

red podría representar una amplia variedad de funciones si los pesos de las conexiones son los adecuados. Sin embargo, el teorema no indica como deberían ser los pesos, solo señala que tal construcción es posible.

Al agregar capas y neuronas la cantidad de pesos que se deben hallar crece significativamente. Para el año 1986 se presentó una perspectiva teórica para encontrar los pesos de las conexiones [72]. La técnica consistió en propagar el error de la salida de la red hacia atrás a través de las capas. Este método sentó las bases para lo que hoy se conoce como *backpropagation* y es el algoritmo más empleado en el entrenamiento de redes neuronales. Esta metodología resulta viable independiente de la arquitectura de la red.

Resolver el problema de aprendizaje impulsó el desarrollo de diversas arquitecturas de redes neuronales, aplicaciones e investigaciones en torno a esta temática. En las últimas décadas gracias al poder computacional y tecnológico disponible, se han generado arquitecturas avanzadas que requieren establecer los pesos para un número elevado de conexiones. Entre estas arquitecturas se destacan las redes convolucionales CNN (*Convolutional Neural Network*), las cuales realizan operaciones de convolución entre sus capas y son útiles para la clasificación de imágenes [2]. También se encuentran las redes LSTM (*Long Short Time Memory*) arquitectura de carácter recurrente generalmente usada en aplicaciones de reconocimiento de voz, reconocimiento de texto, predicción de series de tiempo, entre otras [73]. En los últimos años se han popularizado modelos como las redes adversarias GAN (*Generative Adversarial Networks*), *Encoders*, *Tranformers*, cada uno con sus propias cualidades y aplicaciones [4, 74].

Otra topología de red neuronal destacada es la **Red Neuronal Residual ResNet** (*Residual Neural Network*). Este trabajo se enmarca en el estudio de esta arquitectura. A continuación, en la Sección 2.1 se describe el problema de aprendizaje y el algoritmo *backpropagation* empleado para encontrar los parámetros de cualquier arquitectura de red, en la Sección 2.2 se describe a detalle el modelo matemático y algunas propiedades se la ResNet. En la Sección 2.3 se presenta el problema de sobreajuste y subajuste desde el enfoque general y al finalizar el capitulo se exponen algunos métodos de solución que se han empleado para la problemática descrita.

2.1. Problema de aprendizaje

Considere el conjunto de las entradas $\mathcal{X} \subseteq \mathbb{R}^n$, con n arbitrario según el problema, y las etiquetas $\mathcal{Y}$. De forma general, para un problema de clasificación multiclase se considera $\mathcal{Y} \subseteq \mathbb{R}^p$. Denotaremos $\mathbf{W}$ las matrices de pesos de las capas de la red. Si se conecta una capa de q neuronas con una capa de q' neuronas, la matriz $\mathbf{W}$ se identifica con los elementos de la ecuación Ecuación 2-2.

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1q'} \\ w_{21} & w_{22} & \cdots & w_{2q'} \\ \vdots & \vdots & \ddots & \vdots \\ w_{q1} & w_{q2} & \cdots & w_{qq'} \end{bmatrix}^T \tag{2-2}$$

En la Ecuación 2-2 los elementos de la matriz corresponden a los pesos de las conexiones, es decir representan el valor de ponderación de la salida de la neurona i de la capa de q neuronas que se dirige a la neurona j de la capa de q' neuronas, esto se denota como w_{ji}. Por ejemplo, el peso w_{35} indica el valor de la ponderación de la salida de la neurona 5 de la capa q que se conecta con la neurona 3 de la capa q'. El diagrama de la Figura **2-2** representa la notación sugerida.

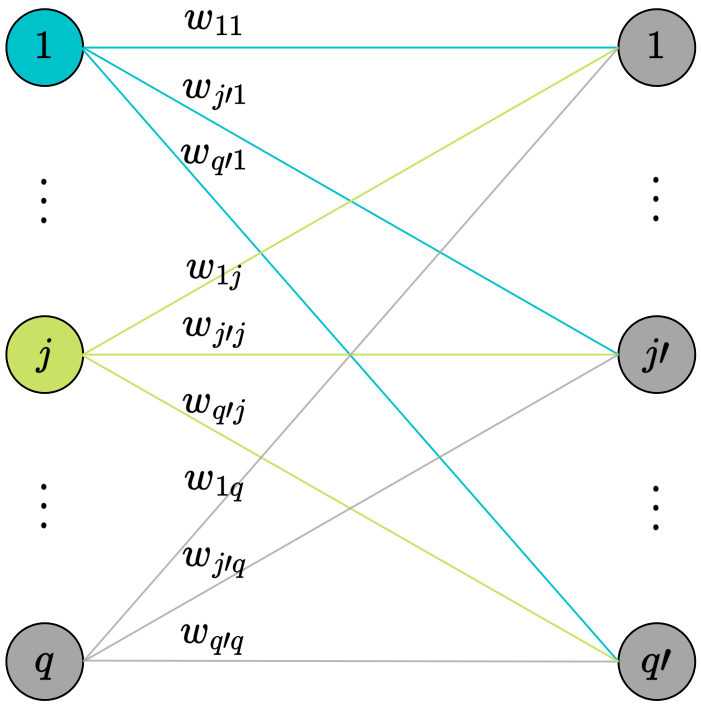


Figura 2-2.: Diagrama de la conexión entre dos capas de una arquitectura convencional de red neuronal con q neuronas en la capa de salida y q' en la capa de llegada.

Considere que se cuenta con una arquitectura convencional de red neuronal con L capas ocultas, denotaremos la función de activación como $\sigma : \mathbb{R} \to \mathbb{R}$. Entonces la salida de la red para algún $\mathbf{x}$ queda determinada por la Ecuación 2-3.

$$\hat{y} = \underbrace{\bar{\sigma}^{(l)}(\mathbf{W}^{(L)}(\bar{\sigma}^{(L-1)}(\mathbf{W}^{(L-1)}\cdots(\bar{\sigma}^{(2)}(\mathbf{W}^{(2)}(\bar{\sigma}^{(1)}(\underbrace{\mathbf{W}^{(1)}\mathbf{x}+\mathbf{b}^{(1)}}_{\text{Transformación afín}}))+\mathbf{b}^{(2)})\cdots+\mathbf{b}^{(L-1)})+\mathbf{b}^{(L)}))}_{\text{Composición de } L \text{ funciones}}. \tag{2-3}$$

La Ecuación 2-3 corresponde a la propagación hacia adelante de alguna entrada $\mathbf{x}$. En la ecuación los superíndices indican el número de la capa. Es importante resaltar que la función $\bar{\sigma}^{(j)}$, en esta versión general, es una función vectorial de la forma $\bar{\sigma} : \mathbb{R} \to \mathbb{R}^{n_j}$, donde n_j serán las neuronas de la capa siguiente j. En el caso que n_j sea mayor que la unidad, entonces la función $\bar{\sigma}^{(j)}$ tendrá la forma $\bar{\sigma}^{(j)} = [\sigma_1^{(j)}, \cdots, \sigma_{n_j}^{(j)}]^T$, donde cada componente es una función de activación $\sigma_i^{(j)} : \mathbb{R} \to \mathbb{R}$, la cual es idéntica para todas las neuronas, por lo que podemos escribir $\bar{\sigma}^{(j)} = [\sigma^{(j)}, \cdots, \sigma^{(j)}]^T$.

Entonces, la red neuronal corresponde a una función $f : \mathcal{X} \to \mathcal{Y}$ que es la composición afín lineal de funciones de activación y tiene la forma funcional descrita por la Ecuación 2-3. El problema de aprendizaje implica encontrar todos los parámetros θ que corresponden a los pesos y bias de cada capa y cada neurona $\theta = \{\mathbf{W}, \mathbf{b}\}$ de manera que dadas las K tuplas de la forma $\{\mathbf{x}_i, y_i\}$, con $\mathbf{x}_i \in \mathcal{X}$ y $y \in \mathcal{Y}$, $\forall i = 1 : K$, se minimice una función de costo que representa la diferencia entre el valor $\hat{y}_i = f(\mathbf{x}_i, \theta)$ y la verdadera etiqueta y_i. Esto se realiza para poder encontrar, con los K datos etiquetados disponibles, los parámetros óptimos $\theta_\star$ para los cuales la función $f(\mathbf{x}, \theta)$ mapea correctamente al espacio $\mathcal{Y}$ datos sin etiquetar.

La función de costo se denota como $l(\hat{y}, y) : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, dependiendo la aplicación l puede tener diferentes formas funcionales. No obstante, independiente de su forma, su objetivo es cuantificar la diferencia entre el valor estimado por la red $\hat{y}$ y la etiqueta verdadera y. Algunas de las funciones de costo mayormente empleadas son: el error cuadrático medio (Ecuación 2-4), la entropía cruzada binaria (Ecuación 2-5), entropía cruzada categórica (Ecuación 2-6); adecuada para problemas de clasificación con C clases con la salida codificada *one-hot*[2], y entropía cruzada categórica dispersa con la misma forma funcional de la Ecuación 2-7 pero para una codificación de la salida en números enteros.

$$l(\hat{y}, y) = \frac{1}{2}||\hat{y} - y||_2^2 \tag{2-4}$$

$$l(\hat{y}, y) = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})) \tag{2-5}$$

[2]La codificación one-hot hace referencia a emplear C neuronas de salida, con C igual al número de clases. En esta codificación las etiquetas de alguna clase c_i se representan como un vector con todas las componentes nulas y valor de unidad en la posición que represente la clase i

$$l(\hat{y}, y) = -\sum_{j=1}^{C}(y_j \log(\hat{y}_j) \tag{2-6}$$

Definida la función de costo a minimizar, el problema de entrenamiento puede formularse como un problema de optimización representado por la expresión 2-7 y con las restricciones de la Ecuación 2-8. Dado que se cuenta con K duplas es necesario encontrar los parámetros que minimicen la función de costo sobre todo el conjunto, por esta razón se minimiza el promedio.

$$\theta_\star = \arg\min_\theta \left(\frac{1}{K} \sum_{i=1}^{K} l(\hat{y}_i, y_i) \right) \tag{2-7}$$

$$\text{s.a. } \hat{y}_i = f(\mathbf{x}_i, \theta), \forall i = 1 : K \tag{2-8}$$

2.1.1. Backpropagation

Los parámetros θ de la red neuronal se obtienen al resolver el problema de optimización planteado en la Ecuación 2-7, sujeto a lo expresado en 2-8. Dada la complejidad de la forma funcional f, la extensión del modelo neuronal, el ruido y la cantidad de datos, no hay una forma analítica de resolver el problema. Uno de los enfoques mayormente empleado en problemas de optimización que no se resuelven analíticamente es el algoritmo del Gradiente descendente ("*Steepest Descent*"). Para encontrar un minimizador de alguna función diferenciable el algoritmo del Gradiente descendente verifica la condición necesaria de primer orden (FONC - *First order necesary condition*), desplazando un punto inicial en la dirección contraria al gradiente de la función a optimizar (o una aproximación de este).

En el contexto de las redes neuronales el proceso iterativo de Gradiente Descendente para establecer los parámetros θ partiendo de algún punto inicial θ_0, puede formularse con la Ecuación 2-9. En la ecuación η, asociado al paso del gradiente en cada iteración, se denomina **tasa de aprendizaje** y ∇l es el gradiente de la función de costo respecto a los parámetros θ.

$$\theta_{i+1} = \theta_i - \eta \nabla l_{\theta_i} \tag{2-9}$$

Para aplicar el método del Gradiente Descendente es necesario contar con la estimación de ∇l_{θ_i} o considerar una aproximación. Este valor resulta complicado de obtener analíticamente,

por lo tanto y como resultado del desarrollo computacional en las décadas de 1970-1990 se platearon algunas formas de realizar esta estimación. En el año 1986 se introdujo el término *backpropagation* [72], los autores lo describen como el proceso de aprendizaje de una estructura neuronal. Este método continua siendo empleado para estimar el gradiente de la función de costo (error cometido) respecto a los pesos.

En el algoritmo de *backpropagation* se fracciona el error total cometido por la red, cuantificado por la función de costo, por la contribución individual de cada unidad neuronal. El algoritmo comienza calculando la derivada parcial de la función de costo respecto a la salida de alguna neurona. Luego, se calcula la derivada parcial de la salida de la neurona previa respecto a la actual. Este proceso continúa a través de la red y finalmente se establece la derivada parcial del error respecto a algún peso w como el producto de las derivadas parciales de entrada-salida a lo largo del trayecto que comienza en la salida y se propaga hacia atrás en la red.

2.1.2. Hiper-parámetros

Antes de iniciar el proceso de aprendizaje en las redes neuronales se necesitan fijar algunos parámetros, ya que estos no se obtienen directamente del proceso de optimización de la función de costo y son de libre elección se conocen como **hiper-parámetros**. A nivel general los hiper-parámetros se pueden agrupar en dos clases, los correspondientes a la arquitectura del modelo y aquellos que se relacionan el proceso de optimización, su elección incidirá en el resultado alcanzado al finalizar el aprendizaje.

Hiper-parámetros de la arquitectura de la red

En este tipo de factores se encuentran aspectos como el número de capas L, el número de neuronas en cada capa, la función de activación σ y las operaciones en la red. Estos elementos definen la complejidad del modelo, una elección inadecuada podría generar modelos sobre-parametrizados o insuficientes para aprender las características de un conjunto de datos.

En el caso de las funciones de activación se considera usar las mismas funciones en las capas ocultas de la red, se podrían considerar mezclas de funciones no obstante, esta perspectiva es de difícil compresión e implementación computacional. La función de la capa final depende del problema de clasificación, la forma en la que es abordado y la función de costo. Por ejemplo, en una clasificación multiclase se emplea la codificación *one-hot*, función de entropía cruzada y funciones de activación sigmoideas, en particular la función logística, que permiten obtener un valor entre $(0, 1)$ que puede ser interpretado como una probabilidad de pertenecer a una clase. En las capas ocultas de la red se pueden considerar funciones de activación diferenciales no lineales, esta ultima característica permite la construcción de fron-

teras de separación no lineales. Generalmente, las funciones mas empleadas son: Rectificador lineal $\text{ReLU}(x) = \text{máx}\{0, x\}$, logística (sigmoidea) $S(x) = (1+e^{-x})^{-1}$ y tangente hiperbólica $\tanh(x) = (e^{x} - e^{-x})/(e^{x} + e^{-x})$. Algunas propiedades de estas funciones se pueden consultar en el Apéndice A.

En la literatura también se menciona el uso de funciones periódicas, sin embargo este enfoque no se ha sido ampliamente explorado y se considera podrían generar dificultades en el proceso de aprendizaje [75]. Las funciones de activación también pueden ser adaptativas y variar sus formas funcionales o sus parámetros de configuración durante el proceso de aprendizaje [76, 77], estas modificaciones pueden mejorar el rendimiento del sistema y generar estructuras flexibles, no obstante su interpretación e implementación computacional puede ser complicada.

Hiper-parámetros del proceso de optimización de la red

El proceso de optimización también requiere definir algunos parámetros, el algoritmo de optimización es uno de estos. Como se mencionó en la sección anterior típicamente el proceso de aprendizaje se realiza con las técnicas del Gradiente Descendente, sin embargo, esta no es la única perspectiva, en la comunidad se ha incursionado en diferentes metodologías con técnicas heurísticas que permiten encontrar los parámetros de la red [78, 79]. En general, los algoritmos de optimización tradicionales continúan preservando la idea general de emplear el gradiente para establecer la dirección de mayor descenso, pero incluyen modificaciones a las reglas de actualización de pesos. Algunos de los algoritmos más empelados son **Adagrad**, **RMSprop**, **Adam**, **Adadelta**.

En en el algoritmo **Adagrad** (*Adaptative Gradient Algorithm*) [80] se disminuye la tasa de aprendizaje en función de los valores históricos del cuadrado del gradiente. Similar a este proceso en **RMSprop** (*Root Mean Square Propagation*) se evita la acumulación excesiva de los valores del cuadrado del gradiente incluyendo un promedio exponencial ponderado, este promedio se pondera según la proximidad de los valores históricos del gradiente. El algoritmo **Adam** [81] puede interpretarse desde el enfoque físico como incluir los factores de inercia y fricción en la optimización. El algoritmo emplea como primer momento el promedio de los gradientes (inercia) y segundo momento la varianza (fricción). Los pesos se actualizan en función del decaimiento exponencial de ambos momentos. En **Adadelta** [82] se realiza una actualización teniendo en cuenta el valor RMS de un promedio exponencial ponderado de los valores cuadrados del gradiente. Este algoritmo de optimización parte de la idea básica de incluir una función de corrección de desajuste de unidades entre los valores de actualización y los parámetros originales.

Otro hiper-parámetro es la tasa de aprendizaje, este valor indica la ponderación del paso en

contra del gradiente (Ecuación 2-9). La elección de este valor es fundamental para el desempeño del proceso de optimización, un valor inadecuado puede conducir a escenarios donde no se logre la convergencia o se puede generar la tendencia a caer en mínimos locales. Los optimizadores, en general, se pueden interpretar como algoritmos que pretenden establecer una tasa de aprendizaje adaptativa. Sin embargo, incluso en estos algoritmos es necesario fijar este valor y las constantes dependiendo la rutina del optimizador. Todos estos valores se consideran hiper-parámetros del proceso de optimización, su elección cambia el resultado y condiciona el desempeño del modelo.

Por otra parte, los algoritmos de optimización basados en gradiente que se resuelven iterativamente requieren en cada iteración usar todo el conjunto de datos disponible para el entrenamiento, esta forma de operación puede ser lenta y computacionalmente costosa. Una manera de mejorar el proceso iterativo es conocida como **Descenso de Gradiente Estocástico** (*Stochastic gradient descent SGD*), algoritmo muy popular, y el más común, en el entrenamiento de redes neuronales [83, 84]. A diferencia del enfoque tradicional, la actualización de los pesos se realiza para cada muestra del conjunto de datos, no sobre su totalidad, cuando todas las muestras han intervenido en el proceso se dice que se cumple una época. El algoritmo 2.1 presenta el proceso realizado en este tipo de optimización.

Algoritmo 2.1: Descenso de Gradiente Estocástico

Entrada:
$\mathbf{X}$ conjunto de datos de entrenamiento,
η tasa de aprendizaje,
N_e número de épocas,
$f(\mathbf{X};\theta)$ modelo red neuronal
Salida: $\theta = \{\mathbf{W}, \mathbf{b}, \}$ parámetros del modelo, pesos y bias
$\theta_0 \leftarrow$ Aleatorio;
para $epoch = 1 : N_e$ **hacer**
  Barajar datos de entrenamiento $\mathbf{X}_s \leftarrow \mathbf{X}$;
  $i \leftarrow 0$;
  para cada $x \in \mathbf{X}_s$ **hacer**
    Calcular gradiente función de costo
      $\hat{y} = f(x;\theta_i)$;
      $\nabla_{l_r} = \nabla_{\theta_i}(l(y,\hat{y}))$;
    Encontrar $\theta_{i+1} \leftarrow \theta_i - \eta\nabla_{l_r}$;
    $i \leftarrow i + 1$;
  $\theta_0 \leftarrow \theta_{i+1}$;
devolver *[θ_0]*

En escenarios con numerosas observaciones en el conjunto de entrenamiento también es ineficiente actualizar los parámetros para todas en cada una de las épocas, como se propone en el SGD. Entonces, una de las variantes consideradas, que preserva una relación adecuada ante el número de observaciones, es la optimización estocástica por "Mini-Lotes" o (*Mini-batch gradient descent*) [83]. En este proceso para cada época se actualizan los parámetros para conjuntos de lotes más pequeños del conjunto de entrenamiento. En el Algoritmo 2.2 se indica la rutina que se considera para realizar el entrenamiento de una red neuronal. Es importante considerar el efecto de las iteraciones, el número de lotes y las épocas. Si se fija el número de lotes entonces se definirá indirectamente la cantidad de elementos sobre los cuales se calculará el gradiente y el numero de iteraciones necesarias para completar una época.

Algoritmo 2.2: Descenso de Gradiente Estocástico por Mini-Lotes

Entrada:
$\mathbf{X}$ Matriz de observaciones datos de entrenamiento,
η tasa de aprendizaje,
N_e número de épocas,
N_b número de lotes,
$f(\mathbf{X};\theta)$ modelo red neuronal
Salida: $\theta = \{\mathbf{W}, \mathbf{b}\}$ parámetros del modelo pesos y bias
$\theta_0 \leftarrow$ Aleatorio;
para $epoch = 1 : N_e$ **hacer**
  Barajar datos de entrenamiento $\mathbf{X}_s \leftarrow \mathbf{X}$;
  Generar los N_b lotes del conjunto $\mathbf{X}_s$;
  $i \leftarrow 0$;
  para $batch = 1 : N_b$ **hacer**
    Calcular gradiente función de costo
      $\hat{y} = f(\mathbf{X}_s[batch]; \theta_i)$;
      $\nabla_{l_r} = \nabla_{\theta_i}(l(y, \hat{y}))$;
    Encontrar $\theta_{i+1} \leftarrow \theta_i - \eta\nabla_{l_r}$;
    $i \leftarrow i + 1$;
  $\theta_0 \leftarrow \theta_{i+1}$;
devolver *[θ_0]*

Dado que el aprendizaje es un proceso iterativo es necesario fijar los parámetros iniciales θ_0. La elección de este parámetro puede afectar el algoritmo atrapándolo en minimizadores locales o situándose muy lejos del mínimo. En general se considera que su valor sea aleatorio y no se recomienda definirlo con valores nulos o con un valor similar para todos, esto estancaría el proceso de aprendizaje.

2.2. Redes neuronales artificiales residuales (ResNet)

En el año 2015 el equipo de *Microsoft Research Asia* presentó la red neuronal residual ResNet. En esta configuración la topología de red incluía un salto entre conexiones de capas, esto implica que la entrada y salida de la red, para algún bloque, se suman antes de ingresar al siguiente [32]. Esta arquitectura logró resultados bastante favorables obteniendo el primer lugar en el desafío ImageNet Large Scale Visual Recognition Challenge 2015[3] (ILSVRC 2015) con un error por debajo de la percepción humana.

La Figura **2-3** presenta un diagrama de la estructura de los bloques residuales que componen los modelos ResNet. Es importante aclarar que los bloques al interior del residual pueden contener diferentes operaciones, por ejemplo, convoluciones, normalizaciones, pooling, entre otras. Sin embargo, la dimensión de la entrada y salida del bloque residual deben ser iguales para que la operación se pueda realizar.

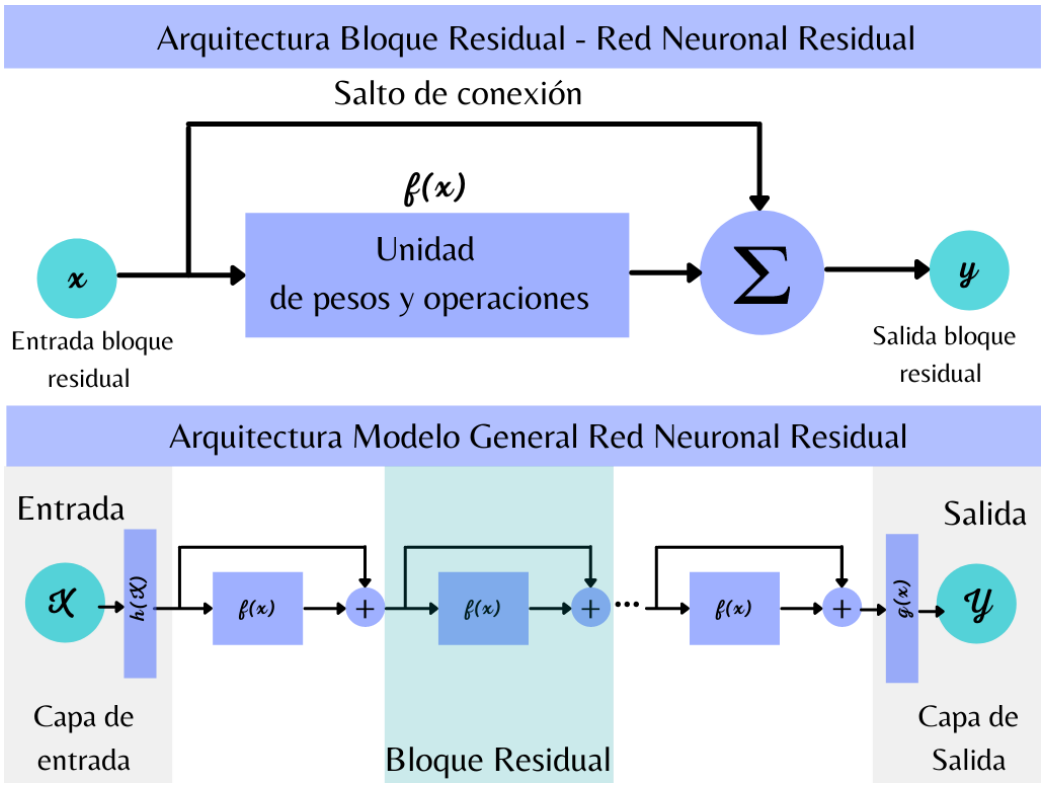


Figura 2-3.: Diagrama de la arquitectura de red neuronal residual. a) Arquitectura bloque residual. b) Arquitectura modelo general ResNet.

[3] https://image-net.org/challenges/LSVRC/2015/

Matemáticamente, en la estructura ResNet la salida para alguna capa j queda determinada por la Ecuación 2-10. La función h representa las operaciones realizadas en el bloque residual, esta función depende de los parámetros $\theta^{(j-1)}$ y $\mathbf{x}^{(j-1)}$, con $\mathbf{x} \in \mathcal{X}$ y $\mathcal{X} \subseteq \mathbb{R}^n$. La función h mapeará al mismo espacio que pertenece $\mathbf{x}$, esto es fundamental para que la operación de adición pueda ser realizada. En la Ecuación 2-10 $\theta^{(j-1)}$ son todos los parámetros que se deben estimar.

$$\mathbf{x}^{(j)} = \mathbf{x}^{(j-1)} + h(\mathbf{x}^{(j-1)},\ \theta^{(j-1)}) \tag{2-10}$$

En el modelo mas simple de la arquitectura ResNet se considera que la operación al interior del bloque residual corresponde a $\bar{\sigma}(\mathbf{W}\mathbf{x}+\mathbf{b})$. En este caso $\bar{\sigma}$ es la función de activación de las neuronas, $\mathbf{W}$ la matriz de pesos de las conexiones y b el valor de bias, entonces la Ecuación 2-10 se transforma en la Ecuación 2-11. Asumiendo que $\mathbf{x} \in \mathcal{X}$ con $\mathcal{X} \subseteq \mathbb{R}^n$, entonces la función $\bar{\sigma}$ de la Ecuación 2-11 es una función de la forma $\bar{\sigma} : \mathcal{X} \to \mathbb{R}^n$, y esta representada por la Ecuación 2-12, donde $\sigma_i : \mathcal{X} \to \mathbb{R}$, $\forall i = 1 : n$, es una función de activación convencional. En la Ecuación 2-12 los pesos $\mathbf{w}_i \in \mathbb{R}^n$ y los bias $b_i \in \mathbb{R}$, con $i = 1 : n$.

$$\mathbf{x}^{(j)} = \mathbf{x}^{(j-1)} + \bar{\sigma}(\mathbf{W}^{(j-1)}\mathbf{x}^{(j-1)} + \mathbf{b}^{(j-1)}) \tag{2-11}$$

$$\bar{\sigma} = [\sigma_1(\mathbf{w}_1^{T(j-1)}\mathbf{x}^{(j-1)} + b_1^{(j-1)}), \cdots, \sigma_n(\mathbf{w}_n^{T(j-1)}\mathbf{x}^{(j-1)} + b_n^{(j-1)})]^T \tag{2-12}$$

Como se puede advertir en el bloque residual (operaciones y capas antes dentro del salto residual) s la salida de la red $\mathbf{x}^{(s)}$ estará en el mismo espacio que la entrada para la primera capa residual $\mathbf{x}^{(1)}$. La situación mencionada permite analizar que a diferencia de otras estructuras a menos que el espacio de las etiquetas $\mathcal{Y}$ coincida con el espacio $\mathcal{X}$ se podrá definir la función de costo tomando la salida en la ultima capa residual $\mathbf{x}^{(s)}$. Generalmente los espacios de los datos de entrada y las etiquetas son diferentes, entonces, la ResNet debe acondicionarse incluyendo una o mas capas que tomen el último estado $\mathbf{x}^{(s)}$ y lo mapeen al espacio de etiquetas, denotaremos esta función como $g : \mathbb{R}^n \to \mathcal{Y}$.

Teniendo en cuenta lo mencionado, en esta arquitectura particular el problema de aprendizaje, formalizado como la minimización de la Ecuación 2-7 sujeto a las restricciones de la Ecuación 2-8, se modifica como se observa en la Ecuación 2-13, sujeto a las restricciones de la Ecuación 2-14.

$$\theta_\star = \arg\min_\theta \left(\frac{1}{K} \sum_{i=1}^{K} l\left(g\left(\mathbf{x}_i^{(s)}\right), y_i\right) \right) \tag{2-13}$$

$$\text{s.a. } \mathbf{x}_i^{(j)} = \mathbf{x}_i^{(j-1)} + h(\mathbf{x}_i^{(j-1)},\ \theta^{(j-1)}),\ \forall i = 1:K, j = 1:s \tag{2-14}$$

Las ecuaciones planteadas modelan una arquitectura que corresponde a la secuencia de s capas residuales y una capa final modelada por la función g. No obstante, las estructuras neuronales que involucran bloques residuales pueden tener diferentes configuraciones, se pueden combinar algunas capas completamente conectadas dentro del bloque residual, incluir ponderaciones u operaciones en la conexión de salto, generar capas convolucionales fuera de los saltos residuales o agregar capas paralelas. Estas configuraciones son difíciles de modelar con expresiones matemáticas, sin embargo las ecuaciones 2-13 y 2-14 son adecuadas para un modelo con bloques residuales que únicamente incluyen una capa completamente conectada y un salto. Incluso, dado que el bloque residual modelado por al función h puede contener operaciones de convolución, *pooling*, normalización, entre otras, el modelo más sencillo es asumir que únicamente se realizan las operaciones definidas por la Ecuación 2-11.

2.3. Problema de sobreajuste y subajuste

Pese a la notable evolución de las redes neuronales artificiales, su diversificación en modelos y arquitecturas, la cantidad de aplicaciones en las que son empleadas y la facilidad computacional de implementación, estas estructuras continúan presentando problemas de sobreajuste [9]. Esta cuestión ha sido ampliamente discutida y se considera uno de los principales riesgos de los algoritmos de aprendizaje de máquina. En el año 1997 los autores Lawrence, Giles y Tsoi expusieron que esta problemática era de los aspectos más relevantes a corregir, sin embargo, en su artículo también encontraron que típicamente no había una solución óptima [85].

En el desarrollo de procesos de clasificación típicamente el conjunto de K duplas se dividen en tres subconjuntos denominados entrenamiento (*train*), validación (*validation*) y prueba (*test*). El mayor porcentaje de los datos se asigna para el entrenamiento y se destina una cantidad menor para validación y prueba. Los datos de entrenamiento se emplean para realizar el proceso de aprendizaje y encontrar los parámetros θ que minimicen el promedio de la función de costo en este conjunto. Al mismo tiempo y en cada época (o iteración) se emplea el conjunto de validación para determinar el desempeño del clasificador con datos que no han sido involucrados en el entrenamiento. El modelo resultado del proceso de entrenamiento se evalúa al finalizar con los datos de prueba, esto brinda una medida del ajuste realizado. En algunas aplicaciones es usual dividir el conjunto únicamente en datos para entrenamiento y prueba, estos últimos se emplean para la evaluación iterativa y final del modelo.

El sobreajuste se puede relacionar con la incapacidad de generalización del modelo neuronal, esta condición se puede analizar al comparar el desempeño entre los resultados logrados con los conjunto de entrenamiento, validación y prueba. Las curvas de aprendizaje ayudan a establecer esta comparación, estas gráficas describen las métricas de desempeño del clasificador. Se espera que el comportamiento para el conjunto de entrenamiento se replique en el de validación, cuando esta condición no sucede se esta ante un problema de sobreajuste o subajuste. En el primer caso, el desempeño alcanzado en el entrenamiento es superior que los resultados en validación, en consecuencia se espera que el modelo no generalice bien y su buen desempeño sea resultado del ajuste excesivo a los datos que se usaron para resolver el problema de aprendizaje. En el segundo fenómeno, el desempeño del clasificador en el conjunto de entrenamiento resultara insuficiente y esto implicaría que el ajuste es inadecuado, en consecuencia la inferencia del modelo a datos no visto también será carente de buenos resultados.

Formalmente se podría definir el problema de sobreajuste de la siguiente manera. La población de los datos representada por X sigue alguna distribución de probabilidad P_X desconocida. Considere que todos los individuos de la población X tienen asignada una etiqueta que identifica su clase, y se cuenta con una función de costo l que cuantifica la diferencia entre dos etiquetas asignadas a un individuo de la población X (ver Sección 2.1), particularmente la diferencia entre la etiqueta real del individuo y la etiqueta estimada por algún modelo neuronal. Dada la aleatoriedad y naturaleza de los datos X para toda esta población se definiría la función de costo como el valor esperado sobre todo el conjunto X, como se presenta en la Ecuación 2-15. La notación y_X corresponde a la etiqueta real para el individuo de la población y $\hat{y}_X = f(X, \theta)$ es el valor estimado por la red neuronal.

$$\mathcal{L}(y, \hat{y}) = \mathrm{E}_{X \sim P_X}[l(y_X, \hat{y}_X)]. \tag{2-15}$$

En la practica solo se tiene una muestra finita de K pares (dato y etiqueta) entonces la Ecuación 2-15 se aproxima con el promedio sobre la muestra (conjunto de datos), de tal manera que

$$l_K(y, \hat{y}) = \frac{1}{K} \sum_{i=1}^{K} [l(y_i, \hat{y}_i)], \tag{2-16}$$

en esta ecuación el valor $\hat{y}_i = f(\mathbf{x}_i, \theta)$. Como se analizó en Sección 2.1 el objetivo es minimizar la función de costo, lo ideal sería realizar el proceso sobre la población, sin embargo se espera que la minimización de Ecuación 2-16 permita encontrar los parámetros que generalicen a toda la población. Entonces, el proceso de aprendizaje busca establecer que $\min_{\theta} l(y, \hat{y})$ sea equivalente a $\min_{\theta} \mathcal{L}(y, \hat{y})$. Es decir, los minimizadores logrados al resolver el problema de

aprendizaje para la muestra (conjunto de K datos) denotados como $\theta_\star$ deben ser buenos estimadores de los minimizadores que resultarían de optimizar la Ecuación 2-15, denotados como $\Theta_\star$. Esto se puede ver como

$$\min_\theta \, l_K(y, \hat{y}) \equiv \min_\theta \, \mathcal{L}(y, \hat{y}). \tag{2-17}$$

Entonces se espera que

$$\arg\min_\theta \left(l_K(y, \hat{y})\right) \approx \arg\min_\theta \left(\mathcal{L}(y, \hat{y})\right) \tag{2-18}$$

$$\theta_\star \approx \Theta_\star. \tag{2-19}$$

Cuando se logra una buena minimización de la función de costo para el conjunto de datos, pero no se satisface la Ecuación 2-19 el modelo estará sobreajustado. En la práctica los parámetros de la población son desconocidos, por lo tanto $\Theta_\star$ no se puede contrastar con $\theta_\star$. Sin embargo, si el desempeño del clasificador difiere entre el conjunto de entrenamiento y los conjuntos de datos que no formaron parte del ajuste de $\theta_\star$ (*validation*, *test*), se puede concluir que el proceso de minimización de la función de pérdidas del conjunto de datos no es equivalente a la minimización sobre la población, es decir no se mantiene la Ecuación 2-17, en consecuencia tampoco se satisface Ecuación 2-19.

2.3.1. Causas identificadas y métodos de solución

Con el problema del sobreajuste identificado múltiples estudios han buscado, desde diferentes enfoques, comprender sus causas y brindar soluciones [9, 11]. Una de las estrategias empleadas es conocida como **parada temprana** (*Early Stopping*), la idea es detener el entrenamiento cuando la estructura neuronal alcance un buen desempeño. Esta perspectiva teórica se fundamenta en que después de una cierta época, el clasificador logra una buena respuesta evitando el subajuste, no obstante continuar el entrenamiento hará que no reciba mejoras significativas y por el contrario puede generar un sobreajuste en los datos. Los métodos inspirados en esta estrategia son diversos [86, 87], sin embargo, esta solución requiere contar con un conjunto de validación apropiado y no garantiza resolver el problema [88]. Actualmente, se han estudiado otras versiones del algoritmo de parada temprana, por ejemplo en [89] los autores usan medidas estadísticas del comportamiento del gradiente de la función de pérdidas para establecer un punto óptimo de parada. Esta técnica no requiere conjunto de validación, sin embargo, como los autores lo mencionan pese a que se cuente con un conjunto de entrenamiento "grande" finalmente solo se tiene una muestra finita de una población que distribuye con parámetros desconocidos.

Una de las causas a las que le atribuyen el problema de sobreajuste es la falta de un conjunto considerable y representativo de datos de entrenamiento y validación. Desafortunadamente en muchas aplicaciones el conjunto de datos es costoso de obtener, un ejemplo de esta situación son las imágenes médicas. Una solución es incrementar y mejorar las características de este conjunto, esta técnica se conoce como **Aumento de Datos** (*Data Augmentation*). Algunos estudios relacionan esta metodología con la disminución de sobreajuste y subajuste [90, 91]. No obstante, esta técnica se condiciona a la aplicación que se desea desarrollar, además los datos generados sintéticamente siempre llevan información de los originales, por lo tanto no se consideran nuevas observaciones. Además, en este proceso se podrían replicar algunas características o patrones no deseados, por ejemplo ruidos.

El sobreajuste esta ligado a las características del conjunto de datos, por ejemplo, algunos estudios mencionan la presencia del problema como resultado de un procesamiento inadecuado del conjunto de datos. En estas investigaciones se considera que la presencia de ruidos, patrones no deseados, valores atípicos o datos sesgados puede repercutir negativamente en el proceso de aprendizaje y generar una estructura con tendencia al sobreajuste. Por ejemplo, en el estudio [92] los autores señalan que la presencia de datos atípicos, denominados *border instances*, son la causa del sobreajuste. Esta afirmación la relacionan con la interpretación del proceso de aprendizaje de los seres humanos, cuando los ejemplos son atípicos o confusos se requerirá mayor tiempo (mas épocas) para aprender, incrementar el tiempo (épocas) terminará fomentando el fenómeno de sobreajuste.

En la misma linea de pensamiento, otras investigaciones exponen que el problema surge cuando se ingresan datos de alta dimensional y no se realiza una efectiva selección de características o un adecuado proceso de reducción de dimensión [37, 38, 93, 94]. Actualmente, el desarrollo computacional permite trabajar con datos que se encuentran en espacios de alta dimensión, sin embargo el constante uso de las redes neuronales, y en general de cualquier clasificador, como herramientas de tipo *black box* permite eludir la interpretación del proceso e incita a trabajar con conjuntos de datos que presentan un elevado numero de características de las cuales no se cuantifica su aporte y se desconoce su significado.

Una técnica que también se ha explorado consiste en desconectar aleatoriamente, con alguna distribución de probabilidad, algunas neuronas. Este método conocido como ***Dropout*** ayuda a controlar la complejidad del modelo, sin embargo, implica la necesidad de un entrenamiento con mayores épocas, lo cual puede resultar contraproducente [95, 96]. Se han investigado algunos modelos más robustos de *Dropout*, por ejemplo, perspectivas adaptativas [97], técnicas controladas de desconexión de neuronas [98], interpretaciones bayesianas [99], entre otras. El gráfico presentado en la Figura **2-4** ilustra una red convencional y el método de *Dropout*.

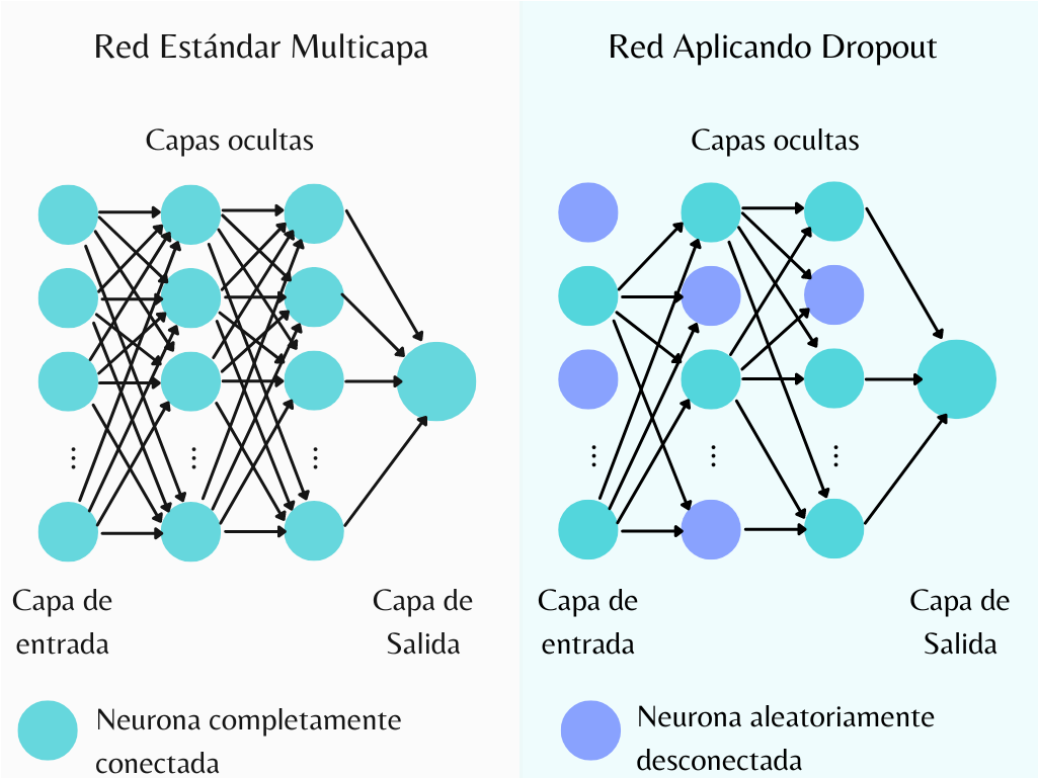


Figura 2-4.: Diagrama de interpretación del método *Dropout*. (izquierda) Red neuronal estándar multicapa. (derecha) Red neuronal aplicando método de *Dropout*.

Otra alternativa ampliamente explorada es la **regularización** de la red neuronal artificial. Este procedimiento busca restringir la complejidad del modelo penalizando los pesos y forzando a que algunos pesos no relevantes para el aprendizaje decaigan a cero [12]. Esta regularización también puede analizarse como una restricción que fuerza a que la norma de los pesos se mantenga por debajo de un umbral. La forma de realizar este proceso es añadir al problema de aprendizaje el factor de penalización relacionado alguna norma de los pesos, cuando se emplea el valor absoluto de la suma de todos los pesos nos encontramos con la regularización tipo L1. También se puede usar como factor la suma del cuadrado de todos los pesos, denominado regularización L2 [100]. Cada enfoque tiene sus consideraciones, por ejemplo, la regularización L2 puede ser empleada en estructuras de datos que exhiban alta no linealidad, sin embargo, es sensible a valores atípicos, situación que es contraria en la regularización L1.

De forma general, la ecuación que representa la regularización se puede analizar en la expresión Ecuación 2-20. La función l denota la función de costo convencional (ver Sección 2.1), el término $\Omega(\mathbf{W})$ corresponde a una función que penaliza la complejidad del modelo, ya que

la regularización solo se realiza sobre los valores de las matrices de pesos se desprecia la dependencia del bias y λ es el factor que pondera la penalización.

$$l_r(y, \hat{y}) = l(y, \hat{y}) + \lambda\Omega(\mathbf{W}) \tag{2-20}$$

Al considerar la regularización L2, denominada en algunas investigaciones como *Weight Decay*, la función $\Omega(\mathbf{W}) = \frac{1}{2}||\mathbf{W}||_2^2$, corresponde a la suma de todos los pesos $w_{i,t}^j$ en la posición i, t en alguna capa j. En la regularización L1 la función $\Omega(\mathbf{W})$ tendrá la forma $||\mathbf{W}||_1$. En cualquier caso dependiendo el proceso de entrenamiento el valor de λ debe ser escalado por el número de datos que se consideraran en una iteración del algoritmo.

La filosofía de las regularizaciones es restringir la complejidad del modelo, se considera que un modelo sobre parametrizado, podría conducir al sobreajuste. En general y analizando la Ecuación 2-20 se puede considerar que al optimizar la función de costo regularizada se busca obtener un clasificador con buenos resultados y menor complejidad. Según se menciona en el estudio [101], este postulado se puede ver como:

$$\text{Costo} = \text{Error de clasificación} + \lambda \text{ Complejidad del modelo.} \tag{2-21}$$

Restringir la complejidad del modelo también se relaciona con el **principio de parsimonia**, "en igualdad de condiciones la explicación más sencilla suele ser la más probable", traducido al contexto de las redes neuronales se podría interpretar como que el modelo de menor complejidad presentará mejor capacidad de generalización. En la literatura se encuentra que algunos autores proponen estimar la complejidad del modelo para establecer su capacidad de generalización en un entorno comparativo. Por ejemplo, en [102] se determinó una forma para establecer la complejidad del modelo a través de la medida denominada *Kolmogorov Growth*. Una vez establecido este valor los autores proponen una regularización para minimizar la medida mientras se realiza el aprendizaje.

Otra medida de la complejidad del modelo es la capacidad de *Vapnik-Chervonenkis*, conocida como *CV dimension*, este valor representa el cardinal del máximo de conjuntos que el modelo podrá separar. Esta medida es la inspiración del estudio [26] en el cual se propone la estimación de la complejidad del modelo con la característica denominada *Geometric Complexity*. Como se puede advertir son diferentes perspectivas que enfocan en medir y controlar la complejidad de una arquitectura de red para mejorar su rendimiento y evitar problemáticas como el sobreajuste.

La problemática del sobreajuste también se ha analizado desde la compensación que el modelo debe tener entre el "sesgo" y la "varianza", típicamente conocido como *bias variance tradeoff*. Al incrementar la complejidad del modelo se espera que el sesgo disminuya, es decir se ajusten mejor los datos de entrenamiento, sin embargo, la varianza presenta una relación inversa. Esta perspectiva ha cambiado en los últimos años, se ha observado que en redes sobre parametrizadas existe la tendencia a mejorar la generalización [103, 104, 105, 106], estas nuevas investigaciones cuestionan que la dicotomía sesgo varianza se continúe cumpliendo en modelos de alta dimensión. Los resultados encontrados exceden el alcance de esta investigación, por lo tanto no serán considerados. No obstante, esta nueva perspectiva permite analizar el dilema de *bias variance tradeoff* en modelos sobre parametrizados y con esto mejorar la capacidad de generalización. Es importante resaltar que, de ser comprobada la hipótesis de que la sobre parametrización mejora la facultad de generalización se requerirá una suficiencia computacional para entrenar los modelos.

Usualmente la compensación entre el sesgo y la varianza se ha abordado desde el enfoque de la selección óptima del modelo neuronal. La metodología consiste en proponer diferentes arquitecturas de red variando los hiper-parámetros[4]. Por ejemplo, analizar modelos incrementando el número de capas ocultas, o el número de neuronas. También se puede pensar en las variaciones de los hiper-parámetros asociados con el entrenamiento, tasa de aprendizaje, optimizador, *batches*, etc.

Los modelos construidos se entrenan y se selecciona el de mejor desempeño. Una de las formas para evaluar el desempeño del clasificador es la técnica denominada **validación cruzada** (*cross validation* CV) [107]. Esta técnica tiene algunas variantes pero en general la metodología requiere dividir aleatoriamente el conjunto de datos de entrenamiento en K_f "pliegues", fracciones del conjunto original. A continuación se realiza el entrenamiento con $K_f - 1$ particiones y se valida en el pliegue restante. El desempeño alcanzado se mide a través del promedio de las métricas en cada entrenamiento, la desviación estándar de estos valores permitirán conocer en un cierto sentido la posible variación del modelo. Teniendo en cuenta estos resultados se selecciona el modelo con mejores métricas promedio y con menor desviación estándar.

Por otra parte, el análisis matemático motivó un nuevo enfoque, se considera que los modelos robustos a pequeñas variaciones de los datos de entrada pueden generalizar de forma adecuada, en consecuencia exhibirán menor tendencia al sobreajuste. La propiedad de **continuidad de Lipschitz de la red neuronal** [23] es la característica que relaciona la magnitud de las

[4] A nivel general esta metodología se aplica para seleccionar el modelo variando el tipo de clasificador, por ejemplo se pueden construir un modelo de clasificador Bayesiano, una red neuronal, una maquina de soporte de vectores y efectuar la comparación.

variaciones en la entrada con el acotamiento de las salidas. La idea general del planteamiento de las investigaciones en esta área es acotar la constante de Lipschitz de la red o forzar que la red tenga un valor deseado de la constante [25, 24, 108, 109]. Los estudios de este tipo presentan algunas dificultades, por ejemplo, encontrar el valor preciso de la constante de Lipschitz de la función que caracteriza una determinada arquitectura de red es un problema NP-Hard, esta contrariedad ha motivado algoritmos de cálculo que se aproximen al valor buscado [52, 53]. Por otra parte, la modificación teórica que sugieren este tipo de estudios implica la ejecución de algoritmos computacionales complejos, lo que afectaría el tiempo de entrenamiento de los modelos y la capacidad de extender los métodos en arquitecturas y datos de alta dimensión.

La continuidad de Lipschitz de la red neuronal esta ligada con la complejidad del modelo. En el estudio [26] los autores proponen una noción para estimar la complejidad de la arquitectura, denominada *Geometric Complexity* (GC). En su estudio se encontró que esta cantidad está acotada por la cota de Lipschitz de la red, de hecho esta ultima se puede pensar como una estimación de la complejidad. Además se identificó, en un modelo lineal, que la regularización L2 es análoga a realizar una regularización explícita GC y que restringir la cota de Lipschitz contraerá el valor GC.

Las condiciones y métodos mencionados son aplicables a cualquier arquitectura de red, evidentemente en cada nueva configuración o variación de la red se modifica su desempeño, el proceso de entrenamiento, la aplicación que mejor resuelve, entre otras características. Sin embargo el sobreajuste puede analizarse, desde el enfoque general, dentro de los estudios mencionados. Particularmente en la red neuronal residual (ResNet) también se puede presentar el sobreajuste como consecuencia de alguna de las causas descritas. De forma análoga, los métodos mencionados se pueden adaptar sin mayor inconveniente a esta arquitectura.

La variación fundamental de las redes neuronales residuales, en comparación con modelos estándar, es el salto entre capas [32]. Esta característica impacta en la propiedad de generalización, computacionalmente se ha observado que las ResNet logran mayor generalización que modelos convencionales y es posible construir modelos profundos (más capas) con menor tendencia al sobreajuste. En el salto residual, al sumar la salida de una capa con la entrada de la misma se preserva, en un cierto sentido, la entrada. Esto último es útil, por ejemplo, si una capa sufre desvanecimiento del gradiente y genera salidas casi nulas, este fenómeno se propagaría en una red convencional, sin embargo, en el modelo residual no se reproduciría el desvanecimiento.

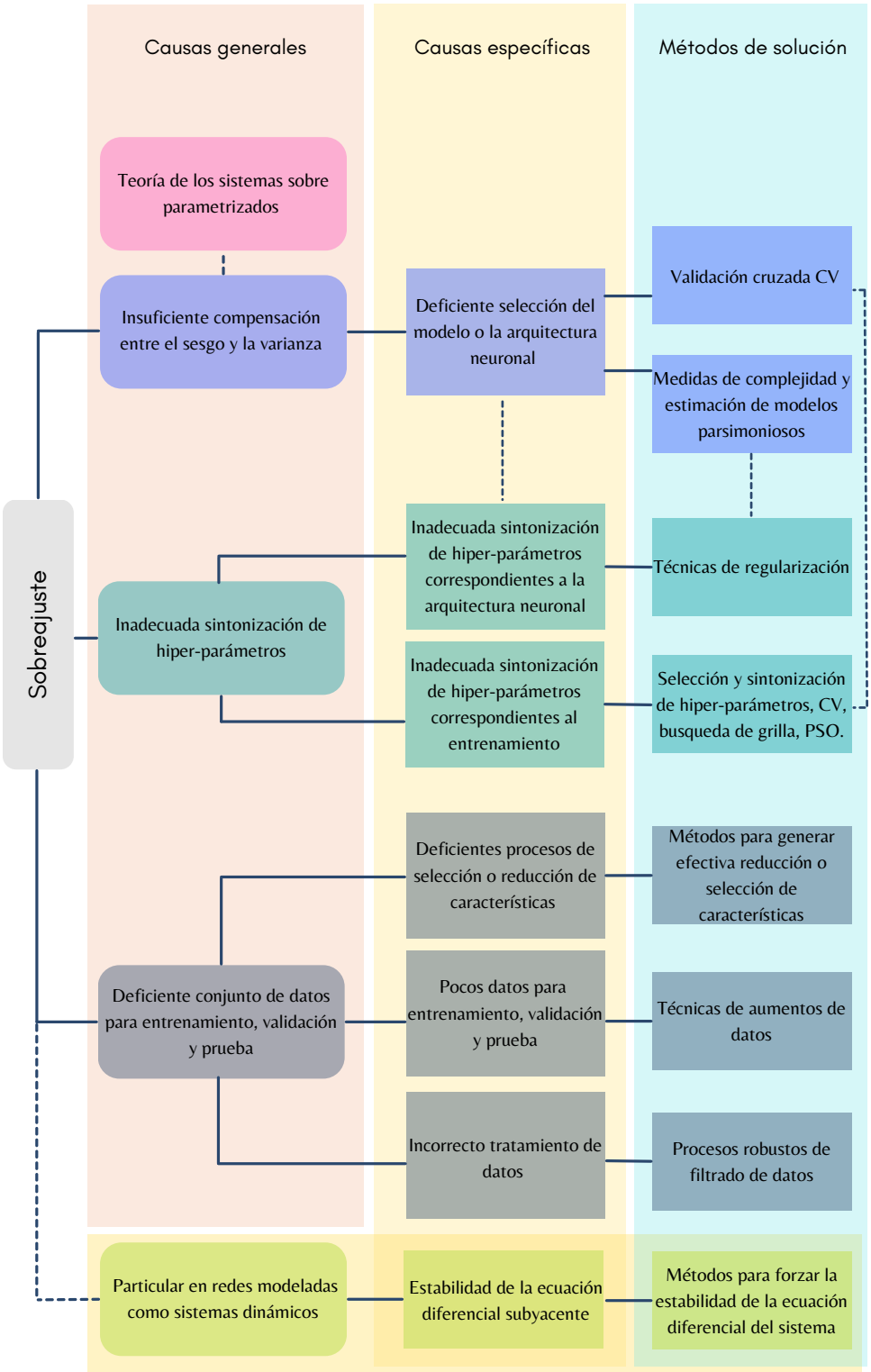


Figura 2-5.: Diagrama de síntesis de algunas causas y métodos de solución del problema de sobreajuste en redes neuronales artificiales.

Las arquitecturas residuales, además de lo descrito anteriormente, también motivaron un nuevo enfoque para el análisis teórico de las redes, la representación en tiempo continuo [13]. El postulado general de esta teoría es que las redes residuales se pueden interpretar como una discretización de una ecuación diferencial ordinaria. Este enfoque permite explorar variedad de perspectivas matemáticas para relacionar propiedades de las dinámicas que sugiere la EDO con características del aprendizaje. En cuanto al sobreajuste se considera que las arquitecturas "estables" previenen este fenómeno [20, 21], esto implica que existe una relación latente entre la estabilidad de la ecuación diferencial que representa el modelo neuronal y su capacidad de generalización. Un análisis más detallado de este enfoque se presenta en la Sección 4.2, incluida como una perspectiva teórica relevante de ser trabajada en investigaciones futuras.

Como se puede notar, particularizado a la arquitectura residual o visto desde el enfoque general, son diferentes las perspectivas empleadas para analizar y proponer una solución al problema de sobreajuste. Incluso, se han considerado técnicas menos convencionales, por ejemplo, diseños de arquitecturas de redes robustas [110], métodos que adaptan enfoques de la teoría física [111] y se ha pensado en combinaciones de varias técnicas [9]. No obstante, el problema del sobreajuste continúa abierto a diferentes aportes, aún no existe una solución general ni un estándar que se pueda emplear independiente de la arquitectura, de los datos de entrada y de la aplicación.

2.3.2. Experimentos computacionales

En esta sección se proponen experimentos computacionales para analizar y reproducir algunos escenarios que conducirían al **sobreajuste** de una red neuronal residual.

Experimento A

En este experimento se trabajó con el conjunto de datos MNIST[5] [112] de imágenes de la escritura manual de los dígitos (0 a 9). El conjunto de datos esta dividido en 60,000 elementos para el entrenamiento y 10,000 para prueba. Cada imagen del conjunto tiene una dimensión de 28×28 píxeles y están únicamente en un canal de color. La Figura **2-6** es una representación de algunos elementos de este conjunto de datos. Para el experimento se tomaron únicamente 10,000 datos del conjunto de entrenamiento, esto con el fin de tener un número menor de datos y que se caiga más fácil en escenarios de clasificadores sobreajustados.

[5]Base de datos modificada del Instituto Nacional de Estándares y Tecnología (NIST)

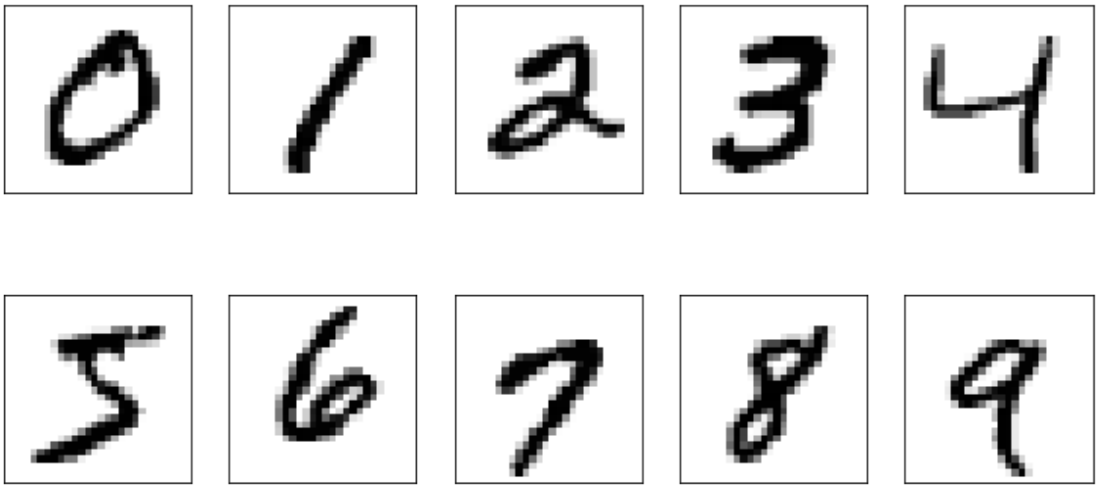


Figura 2-6.: Representación gráfica de algunas imágenes de los dígitos 0 al 9 de la base de datos MNIST. Fuente: a partir de [112].

Para el desarrollo del experimento se consideró entrenar 6 modelos variando en cada uno el número de capas residuales iniciando en 2, finalizando en 12 y con incrementos de 2 capas. En cada entrenamiento se efectuó la metodología de validación cruzada con el método de K-fold para 5 pliegues. Esta estrategia no se realizó para sintonizar ningún hiper-parámetro, si no para establecer métricas en un escenario repetitivo. En cada pliegue se realizó la prueba con las 10,000 muestras incluidas en el conjunto de *test*. La codificación y simulación se realizo en el entorno *Google Colaboratory*[6] en el lenguaje de programación *python*. En la implementación utilizaron alguna clases, funcioens y rutinas de la librería *TensorFlow*[7].

En cuanto a los hiper-parámetros relacionados con el entrenamiento, como optimizador se eligió el Gradiente Estocástico (SGD), dispuesto para recibir 100 lotes por iteración y con un número de 60 épocas. La tasa de aprendizaje se fijó en 0,01, no se consideró añadir parámetros adicionales como *momentum* o activación de la subrutina *nesterov*[8]. En la arquitectura del modelo se incluyó una capa inicial completamente conectada de 20 neuronas, conectada a los saltos residuales, seguido por una capa final completamente conectada de 10 neuronas. La capa final se puede analizar como una transformación afín que mapea del espacio de los saltos residuales al espacio de las etiquetas con codificación *one-hot*. La arquitectura mencionada se presenta en la Figura **2-7**.

[6] https://colab.research.google.com/

[7] https://www.tensorflow.org/

[8] Los elementos mencionados corresponden a modificaciones del algoritmo SGD en las cuales se incluyen variaciones en la regla de actualización de los pesos. La información detallada de estos parámetros se encuentra en la documentación https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/SGD

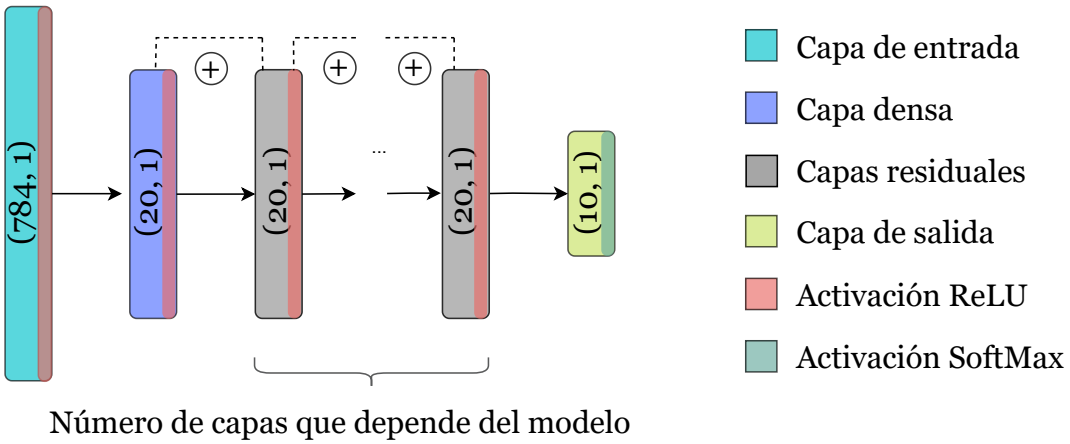
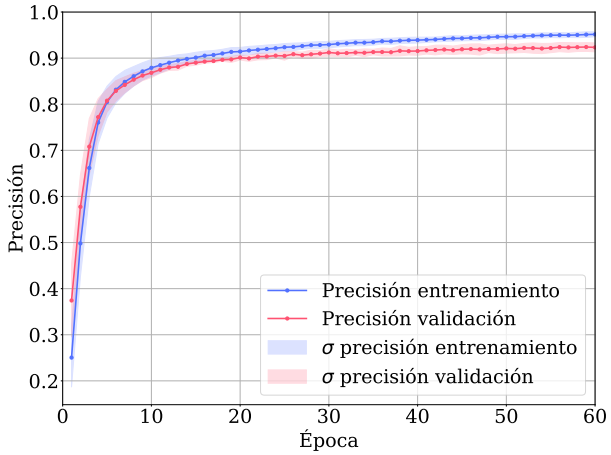


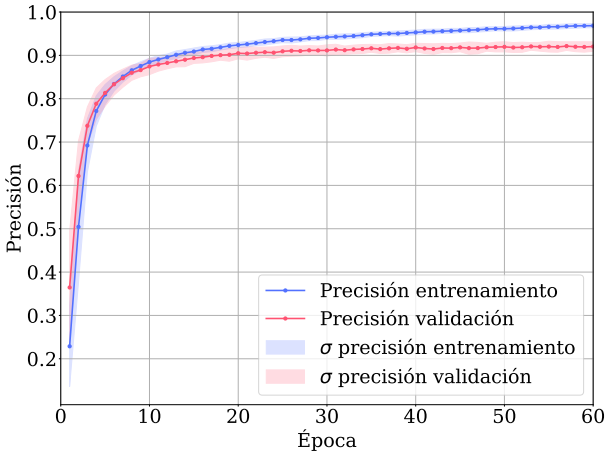
Figura 2-7.: Diagrama de la arquitectura del modelo neuronal implementado en los experimentos.

Los resultados de la curva de aprendizaje a través de las épocas se presentan en la Figura **2-8**. Como se puede observar a medida que el modelo incrementa su complejidad añadiendo capas residuales la precisión lograda en el entrenamiento también aumenta. Contrario a este resultado se puede notar que la curva de validación se queda por debajo de la entrenamiento y la distancia es más pronunciada a medida que se incrementan las capas. La variabilidad del modelo también va cambiando, en los modelos más simples se presenta una desviación estándar pequeña, sin embargo en los modelos con mayor complejidad se observa una dispersión mayor. En las gráficas se denota la desviación con σ y corresponde al área sombreada en color azul para el entrenamiento y rojo para validación.

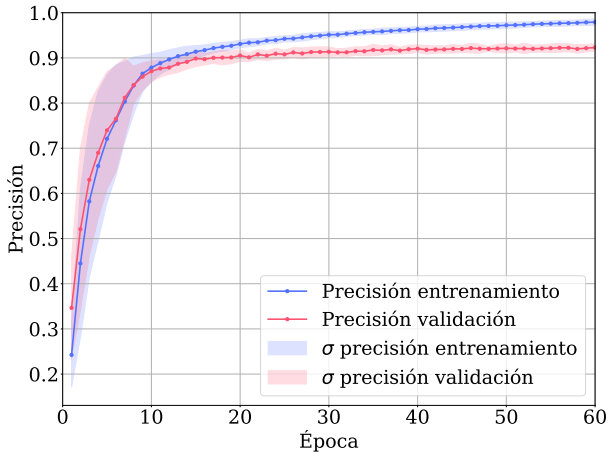
El comportamiento para la función de costo presentada en la Figura **2-9** es similar, a medida que aumenta el número de capas residuales la diferencia entre entrenamiento y validación crece, igual que la dispersión de los resultados. Estos resultados indican que los modelos con mayores parámetros para el ajuste tienden al sobreajuste, presentando diferencias entre los comportamientos de la validación y el entrenamiento y evidenciando una variabilidad acentuada.



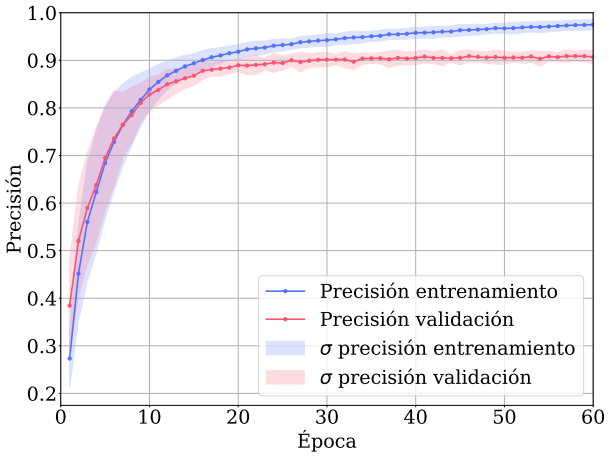
(a) Modelo 2 capas residuales



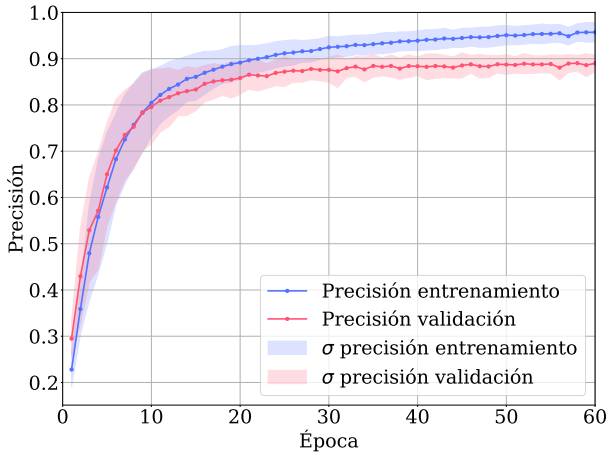
(b) Modelo 4 capas residuales



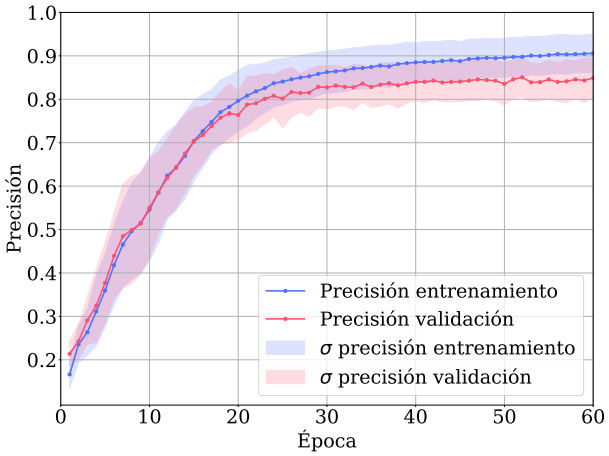
(c) Modelo 6 capas residuales



(d) Modelo 8 capas residuales

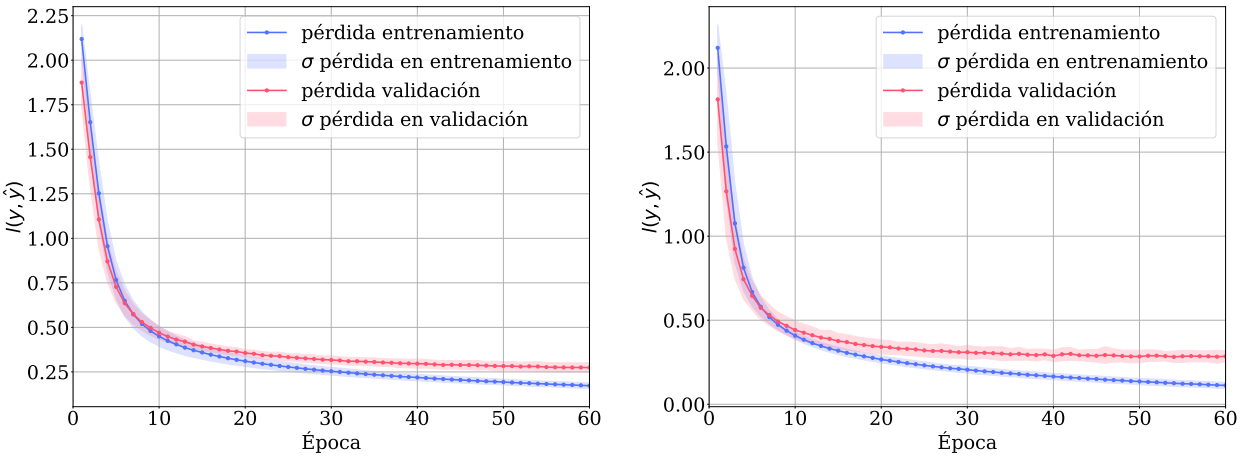


(e) Modelo 10 capas residuales



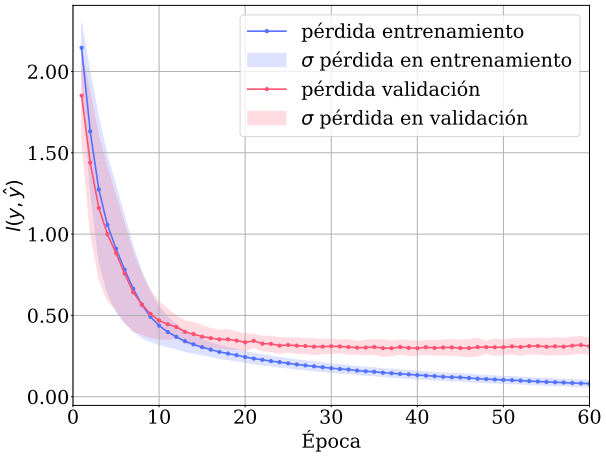
(f) Modelo 12 capas residuales

Figura 2-8.: Curvas de aprendizaje para modelos neuronales con diferente cantidad de capas residuales. Experimento A de la Subsección 2.3.2.

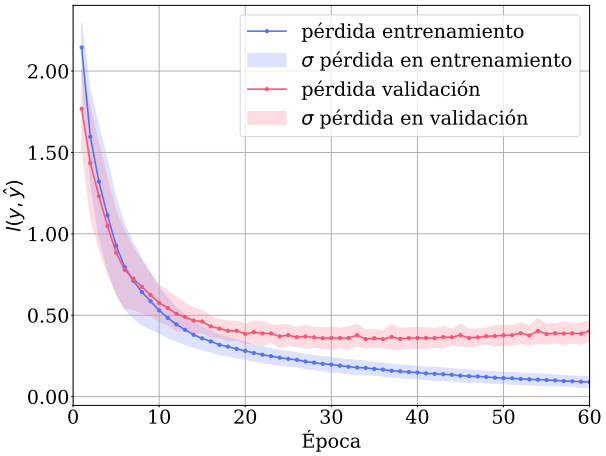


(a) Modelo 2 capas residuales

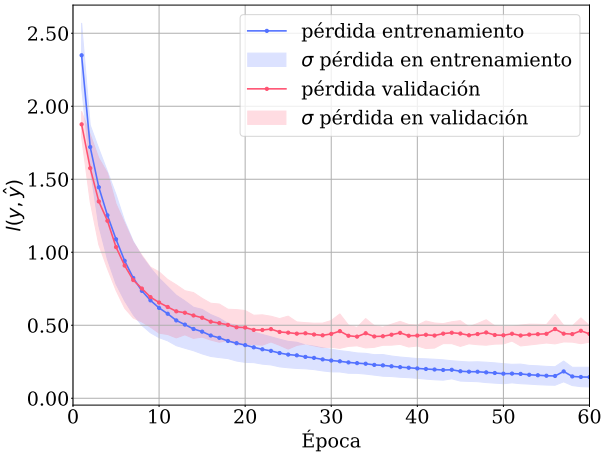
(b) Modelo 4 capas residuales



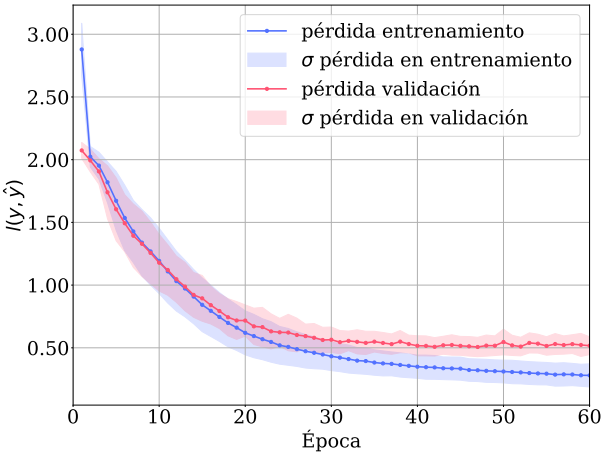
(c) Modelo 6 capas residuales



(d) Modelo 8 capas residuales



(e) Modelo 10 capas residuales



(f) Modelo 12 capas residuales

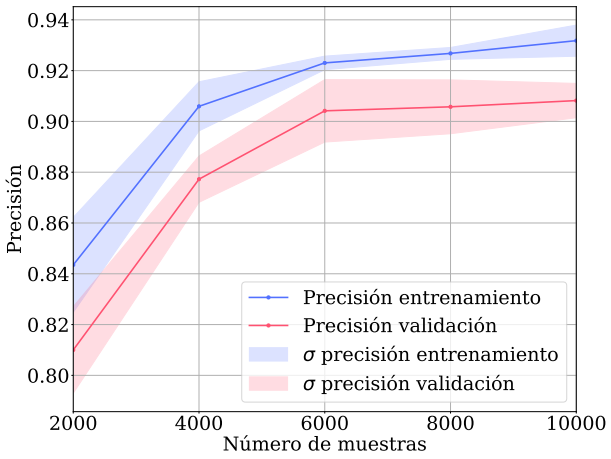
Figura 2-9.: Curvas de la función de pérdida para modelos neuronales con diferente cantidad de capas residuales. Experimento A de la Subsección 2.3.2.

Tabla 2-1.: Resultados de la precisión en el entrenamiento y la prueba de los diferentes modelos de redes neuronales residuales para el Experimento A de la sección Subsección 2.3.2. Fuente: elaborado por el autor. (En adelante todas las tablas que no mencionen fuente se atribuyen a la elaboración del autor).

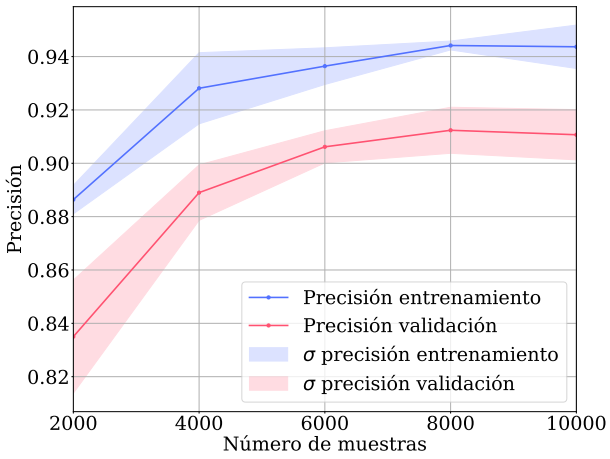
Número de capas residuales	**2**	**4**	**6**	**8**	**10**	**12**
% Promedio precisión en entrenamiento	95.16	96.85	97.91	97.49	95.71	**90.58**
% Promedio precisión en prueba	91.96	92.00	91.56	90.41	89.04	**84.77**
% Diferencia del promedio de la precisión entre entrenamiento y prueba	3.20	4.85	6.35	**7.08**	6.67	5.81
± Desviación estándar de la precisión en entrenamiento	0.486	0.442	0.641	0.970	1.888	**4.088**
± Desviación estándar de la precisión en prueba	0.311	0.292	0.309	0.909	1.183	**3.782**

Al finalizar cada pliegue de la validación cruzada se evaluó el clasificador con el conjunto de *test* de 10,000 imágenes. En la Tabla **2-1** se presenta el resultado del promedio y la desviación estándar de la precisión en la prueba y el entrenamiento. En la Tabla se resaltaron los valores que sugieren un desempeño inferior, señalando los valores mínimos de promedio en precisión y el valor máximo para las diferencias de estos promedios y las desviaciones estándar. Como se puede notar los resultados refuerzan las conclusiones previas, los modelos con más capas fueron los que menor desempeño presentaron con el conjunto de prueba, contrario a sus buenos resultados en el entrenamiento. La diferencia entre el valor logrado en entrenamiento y prueba se acentúa cuando el modelo tiene más capas, igual comportamiento exhibe la desviación estándar, esta medida también se incrementa incluso en el conjunto de entrenamiento.

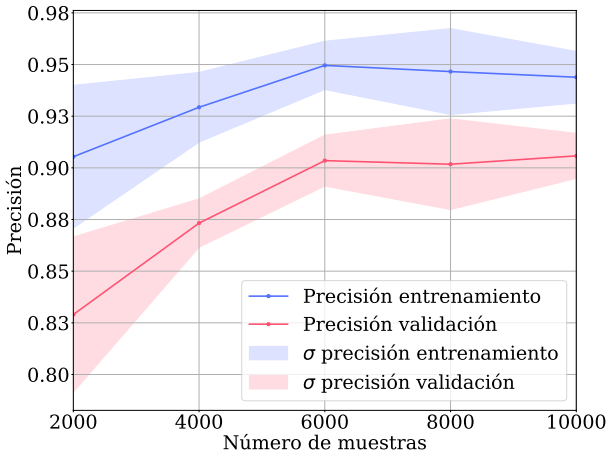
Para los modelos también se realizaron las simulación de su desempeño de acuerdo con el número de datos empleados para el entrenamiento. En esta simulación se optó por entrenar los clasificadores con muestras de 2,000 a 10,000 datos, incrementando en grupos de 2,000. Los resultados logrados para la precisión en entrenamiento y validación se presentan en la Figura **2-10** y la función de pérdida se ilustra en la Figura **2-11**. En las gráficas se puede observar que a medida que se incrementan los datos para el proceso de aprendizaje se mejora el desempeño del clasificador, tanto en el entrenamiento como en la validación. También se puede advertir que la variabilidad del modelo es mayor cuando se tienen pocas muestras, y esto afecta más a los modelos que requieren la estimación de un elevado número de parámetros, por ejemplo en las Figuras **2-10**f y **2-11**f se puede observar una variabilidad alta cuando se tienen pocos datos de entrenamiento, tanto en la precisión como en la pérdida.



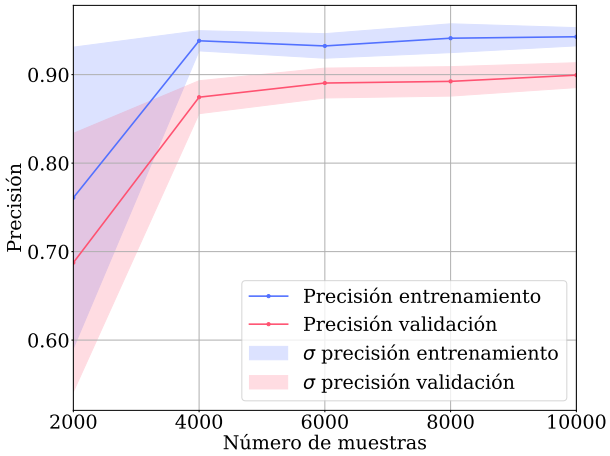
(a) Modelo 2 capas residuales



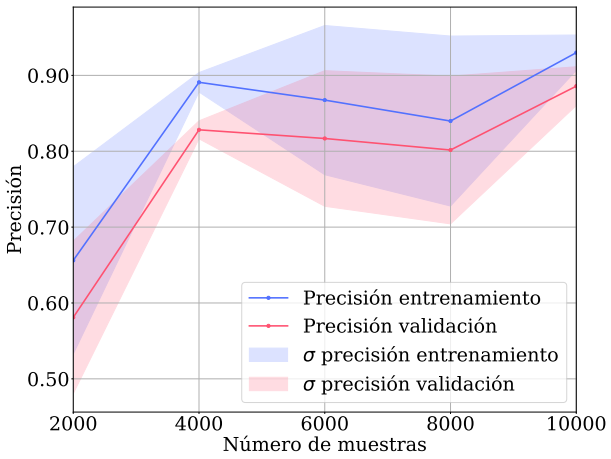
(b) Modelo 4 capas residuales



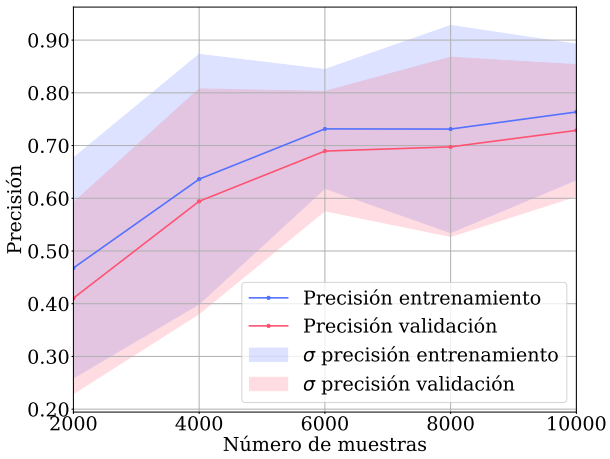
(c) Modelo 6 capas residuales



(d) Modelo 8 capas residuales

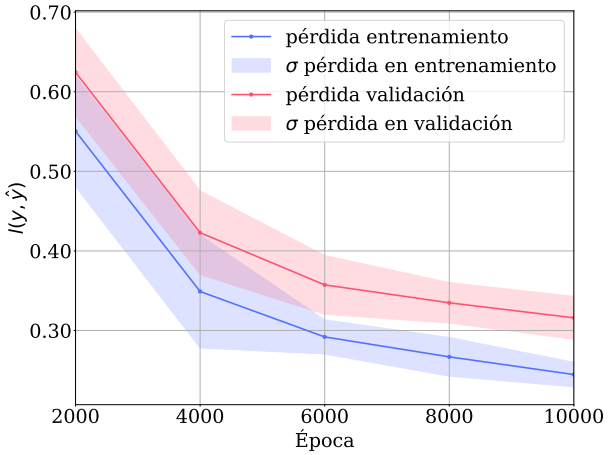


(e) Modelo 10 capas residuales

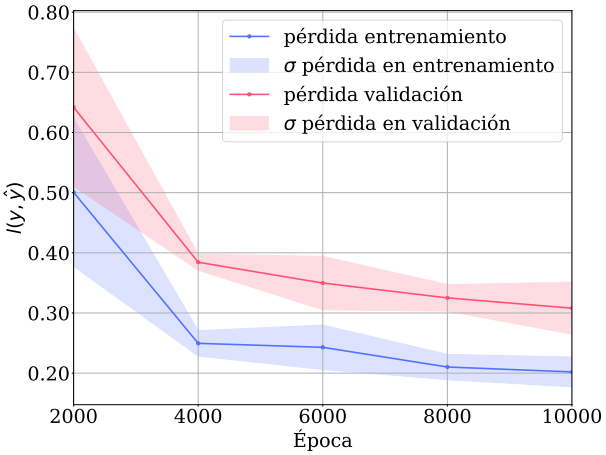


(f) Modelo 12 capas residuales

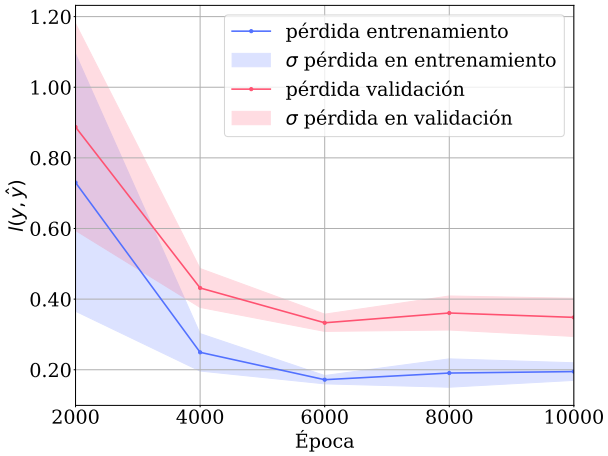
Figura 2-10.: Curvas de aprendizaje para modelos neuronales entrenados con diferentes tamaños de observaciones. Experimento A de la Subsección 2.3.2.



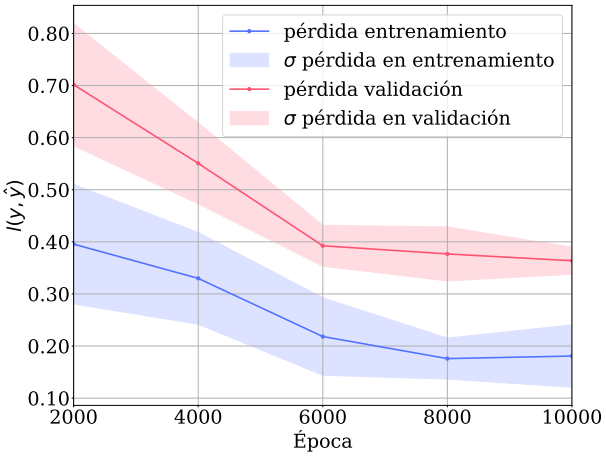
(a) Modelo 2 capas residuales



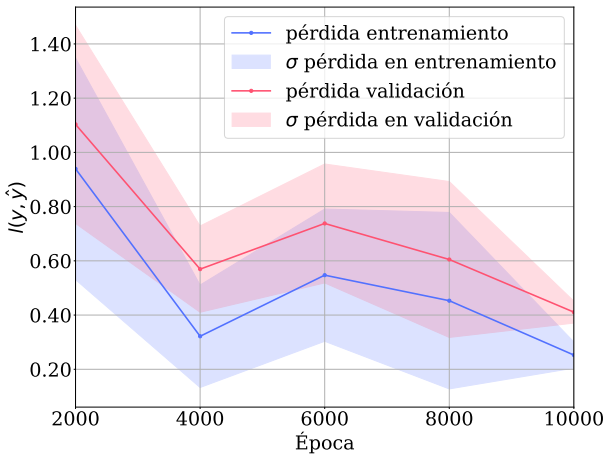
(b) Modelo 4 capas residuales



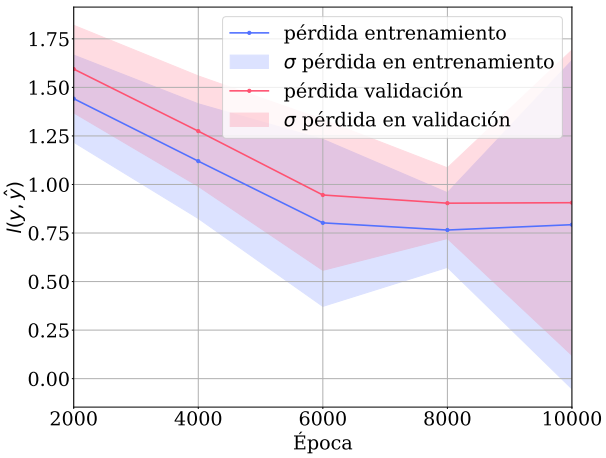
(c) Modelo 6 capas residuales



(d) Modelo 8 capas residuales



(e) Modelo 10 capas residuales



(f) Modelo 12 capas residuales

Figura 2-11.: Curvas de la función de pérdida para modelos neuronales entrenados con diferentes tamaños de observaciones. Experimento A de la Subsección 2.3.2.

Experimento B

En este experimento se consideraron los mismos hiper-parámetros y metodología que se desarrolló en el **Experimento A**, ahora para el conjunto de datos *Fashion* MNIST [113]. El nuevo conjunto comparte características similares con el trabajado en el **Experimento A**, contiene 70,000 imágenes distribuidas en 60,000 para entrenamiento y 10,000 para prueba. En este conjunto de datos las observaciones corresponden a imágenes de prendas de vestir las cuales se encuentran en un solo canal, tiene 28×28 píxeles y están distribuidas en 10 clases, para cada clase la Figura **2-12** presenta una imagen disponible en el conjunto de datos. A diferencia del experimento pasado en los 6 modelos entrenados se varió el número de capas iniciando en 1 y finalizando en 11 (número de capas impares), con incremento de 2. La arquitectura de la red se presenta en la Figura **2-7**. El proceso de aprendizaje de la red se realizó con 10,000 datos del conjunto de entrenamiento.

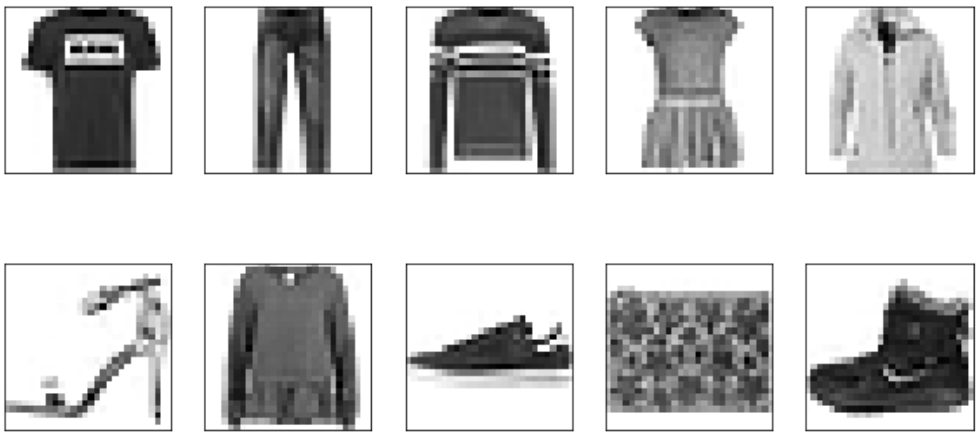
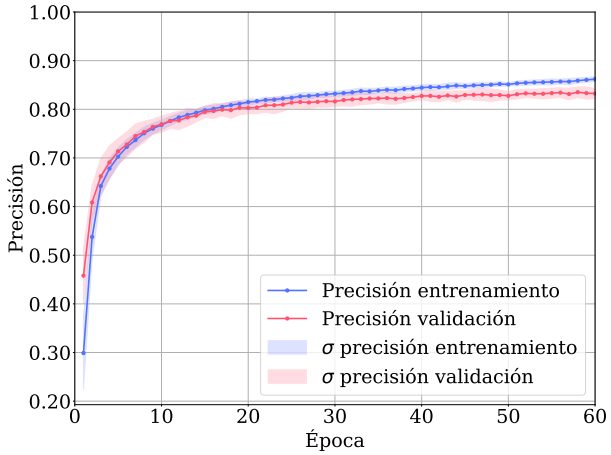
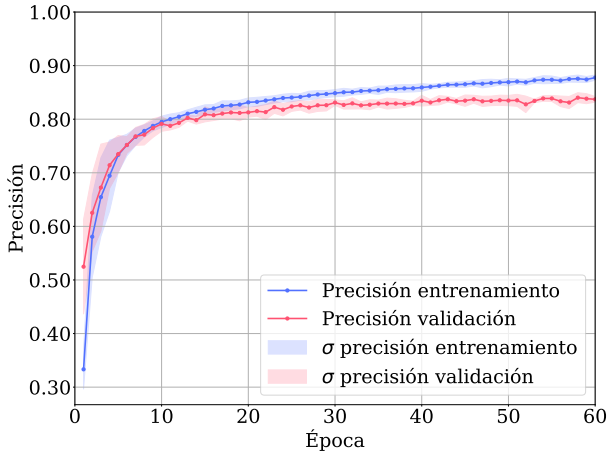


Figura 2-12.: Representación gráfica de algunas imágenes del conjunto de datos *Fashion* MNIST por clase. Fuente: a partir de [113]

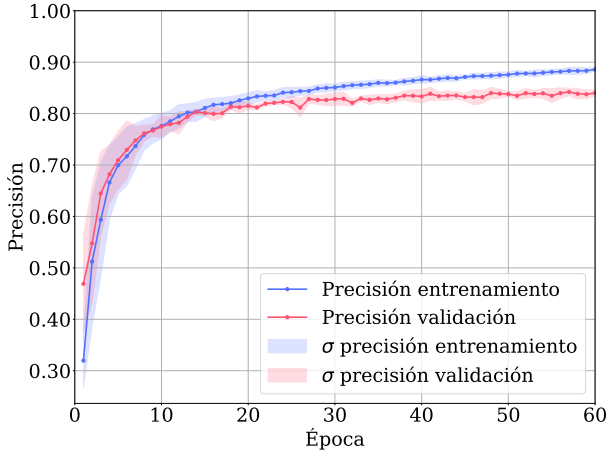
Las curvas de aprendizaje para este experimento se presentan en la Figura **2-13** y la curvas de la función de pérdida en la Figura **2-14**. Los resultados indican un comportamiento similar al obtenido en el **Experimento A**. A medida que incrementan las capas el modelo amplia la diferencia entre los resultados del conjunto de entrenamiento y de validación, este comportamiento se replica en ambas curvas, precisión y pérdida. Otro factor que también se acentúa con el incremento de capas es la variabilidad, lo que indica que modelos de más capas cambian su desempeño dependiendo la muestra del conjunto que se use para su entrenamiento. A diferencia del **Experimento A** se puede notar que en los modelos de 9 y 11 capas no se logra un entrenamiento adecuado, existiendo un subajuste, esta contrariedad puede ser explicada porque el modelo debe estimar un gran número de parámetros con una muestra insuficiente de datos.



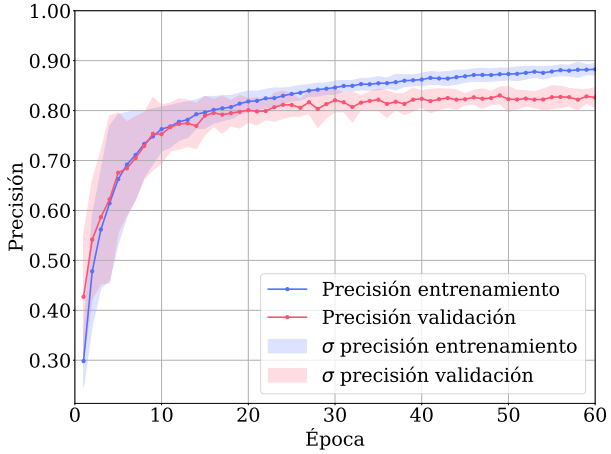
(a) Modelo 1 capas residuales



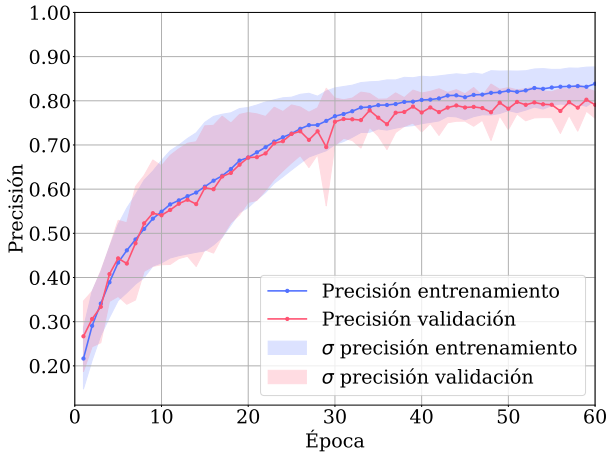
(b) Modelo 3 capas residuales



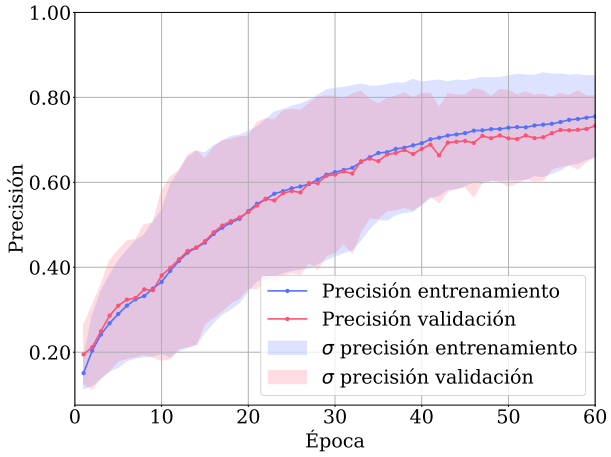
(c) Modelo 5 capas residuales



(d) Modelo 7 capas residuales

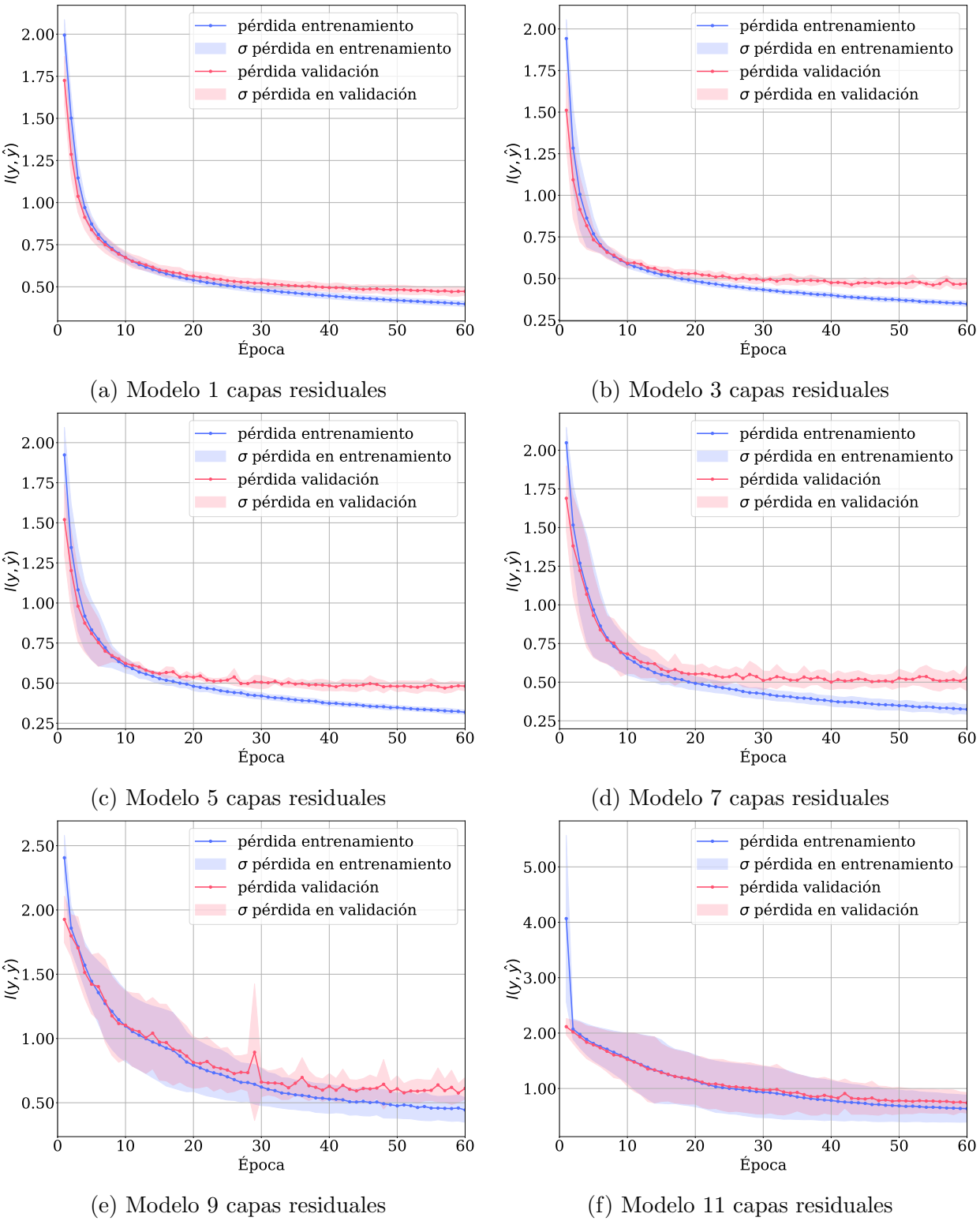


(e) Modelo 9 capas residuales



(f) Modelo 11 capas residuales

Figura 2-13.: Curvas de aprendizaje para modelos neuronales con diferente cantidad de capas residuales. Experimento B de la Subsección 2.3.2.



(a) Modelo 1 capas residuales

(b) Modelo 3 capas residuales

(c) Modelo 5 capas residuales

(d) Modelo 7 capas residuales

(e) Modelo 9 capas residuales

(f) Modelo 11 capas residuales

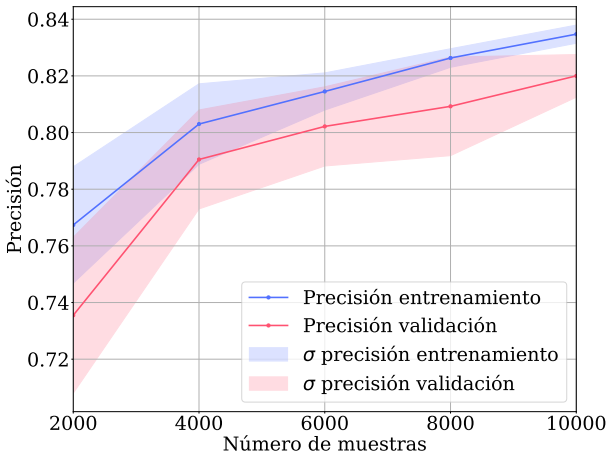
Figura 2-14.: Curvas de la función de pérdida para modelos neuronales con diferente cantidad de capas residuales. Experimento B de la Subsección 2.3.2.

Para este experimento también se estimaron métricas finales, correspondientes al promedio y desviación estándar de la precisión de los 5 pliegues de la validación cruzada en entrenamiento. También se estimó la medida de precisión en el conjunto de prueba, métrica resultado de evaluar el clasificador entrenado en cada pliegue con los 10,000 datos de prueba. Los resultados se presentan en la Tabla **2-2**, similar a la Tabla reportada para el experimento anterior se señalaron los valores que indican el peor desempeño del clasificador, destacando los valores mínimos de precisión y máximos para la diferencia y las desviaciones estándar. Los valores verifican que existe problema de sobreajuste en modelos con mayor complejidad, además se puede caer en escenarios de subajuste a tener muchos parámetros que estimar y una muestra que no sea representativa y lo suficientemente "grande" para realizar el entrenamiento. En la Tabla **2-2** se ha señalado en negrita los valores máximos de cada fila, se puede ver que en los modelos de más capas es donde mayor es la diferencia entre entrenamiento y prueba.

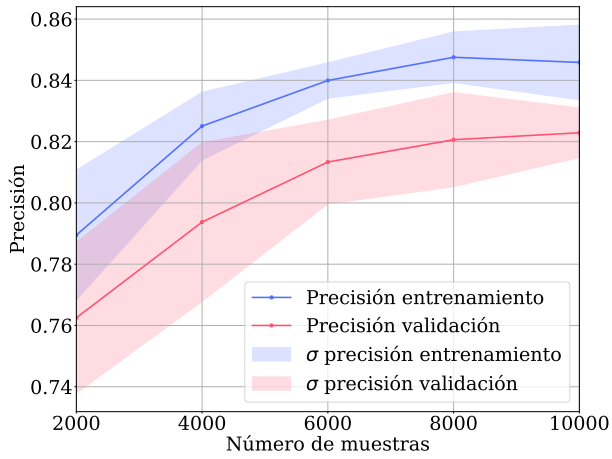
Tabla 2-2.: Resultados de la precisión en el entrenamiento y la prueba de los diferentes modelos de redes neuronales residuales para el Experimento B de la Subsección 2.3.2.

Número de capas residuales	**1**	**3**	**5**	**7**	**9**	**11**
% Promedio precisión en entrenamiento	86.19	87.76	88.58	88.27	83.85	**75.52**
% Promedio precisión en prueba	82.52	82.74	82.81	81.90	77.81	**72.38**
% Diferencia del promedio entrenamiento y prueba	3.17	5.27	**6.40**	6.36	5.49	2.77
± Desviación estándar de la precisión en entrenamiento	0.444	0.437	0.337	1.040	3.415	**8.592**
± Desviación estándar de la precisión en prueba	0.563	0.484	0.642	1.010	2.301	**6.668**

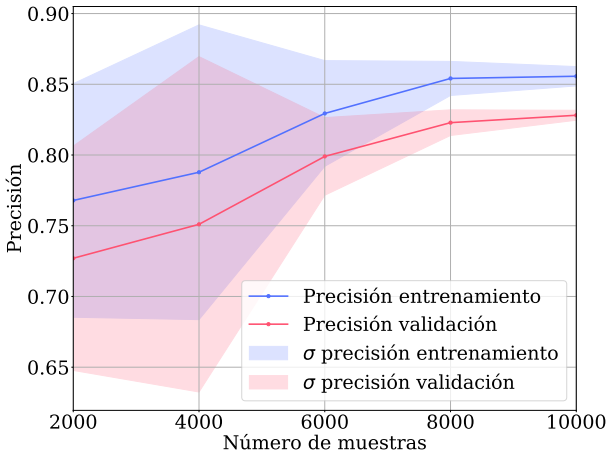
El Experimento B también se desarrolló en un escenario de simulación con segmentos de muestras del conjunto de 10,000 datos, igual que en el experimento anterior se distribuyeron los datos iniciando en 2,000 muestras y realizando incrementos del 20 % hasta 10,000. Los resultados indican que a menor datos el desempeño del clasificador es bastante regular y presenta una alta variabilidad condicionada a la elección de las muestras para el entrenamiento. Las curvas de aprendizaje se presentan en la Figura **2-15** y la gráficas del desempeño de la función de costo se indican en Figura **2-16**.



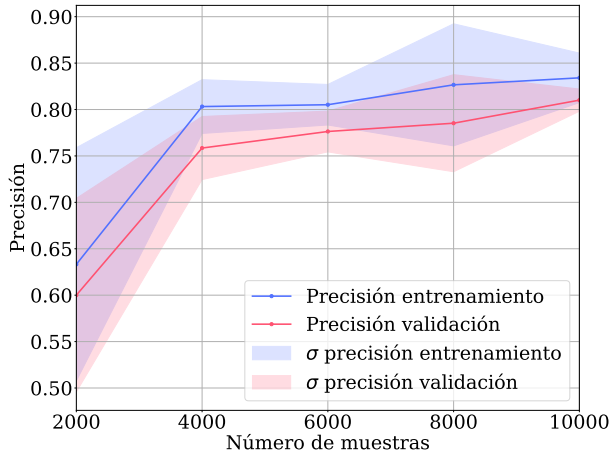
(a) Modelo 1 capa residual



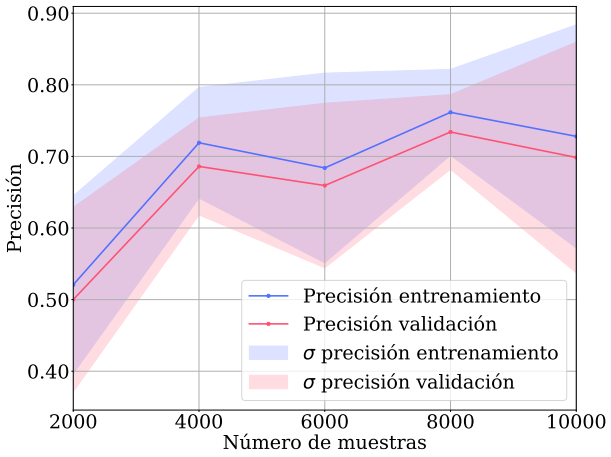
(b) Modelo 3 capas residuales



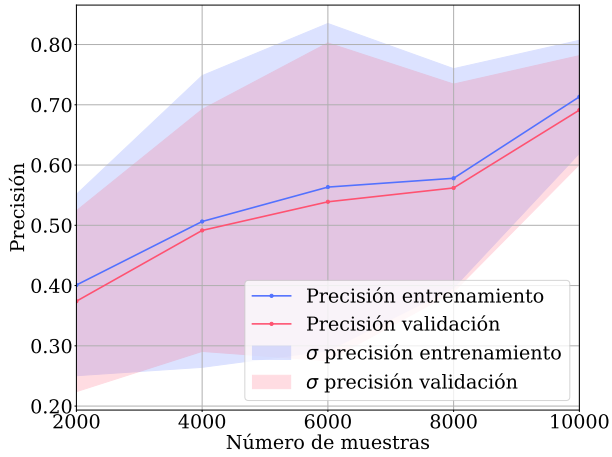
(c) Modelo 5 capas residuales



(d) Modelo 7 capas residuales

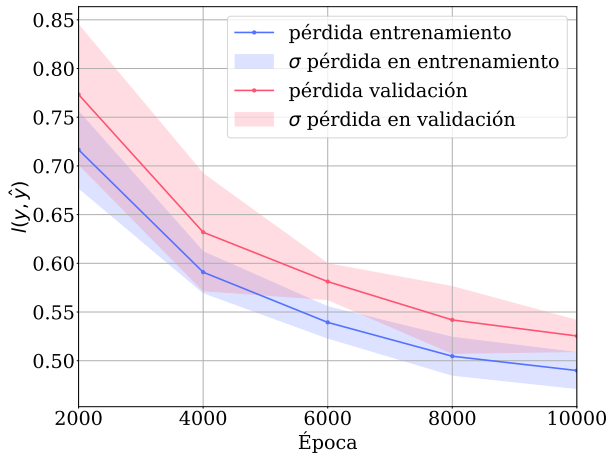


(e) Modelo 9 capas residuales

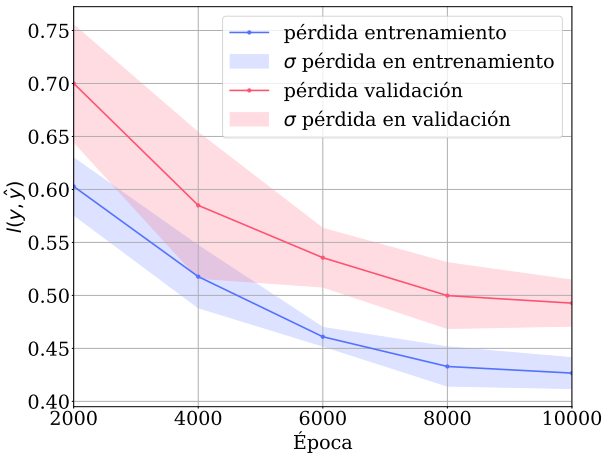


(f) Modelo 11 capas residuales

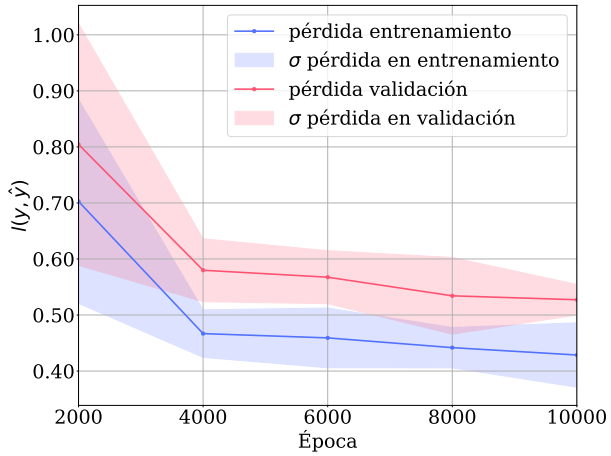
Figura 2-15.: Curvas de aprendizaje para modelos neuronales entrenados con diferentes tamaños de observaciones. Experimento B de la Subsección 2.3.2.



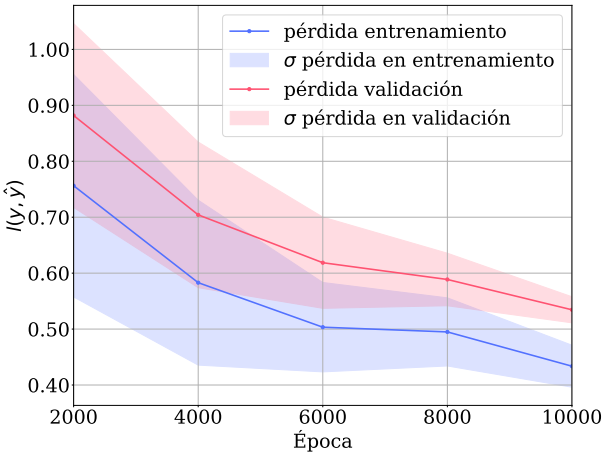
(a) Modelo 1 capa residual



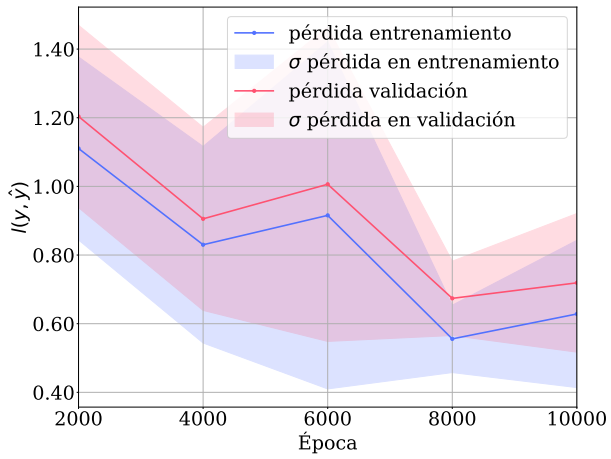
(b) Modelo 3 capas residuales



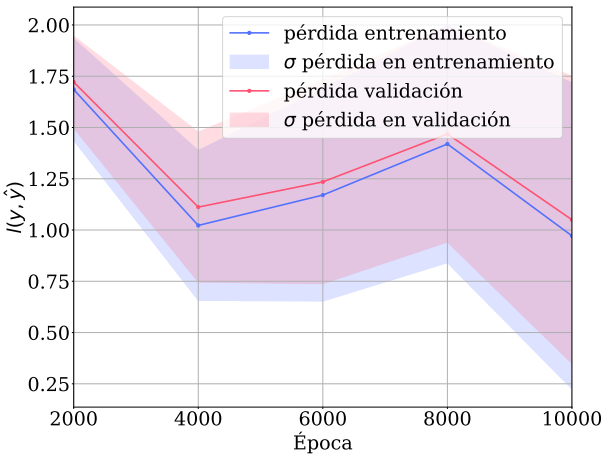
(c) Modelo 5 capas residuales



(d) Modelo 7 capas residuales



(e) Modelo 9 capas residuales



(f) Modelo 11 capas residuales

Figura 2-16.: Curvas de la función de pérdida para modelos neuronales entrenados con diferentes tamaños de observaciones. Experimento B de la Subsección 2.3.2.

Discusión de los experimentos

Con los experimentos realizados en esta sección se pueden resumir los siguientes aspectos relevantes:

- Los resultados indican que en los modelos con más capas se evidencia un mayor sobreajuste del modelo, esto se refleja en que se logran buenos rendimientos en el entrenamiento, pero no se replican los resultados en las curvas de prueba.
- La dependencia del conjunto de datos se refleja en la variabilidad de las curvas de aprendizajes, este fenómeno se ilustra como franjas de desviación estándar. Al incrementar el número de capas en la arquitectura del modelo la variabilidad aumenta, plasmando que los modelos más complejos resultan más dependientes del conjunto de datos que se empleó para entrenar. Esta dependencia no es deseada ya que se considera que los parámetros de la red θ deben ser independientes de la elección de una muestra representativa de datos para el entrenamiento, y deben ser cercanos a los parámetros que permitan generalizar a la población. Por lo tanto, una mayor variabilidad señala estructuras con deficiencia en la generalización y posible sobreajuste.
- En las Figuras **2-10**, **2-15**, **2-11**, **2-16** se puede observar que a menor tamaño de observaciones el desempeño logrado por el clasificador es también menor. Este comportamiento sugiere y verifica el planteamiento de que es necesario contar con un conjunto de datos de tamaño suficiente que garantice una muestra representativa de la población.
- En las arquitecturas con 12 capas (Experimento A) y 11 (Experimento B) se puede evidenciar menor convergencia y mayor variabilidad. Este comportamiento se puede relacionar con el balance inadecuado entre la cantidad de parámetros que se requieren estimar y la cantidad de observaciones empleadas en el proceso. Con esto se verifica la idea que es necesario establecer modelos acordes con la dimensión de las observaciones y el número disponible de estas. Además se puede interpretar que no siempre es adecuado incrementar la complejidad del modelo para resolver un problema, por el contrario es mejor encontrar el modelo más simple que resuelva la aplicación particular.
- Para los experimentos realizados se puede afirmar que es necesario emplear estrategias para enfrentar el sobreajuste, sin estos métodos los modelos presentan diferencias significativas entre el los resultados de aprendizaje y la generalización, además se observa dependencia del conjunto de entrenamiento y dificultad de convergencia. Estas problemáticas se agudizan en modelos de arquitecturas más complejas.

3. Método propuesto

La problemática del sobreajuste, de acuerdo con lo estudiando en la Sección 2.3, responde a diversos factores, razón por la cual los métodos de solución actuales también son variados. En general, se pueden discriminar dos causas globales que inciden en el fenómeno. En primer lugar la estructura del modelo, ya que, la variación de la arquitectura y los hiper-parámetros determinan el comportamiento posterior que solo se conoce a medida que avanza el entrenamiento y al final de este proceso. Por otra parte, la distribución y naturaleza de los datos disponibles para el entrenamiento es también determinante en el resultado final. Desde esta perspectiva se considera, en esta investigación, que una forma general de afrontar la problemática del sobreajuste es enfocar la atención en la estructura neuronal. Esto debido a que la estructura, incluyendo la arquitectura y los hiper-parámetros, impacta en el desempeño del modelo y determina su robustez ante variaciones de datos de entrada. Por lo tanto, se pueden orientar los esfuerzos por construir estructuras adaptativas las cuales sean capaces de lograr un desempeño adecuado tanto en el proceso de aprendizaje como en la fase de generalización.

La estructura del modelo se relaciona con su complejidad y es resultado de la definición de los hiper-parámetros, tanto de la arquitectura como del entrenamiento. La capacidad de generalización se puede vincular con la robustez de la red para tolerar variaciones en la entrada. Típicamente las técnicas de regularización funcionan de forma adecuada restringiendo la complejidad del modelo [12, 100], por su parte, la teoría para afrontar la robustez se ha encaminado al estudio de la propiedad de continuidad de Lipschitz de la red neuronal [25, 24]. Estos dos enfoques se combinaron y adaptaron para establecer un método teórico y computacional para reducir el sobreajuste en estructuras neuronales residuales convencionales. En la propuesta también se consideró la facilidad de implementación y evitar, en la medida de lo posible, un elevado costo computacional.

En este Capítulo se describe la propuesta teórica y los resultados computacionales de la implementación del método. Inicialmente se plantea la formulación matemática del procedimiento y el algoritmo resultante. También se incluyen algunas demostraciones matemáticas que relacionan la propuesta teórica con las perturbaciones en los datos de entrada, el gradiente de la función de costo y la equivalencia entre normas convencionalmente usadas en la regularización. En diferentes arquitecturas de modelos neuronales residuales se verificó el desempeño del método a través de simulaciones para tareas de clasificación. Las validaciones

se realizaron en conjuntos estándar de datos, además se incluyó la metodología de validación con imágenes adversarias.

3.1. Propuesta teórica

El método propuesto, denominado **Regularización Adaptativa con Restricciones Aleatorias** (simplificado Regularización LBA), se fundamenta en dos modificaciones del algoritmo tradicional de regularización. En primer lugar se consideró incluir restricciones aleatorias de carácter individual para las matrices de pesos de las capas de la red. Esto implica que en cada iteración se elige de manera aleatoria una capa de la red y se penaliza su matriz de pesos. La segunda modificación consiste en escoger un parámetro de regularización adaptativo, relacionado con el aporte de la capa aleatoria al sobreajuste del modelo. Estas dos modificaciones y las consideraciones que motivaron su propuesta se detallan en las siguientes secciones.

3.1.1. Regularización con restricciones aleatorias

En una estructura convencional de red neuronal con capas completamente conectadas, si se aplica una regularización tal que se produzca un decaimiento acentuado de los pesos de alguna capa, entonces la propagación hacia adelante se verá afectada. Supongamos que para algún índice de capa h la capa $\mathbf{W}^{(h)}$ decae sus pesos, es decir $||\mathbf{W}^{(h)}||_F \to 0$, donde $||\mathbf{W}^{(h)}||_F$ denota la norma de Frobenius, entonces la propagación hacia adelante se restringirá dado que las componentes $w^h_{i,j} \approx 0$. En este escenario la salida de la capa $\mathbf{W}^{(h)}$, sin importar el resultado de la propagación de los datos de entrada hasta la capa $h-1$, se mantendrá siempre en valores nulos, los cuales, a su vez, se propagaran hacia adelante impidiendo completamente el proceso de aprendizaje. Este fenómeno se puede asimilar como un corte en el proceso del flujo de información, la Figura **3-1** presenta una interpretación de este caso.

El fenómeno descrito se puede evitar en las estructuras residuales, dado que, si $||\mathbf{W}^{(h)}||_F \to 0$ se bloqueará la activación de la capa h, pero por acción del salto de conexión se recibe la salida de la capa $h-1$. Esta característica no afecta el proceso de aprendizaje al mantener la propagación de información, el efecto que causa es la disminución de complejidad del modelo que se puede interpretar como la "eliminación de una capa". Otra forma de evitar el decaimiento de los pesos de una capa es considerar incluir la restricción de manera aleatoria durante el proceso iterativo. En general la regularización se aplica en cada iteración para todas las capas, o algunas de estas, sin embargo es complicado identificar para cuales capas incluir esta técnica. Estas consideraciones motivaron la primera modificación del algorimo de regularización para una estructura residual.

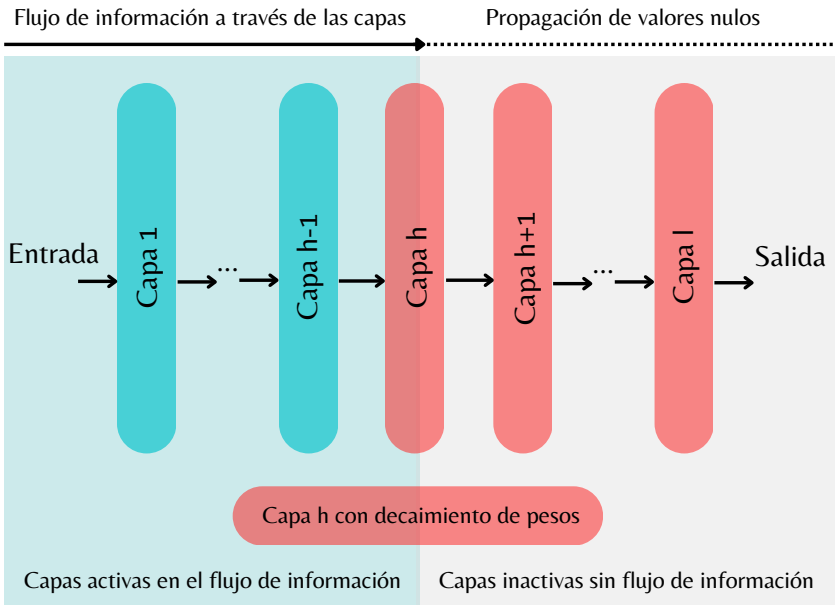


Figura 3-1.: Flujo de información en la red neuronal convencional completamente conectada en presencia de una capa con decaimiento de pesos.

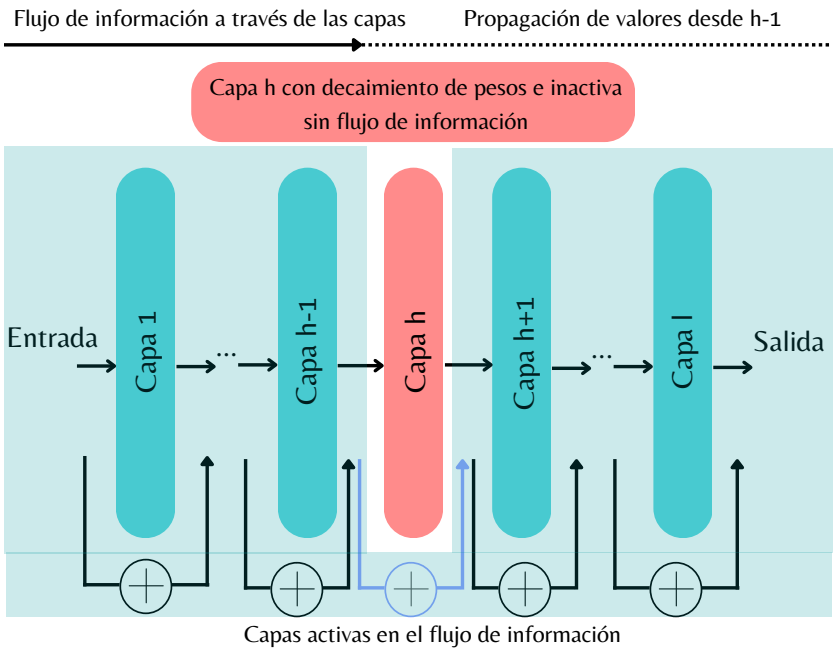


Figura 3-2.: Flujo de información en la red neuronal residual completamente conectada en presencia de una capa con decaimiento de pesos.

Modificación 1. Regularización con restricciones aleatorias

Se propone considerar las regularizaciones individuales por cada capa de forma aleatoria a través de la norma de Frobenius. Es decir, se incluye en la función de costo la penalización para alguna capa elegida de manera aleatoria durante el proceso iterativo. Esta metodología se propone para algoritmos de aprendizaje con optimización por lotes, en particular para el Gradiente Estocástico y en la estructura de red residual (ResNet). La Ecuación 3-1 presenta la función de costo modificada. En la ecuación el súper índice a corresponde a un valor aleatorio generado uniformemente entre los índices de las capas residuales, denotados con $l_s^{(1)}$ el primer índice y $l_s^{(f)}$ para el final, de manera que $l_s^{(1)} \leq a \leq l_s^{(f)}$. En algoritmo 3.1 se presenta el pseudocódigo de esta propuesta.

$$l_r(y, \hat{y}) = l(y, \hat{y}) + \lambda ||\mathbf{W}^{(a)}||_F \tag{3-1}$$

Algoritmo 3.1: Entrenamiento regularización aleatoria

Entrada:
$\mathbf{X}$ conjunto de datos de entrenamiento,
λ factor de regularización,
η tasa de aprendizaje,
N_e número de épocas,
N_b número de lotes,
$f(\mathbf{X}; \theta)$ modelo red neuronal residual
Salida: $\theta = \{\mathbf{W}, \mathbf{b}\}$ parámetros del modelo, pesos y bias
$\theta_0 \leftarrow$ Aleatorio;
para $epoch = 1 : N_e$ **hacer**
  Barajar datos de entrenamiento $\mathbf{X}_s \leftarrow \mathbf{X}$;
  Generar N_b lotes del conjunto de entrenamiento barajado $\mathbf{X}_s$;
  $i \leftarrow 0$;
  para $batch = 1 : N_b$ **hacer**
    Generar índice capa residual aleatoria $a \leftarrow$ Aleatorio;
    Calcular gradiente función de costo
      $\hat{y} = f(\mathbf{X}[batch]; \theta_i)$;
      $\nabla_{l_r} = \nabla_{\theta_i}(l(y, \hat{y}) + \lambda ||\mathbf{W}^{(a)}||_F)$;
    Encontrar $\theta_{i+1} \leftarrow \theta_i - \eta \nabla_{l_r}$;
    $i \leftarrow i + 1$;
  $\theta_0 \leftarrow \theta_{i+1}$;
devolver *[θ_0]*

Para la modificación propuesta se debe tener en cuenta la relación entre el número de lotes y la cantidad de capas del modelo. Un número alto de lotes en relación a la cantidad de capas permitirá, en cada época, que se tengan en cuenta la mayoría de las capas. En el caso contrario, si los lotes son pocos, y hay muchas capas no se puede garantizar que la penalización aleatoria se lleve a cabo para todas las capas.

Por otra parte, se observó que puede existir una relación de la solución de un problema de múltiples restricciones cuando se trabaja con restricciones aleatorias iterativas. Estas ideas se formalizan matemáticamente con las siguiente proposición (por simplicidad únicamente se consideran como variables de decisión los pesos de las capas $\mathbf{W}$).

Proposición 3.1.1

Considere la función de costos a optimizar $l(\mathbf{y}, f(\mathbf{X}, \mathbf{W}))$ donde $\mathbf{W}$ es un arreglo de variables que corresponde a las matrices de pesos de las capas y el bias de la forma $\mathbf{W} = [\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \cdots, \mathbf{W}^{(L)}]$, $\mathbf{X} \in \mathbb{R}^{k \times n}$ es la matriz de todos los individuos y $\mathbf{y}$ es el vector de todas las etiquetas para los K individuos. La función esta sujeta a L restricciones de la forma $0 \leq ||\mathbf{W}^{(j)}||_F \leq U_j$, con $U_j > 0$, independientes entre sí. Entonces, por multiplicadores de *Lagrange* el problema se puede expresar como

$$l(\mathbf{y}, f(\mathbf{X}, \mathbf{W})) + \Lambda g(\mathbf{W}), \tag{3-2}$$

con $\Lambda = [\lambda_1, \cdots, \lambda_l]$ y $g(\mathbf{W}) = [||\mathbf{W}^{(1)}||_F, \cdots, ||\mathbf{W}^{(L)}||_F]^T$. Considere que este problema se puede resolver aproximadamente con un proceso iterativo en algún número de épocas N_e (iteración) con estrategias de gradiente descendente.

Al considerar una estrategia de solución que emplee el método del gradiente estocástico por lotes, entonces para cada época el proceso se fracciona la matriz $\mathbf{X}$ en matrices $\mathbf{X}_j$ con $j \in \mathbb{Z}, 1 < j < L$ y de modo que $\mathbf{X}_j \in \mathbb{R}^{b \times n}$, con b un divisor de k y $b < k$. (Este mismo análisis de replica para $\mathbf{y}$ fraccionándolo en $\mathbf{y}_j$). Cada época se considera como el paso de todos los lotes $(\mathbf{X}_j, \mathbf{y}_j)$. Si en esta misma estrategia se alternan de forma aleatoria las restricciones, con a un valor aleatorio entre los índices de las capas, resolviendo para cada iteración el sub-problema

$$l(\mathbf{y}_j, f(\mathbf{X}_j, \mathbf{W})) + \lambda_a ||\mathbf{W}^{(a)}||_F. \tag{3-3}$$

Además, si la relación entre $N_b = \frac{k}{b}$ (número de lotes) y L (número de capas) es de forma que $N_b \gg L$, entonces, el minimizador que se alcanza al resolver el problema de la Ecuación 3-3 se aproxima al minimizador solución del problema de la Ecuación 3-2.

Considerando la relevancia de la Proposición anterior se sugiere su análisis y demostración para un trabajo futuro. Las ideas que motivaron la modificación del algoritmo, expresadas al inicio de esta sección, se orientan a efectuar estrategias efectivas de reducción del sobreajuste sin impactar negativamente el proceso de aprendizaje. Por otra parte, como se puede observar en la Ecuación 2-20, el valor λ puede ser interpretado como la formulación en multiplicadores de Lagrange de restricciones de desigualdad para Ω. Esto se estudio en la formulación de la proposición, particularmente tomando como $\Omega = ||.||_F$, entonces las restricciones indicaran que $||\mathbf{W}||_F \leq U$. Típicamente se desconoce que valor debe tener U, pero este parámetro condicionará la norma de los pesos a moverse dentro de una bola de radio U.

Para resolver el problema con restricciones (convencional o aleatorias) por multiplicadores de Lagrange, se debe conocer el valor de U verificando que las restricciones de desigualdad se cumplan. Además, los valores de los multiplicadores deben satisfacer ser no negativos y cumplirse que $\lambda_j||\mathbf{W}^{(j)}||_F \leq 0$, $\forall j = 1 : L$, y cuando $\mathbf{W}^{(j)}$ sea el valor óptimo. Dado que no se conoce U y usualmente el problema no se interpreta como un problema de optimización con restricciones, la solución se realiza ajustando el valor de λ como un hiper-parámetro. La sintonización de λ se puede realizar por métodos de validación cruzada, búsqueda en grillas, búsquedas aleatorias, o algoritmos especializados de configuración automática de hiper-parámetros [114, 115]. Sin embargo, en su mayoría las técnicas implican metodologías prueba-error, siendo necesario entrenar varios modelos y evaluar su rendimiento. La dificultad para establecer un valor adecuado de λ motivó la segunda modificación del algoritmo, la regularización a través de la cota de Lipschitz.

3.1.2. Regularización a través de la cota de Lipschitz

Para esta regularización se propone que el valor de λ se relacione con el aporte de la capa a al problema de sobreajuste, el cual se denotará λ_a. Un valor alto de λ_a para la penalización $||\mathbf{W}^{(a)}||_F$, se efectuará cuando la capa exhiba una alta contribución al sobreajuste, por el contrario se tendrá λ_a "pequeño" cuando la capa no aporte significativamente en la generación de una red sobre-ajustada. La idea ahora se traslada a como medir, de acuerdo con las matrices de pesos $\mathbf{W}$, el sobreajuste del modelo. Esto se podría resolver incluyendo medidas de la complejidad y robustez del modelo, una de estas perspectiva es el análisis de la propiedad de **continuidad Lipschitz** [23, 25, 24, 108, 109]. Este enfoque asume la red neuronal como una función, en el sentido matemático, y sostiene que su capacidad de generalización mejora en aquellas arquitecturas que soporten pequeñas variaciones en los datos de entrada, lo que se relaciona con la continuidad de Lipschitz de la función. Para este análisis se iniciará definiendo la propiedad de **continuidad Lipschitz**, la **constante de Lipschitz** y **cota superior de Lipschitz**.

Definición 3.1.1

Considere $x, y \in \mathcal{X}$ y $f : \mathcal{X} \to \mathcal{Y}$, una función que además depende de unos ciertos parámetros $\mathbf{W}$. Con $\mathcal{X}$ y $\mathcal{Y}$ espacios métricos, de manera que denotamos $d_{\mathcal{X}}$ la métrica de $\mathcal{X}$ y $d_{\mathcal{Y}}$ la métrica de $\mathcal{Y}$. Entonces si existe un $K_L > 0 \in \mathbb{R}$ tal que $\forall x, y \in \mathcal{X}$

$$d_{\mathcal{Y}}(f(x), f(y)) \leq K_L d_{\mathcal{X}}(x, y),$$

se dice que la función f es **Lipschitz Continua**.

Considerando la norma inducida por la métrica, se puede establecer que, si existe un $K_L > 0 \in \mathbb{R}$ tal que $\forall x, y \in \mathcal{X}$

$$||f(x) - f(y)||_{\mathcal{Y}} \leq K_L ||x - y||_{\mathcal{X}},$$

entonces se dice que la función f es **Lipschitz Continua**.

Definición 3.1.2

Si una función f es Lipschitz Continua, entonces se denominará **Constante de Lipschitz** a la constante K_L tal que

$$K_L = \sup_{x,y \in \mathcal{X}} \left\{ \frac{||f(x) - f(y)||_{\mathcal{Y}}}{||x - y||_{\mathcal{X}}} \right\},$$

si se tiene que

$$LB \geq K_L,$$

entonces se dice que LB es una cota superior de Lipschitz.

Al considerar toda la red neuronal como una función $f(\mathbf{X}; \mathbf{W})$ se puede plantear si esta estructura cumple la propiedad de la continuidad de Lipschitz. Inicialmente consideremos esta proposición para una red neuronal convencional, representada por la Ecuación 2-3. En este tipo de arquitecturas la función es la composición de funciones de activación σ y transformaciones afines. Como se puede verificar en el Apéndice A, en general, las funciones comúnmente usadas para la activación son Lipschitz continuas. De acuerdo con esta información se establece la siguiente proposición.

Proposición 3.1.2

Sea $f(\mathbf{X};\mathbf{W})$ la función que representa una arquitectura neuronal convencional, donde f corresponde a la composición finita de funciones de activación y transformaciones afines. Si las funciones de activación de las neuronas son Lipschitz continuas, entonces, f también lo es.

Demostración:

- Las transformaciones afines son Lipschitz continuas (Anexo Sección B.2).
- Las funciones de activación son Lipschitz continuas (Anexo A).
- Las composición de funciones Lipschitz continuas también es Lipschitz continua (Anexo Sección B.1).

□

Por otro lado, de acuerdo con lo detallado en el Anexo Sección B.3 y en general en el Anexo B, una cota de Lipschitz para esta estructura será el producto de las normas espectrales de las matrices de pesos, como se presenta en la Ecuación 3-4.

$$LB = \prod_{i=1}^{L} \sqrt{\rho(\mathbf{W}^{(i)T}\mathbf{W}^{(i)})}, \tag{3-4}$$

donde $\rho(\cdot)$ representa el radio espectral, que corresponde al máximo de los valores propios en valor absoluto.

Finalizada la demostración de la continuidad de Lipschitz para una red neuronal convencional también es importante indagar sobre la propiedad en una red de estructura residual. Para esto se considera la siguiente proposición.

Proposición 3.1.3

Sea $f(\mathbf{X};\mathbf{W})$ la función que representa una arquitectura neuronal residual convencional, donde f corresponde a la composición finita de funciones de activación y transformaciones afines con saltos entre conexiones. Si las funciones de activación de las neuronas son Lipschitz continuas, entonces, f también lo es.

Demostración:

Se ha establecido que la composición de funciones Lipschitz conservan la propiedad y que las transformaciones afines lo son. El salto residual corresponde a una suma de dos funciones, el

acumulado previo y la identidad. En el Anexo Sección B.4 se demuestra que la suma de dos funciones Lipschitz es también Lipschitz continua. En consecuencia, una estructura residual convencional también preserva esta propiedad, si las funciones de activación lo son y si el salto de conexión también es una función de este tipo.

□

Hasta este momento se ha demostrado que una estructura neuronal convencional, sin o con saltos residuales, se cumple la propiedad de la continuidad de Lipschitz si las funciones de activación también lo son. A continuación, se realizan algunos análisis que relacionan el comportamiento de la red con esta propiedad.

Relación de la cota de Lipschitz con las perturbaciones en los datos de entrada

Considérese que se tiene una observación $x_0 \in \mathcal{X}$ para la cual se ha establecido una etiqueta $y_0 \in \mathcal{Y}$, además la red representada por la función $f(\mathbf{X}; \mathbf{W})$ puede mapear de forma adecuada para el dato x_0, de modo que $y_0 = f(x_0; \mathbf{W})$. Ahora, se asume que existen errores o ruidos en la estimación del dato x_0, se tiene entonces una bola definida como $B_{\mathcal{X}}(x_0, \epsilon) = \{x \in \mathcal{X} |\ ||x_0 - x||_{\mathcal{X}} \leq \epsilon\}$, para $\epsilon > 0$ un valor arbitrariamente pequeño.

Además considérese que si se toma un $x_n \in B_{\mathcal{X}}(x_0, \epsilon)$ la red generará una salida $y_n = f(x_n; \mathbf{W})$. Los valores y_n y y_0 deberían ser cercanos al ser el resultado de observaciones que provienen de un valor y su perturbación. La "cercanía" de los valores se puede analizar con la norma de sus diferencias $||y_0 - y_n||_{\mathcal{Y}} = ||f(x_0) - f(x_n)||_{\mathcal{Y}}$. La función f representa una red neuronal convencional con funciones de activación Lipschitz continuas, incluyendo o no saltos residuales. Entonces, como ya se demostró, esta función es Lipschitz continua. De acuerdo con lo anterior se satisface para algún $LB > 0$ que

$$||y_0 - y_n||_{\mathcal{Y}} \leq LB||x_0 - x_n||_{\mathcal{X}}.$$

Además, como $x_n \in B(x_0, \epsilon)$, por definición se cumple en particular para x_n que $||x_0 - x_n|| \leq \epsilon$, de modo que

$$||y_0 - y_n||_{\mathcal{Y}} \leq LB\epsilon.$$

Entonces se puede decir que la salida $y_n \in B_{\mathcal{Y}}(y_0, LB\epsilon)$, es decir la distancia entre el valor de la etiqueta y_0 y la salida de la red para entradas arbitrarias x_n en la bola $B_{\mathcal{X}}(x_0, \epsilon)$ estará acotada por LB veces el valor del radio de dispersión ϵ del dato x_0.

Del análisis desarrollado se puede interpretar que si se tienen datos de entrada similares ("cercanos"), que se podría intuir son de la misma clase, la red genera salidas que distan

de manera proporcional a la distancia de los datos ponderada por una cota de Lipschitz. Otra interpretación que se puede establecer es que la perturbación o el ruido en los datos se propaga y se escala a través de la red de manera proporcional al ruido de entrada y a una cota Lipschitz, la Figura **3-3** ilustra el comportamiento de la propagación de este tipo. La perspectiva del análisis de la continuidad de Lipschitz, en general, ha motivado la idea de construir estructuras neuronales que fuercen la continuidad de Lipschitz [23, 24, 108, 109]. En particular conseguir arquitecturas con constantes de Lipschitz igual a la unidad [25] permite relacionar las métricas originales entre los datos de entrada y sus imágenes a través de la red. Es importante resaltar que la restricción sobre la cota o la constante de Lipschitz debe permitir que se continúe optimizando la función de costo, la cual finalmente indica el error de ajuste y determina el aprendizaje del modelo.

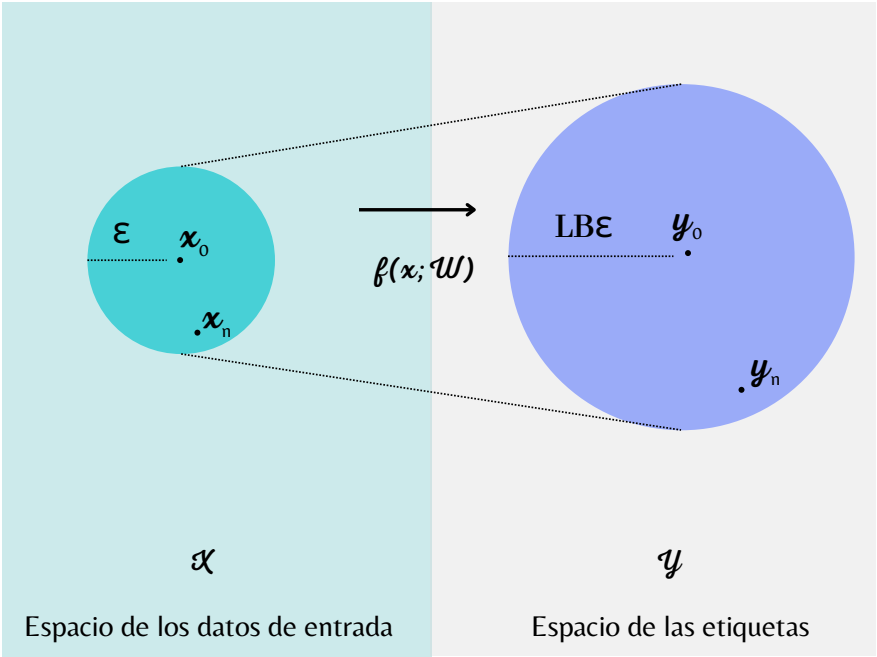


Figura 3-3.: Diagrama de la propagación de la perturbación en los datos de entrada a través de la función que representa la red neuronal.

Relación de la cota de Lipschitz con el gradiente de la función de costo

De acuerdo con lo expuesto en la Sección 2.1 la forma de establecer los parámetros de la red neuronal depende de la magnitud del gradiente de la función de pérdida. En el entrenamiento de las redes neuronales un problema usual es el desvanecimiento del gradiente, la situación contraria es la explosión. En el primer caso el valor del gradiente se desvanece, es decir, toma valor pequeños que se propagan a través de las capas impidiendo la actualización de los pesos y estacando al algoritmo en minimizadores locales. En la explosión del gradiente

sucede el efecto contrario, el gradiente crece a través de las capas provocando un cambio en los parámetros de magnitud agigantada, esto ocasiona problemas en la convergencia del algoritmo de aprendizaje.

No es clara la relación de ambos fenómenos con el problema de sobreajuste, sin embargo, la tendencia a detener el algoritmo en minimizadores locales, consecuencia del desvanecimiento, y la divergencia del algoritmo cuando existe explosión del gradiente, son problemas que afectan el proceso de establecer el valor óptimo de los parámetros de la red. En consecuencia, tanto el desvanecimiento del gradiente como su explosión son escenarios no deseados. A continuación se propone establecer un análisis para determinar una cota sobre la magnitud de las componentes del gradiente de la función de costo, este valor permitirá identificar que elementos inciden en la explosión o desvanecimiento del gradiente.

Análisis 1. Perceptrón

Proposición 3.1.4

Si se cuenta con una estructura de red neuronal de tipo Perceptrón con función de activación sigmoide, entonces la magnitud de la derivada parcial de la función de costo respecto a algún peso w_j esta acotada por la constante de Lipschitz LB.

Para esta demostración se toma como función de costo la entropía cruzada, de manera que el proceso de aprendizaje del Perceptrón se sintetiza en resolver el problema de optimización

$$\min_{\theta} \frac{1}{K} \sum_{i=1}^{K} y_i \ln(\hat{y}_i) + (1 - y_i) \ln(1 - \hat{y}_i).$$

Donde $\hat{y}_i = f(\mathbf{x}_i; \theta)$, es decir, $\hat{y}_i$ corresponde a la salida de la red f que depende de los parámetros θ, y $\mathbf{x}_i$ es el vector i-ésimo de entrada, tal que $\mathbf{x}_i = [x_1, \cdots, x_n]$. Entonces respecto a cada uno de los pesos w_j, se tiene

$$\frac{\partial l}{\partial w_j} = \frac{1}{K} \left[\sum_{i=1}^{K} \frac{\partial}{\partial w_j} \left(y_i \ln(f(\mathbf{x}_i; \theta)) + (1 - y_i) \ln(1 - f(\mathbf{x}_i; \theta)) \right) \right].$$

En la arquitectura de Perceptrón con función de activación sigmoide la función f esta determinada como $f(\mathbf{x}_i; \theta) = \sigma(\mathbf{w}^T \mathbf{x}_i + b)$, en este caso $\mathbf{w}$ es un vector de pesos de la forma $\mathbf{w}^T = [w_1, \cdots, w_n]$. Para efectos de esta prueba no se tendrá en cuenta el valor del bias, de modo que $f(\mathbf{x}_i; \mathbf{w}) = \sigma(\mathbf{w}^T \mathbf{x})$. Entonces se tendría que

$$\frac{\partial l}{\partial w_j} = \frac{1}{K}\left[\sum_{i=1}^{K}\frac{\partial}{\partial w_j}\left(y_i \ln\left(\sigma(\mathbf{w}^T\mathbf{x}_i)\right) + (1-y_i)\ln\left(1-\sigma(\mathbf{w}^T\mathbf{x}_i)\right)\right)\right]$$
$$= \frac{1}{K}\left[\sum_{i=1}^{K}\frac{y_i}{\sigma(\mathbf{w}^T\mathbf{x}_i)}\sigma(\mathbf{w}^T\mathbf{x}_i)(1-\sigma(\mathbf{w}^T\mathbf{x}_i))x_{ji} - \frac{(1-y_i)}{(1-\sigma(\mathbf{w}^T\mathbf{x}_i))}\sigma(\mathbf{w}^T\mathbf{x}_i)(1-\sigma(\mathbf{w}^T\mathbf{x}_i))x_{ji}\right].$$

La notación x_{ji} hace referencia a la componente j-ésima de la observación i-ésima del conjunto de datos $\{\mathbf{x}_i, y_i\}$.

$$\frac{\partial l}{\partial w_j} = \frac{1}{K}\left[\sum_{i=1}^{K}\frac{y_i}{\sigma(\mathbf{w}^T\mathbf{x}_i)}\sigma(\mathbf{w}^T\mathbf{x}_i)(1-\sigma(\mathbf{w}^T\mathbf{x}_i))x_{ji} - \frac{(1-y_i)}{(1-\sigma(\mathbf{w}^T\mathbf{x}_i))}\sigma(\mathbf{w}^T\mathbf{x}_i)x_{ji}(1-\sigma(\mathbf{w}^T\mathbf{x}_i))\right]$$
$$= \frac{1}{K}\left[\sum_{i=1}^{K}(\sigma(\mathbf{w}^T\mathbf{x}_i)(1-\sigma(\mathbf{w}^T\mathbf{x}_i))x_{ji})\left(\frac{y_i}{\sigma(\mathbf{w}^T\mathbf{x}_i)} - \frac{(1-y_i)}{(1-\sigma(\mathbf{w}^T\mathbf{x}_i))}\right)\right]$$
$$= \frac{1}{K}\left[\sum_{i=1}^{K}(\sigma(\mathbf{w}^T\mathbf{x}_i)(1-\sigma(\mathbf{w}^T\mathbf{x}_i))x_{ji})\left(\frac{y_i(1-\sigma(\mathbf{w}^T\mathbf{x}_i)) - (1-y_i)(\sigma(\mathbf{w}^T\mathbf{x}_i))}{\sigma(\mathbf{w}^T\mathbf{x}_i)(1-\sigma(\mathbf{w}^T\mathbf{x}_i))}\right)\right]$$
$$= \frac{1}{K}\left[\sum_{i=1}^{K}x_{ji}(y_i(1-\sigma(\mathbf{w}^T\mathbf{x}_i)) - \sigma(\mathbf{w}^T\mathbf{x}_i) + y_i\sigma(\mathbf{w}^T\mathbf{x}_i)\right]$$
$$= \frac{1}{K}\left[\sum_{i=1}^{K}x_{ji}(y_i - y_i\sigma(\mathbf{w}^T\mathbf{x}_i) - \sigma(\mathbf{w}^T\mathbf{x}_i) + y_i\sigma(\mathbf{w}^T\mathbf{x}_i)\right]$$
$$= \frac{1}{K}\left[\sum_{i=1}^{K}x_{ji}(y_i - \sigma(\mathbf{w}^T\mathbf{x}_i))\right]$$
$$= \frac{1}{K}\left[\sum_{i=1}^{K}x_{ji}(e_i))\right].$$

El coeficiente e_i representa la diferencia entre el valor estimado por la red para la observación $\mathbf{x}_i$ y la etiqueta verdadera y_i. Calculando la norma del gradiente de $\frac{\partial l}{\partial w_j}$ se tiene que:

$$\left|\frac{\partial l}{\partial w_j}\right| = \left|\frac{1}{K}\sum_{i=1}^{K}x_{ji}e_i\right|$$
$$= \left|\frac{1}{K}\right|\left|\sum_{i=1}^{K}x_{ji}e_i\right|$$
$$\leq \left|\frac{1}{K}\right|\left(\sum_{i=1}^{K}x_{ji}^2\right)^{\frac{1}{2}}\left(\sum_{i=1}^{K}e_i^2\right)^{\frac{1}{2}}$$

Asumamos que existe un valor $|e_{\max}|$ tal que $|e_{\max}| \geq |e_i|$, $\forall i = 1 : K$. Entonces se puede ver que:

$$\left|\frac{\partial l}{\partial w_j}\right| \leq \left|\frac{1}{K}\right| \left(\sum_{i=1}^{K} x_{ji}^2\right)^{\frac{1}{2}} \left(\sum_{i=1}^{K} e_{\max}^2\right)^{\frac{1}{2}}$$

Previamente se establecieron dos aspectos importantes: (i) Las arquitecturas neuronales convencionales con funciones de activación continuas en el sentido Lipschitz son también Lipschitz continuas, (ii) la diferencia entre dos valores de la imagen de una función Lipschitz esta acotada por el producto de la cota de Lipschitz y el error en los datos de entrada $||y_0 - y_n||_{\mathcal{Y}} \leq K\epsilon$. Entonces, dado que el Perceptrón es una función continua en el sentido de Lipschitz se tiene que el error en la salida e_i esta acotada por $LB\epsilon_i$, con ϵ_i la perturbación al dato $\mathbf{x}_i$. Lo anterior se cumple en particular para $e_{\max}$, que se relacionaría con algún $\epsilon_{\max}$ de alguna observación, entonces:

$$\begin{aligned}
\left|\frac{\partial l}{\partial w_j}\right| &\leq \left|\frac{1}{K}\right| \left(\sum_{i=1}^{K} x_{ji}^2\right)^{\frac{1}{2}} \left(\sum_{i=1}^{K} |e_{\max}|^2\right)^{\frac{1}{2}} \\
&\leq \left|\frac{1}{K}\right| \left(\sum_{i=1}^{K} x_{ji}^2\right)^{\frac{1}{2}} K^{\frac{1}{2}} |e_{\max}| \\
&\leq \left|\frac{1}{K}\right| \left(\sum_{i=1}^{K} x_{ji}^2\right)^{\frac{1}{2}} K^{\frac{1}{2}} LB\epsilon_{\max}
\end{aligned}$$

Considérese también que existe una componente j en las K observaciones que satisface que $|x_{j\max}| \geq |x_{ji}|$, $\forall i = 1 : K$, por lo tanto:

$$\begin{aligned}
\left|\frac{\partial l}{\partial w_j}\right| &\leq \left|\frac{1}{K}\right| \left(\sum_{i=1}^{K} |x_{ji}|^2\right)^{\frac{1}{2}} K^{\frac{1}{2}} LB\epsilon_{\max} \\
&\leq \left|\frac{1}{K}\right| K^{\frac{1}{2}} |x_{j\max}| K^{\frac{1}{2}} LB\epsilon_{\max} \\
&\leq |x_{j\max}| \epsilon_{\max} LB.
\end{aligned}$$

Al considerar una normalización sobre los datos de entrada entonces:

$$\left|\frac{\partial l}{\partial w_j}\right| \leq \epsilon_{\max} LB.$$

Lo anterior implica que la magnitud de las componentes del gradiente de la función de costo para algún peso w_j esta acotada por la cota de Lipschitz, y factores asociados a la perturbación de los datos. La norma del gradiente esta asociada directamente con el proceso de aprendizaje y permite, junto con la tasa η, establecer el paso de actualización de los pesos. La dirección del gradiente es fundamental, sin embargo una magnitud muy "grande" afectaría la convergencia del método y podría llegar a generar saltos en el espacio de búsqueda de los parámetros óptimos, incluso se puede relacionar con el fenómeno de explosión del gradiente. Por el contrario, al tener escenarios para los cuales se generen normas del gradiente "pequeñas" no se efectúan cambios significativos de los parámetros, esto afecta la convergencia y dificulta la capacidad de evadir de mínimos locales.

Análisis 2. Estructura residual convencional de una capa

Proposición 3.1.5

Si se cuenta con una estructura de red neuronal residual con una capa oculta como se presenta en la Figura **3-4** y con función de activación sigmoide, entonces la magnitud de la derivada parcial de la función de costo respecto al peso $w_{1j}^{(2)}$ esta acotada por la constante LB. La notación del súper índice entre paréntesis representa la capa en la cual está presente el peso.

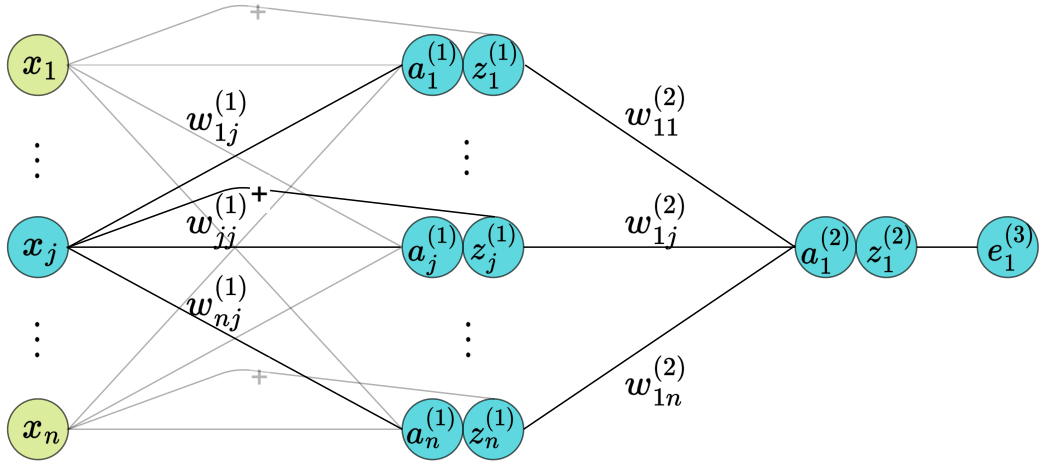


Figura 3-4.: Diagrama de la arquitectura de red neuronal residual de una capa oculta.

En el modelo presentado en la Figura **3-4** la notación a señala la pre-activación de la neurona y z las activaciones, el súper índice hace referencia a la capa. De forma general se pueden establecer las siguiente ecuaciones:

$$a_j^{(1)} = x_1 w_{j1}^{(1)} + \cdots + x_j w_{jj}^{(1)} + \cdots x_n w_{jn}^{(1)}, \tag{3-5}$$
$$z_j^{(1)} = \sigma(a_j^{(1)}) + x_j, \tag{3-6}$$
$$a_1^{(2)} = z_1^{(1)} w_{11}^{(2)} + \cdots + z_j^{(1)} w_{1j}^{(2)} + \cdots z_n^{(1)} w_{1n}^{(2)}, \tag{3-7}$$
$$z_1^{(2)} = \sigma(a_1^{(2)}). \tag{3-8}$$

Las Ecuaciones 3-5 y 3-7 señalan las pre-activaciones de las neuronas en la primera y segunda capa, respectivamente. Por su parte, las Ecuaciones 3-6 y 3-8 indican la activación en la primera y segunda capa. El factor $e_1^{(3)}$ se considera el error cometido en la salida de la red, el súper índice (3) denota que ocurre posterior a la activación de la capa (2).

Calculando la derivada de la función de costo respecto al peso $w_{1j}^{(2)}$, se tiene

$$\frac{\partial l}{\partial w_{1j}^{(2)}} = \frac{\partial e_1^{(3)}}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial w_{1j}^{(2)}}.$$

Las derivadas parciales tienen la siguiente forma,

$$\begin{aligned}\frac{\partial e_1^{(3)}}{\partial z_1^{(2)}} &= \frac{1}{K} \sum_{i=1}^{K} \left(\frac{y_i}{z_1^{(2)}} - \left(\frac{1 - y_i}{1 - z_1^{(2)}} \right) \right) \\ &= \frac{1}{K} \sum_{i=1}^{K} \left(\frac{y_i}{\sigma(a_1^{(2)})} - \left(\frac{1 - y_i}{1 - \sigma(a_1^{(2)})} \right) \right),\end{aligned}$$

$$\frac{\partial z_1^{(2)}}{\partial a_1^{(2)}} = \sigma\left(a_1^{(2)}\right)\left(1 - \sigma\left(a_1^{(2)}\right)\right),$$

$$\frac{\partial a_1^{(2)}}{\partial w_{1j}^{(2)}} = z_j^{(1)}.$$

De manera que se tiene:

$$\begin{aligned}
\frac{\partial l}{\partial w_{1j}^{(2)}} &= \frac{\partial e_1^{(3)}}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial w_{1j}^{(2)}} \\
&= \frac{1}{K} \sum_{i=1}^{K} \left(\frac{y_i}{\sigma(a_1^{(2)})} - \left(\frac{1-y_i}{1-\sigma(a_1^{(2)})} \right) \right) \sigma(a_1^{(2)})(1-\sigma(a_1^{(2)})) z_j^{(1)} \\
&= \frac{1}{K} z_j^{(1)} \sum_{i=1}^{K} (y_i - y_i\sigma(a_1^{(2)}) - \sigma(a_1^{(2)}) + y_i\sigma(a_1^{(2)})) \\
&= \frac{1}{K} z_j^{(1)} \sum_{i=1}^{K} (y_i - \sigma(a_1^{(2)})) \\
&= \frac{1}{K} z_j^{(1)} \sum_{i=1}^{K} e_i
\end{aligned}$$

De forma análoga al análisis anterior se asume que existe un valor $|e_{\max}|$ tal que $|e_{\max}| \geq |e_i|$, $\forall i = 1 : K$. Entonces,

$$\begin{aligned}
\frac{\partial l}{\partial w_{1j}^{(2)}} &= \frac{1}{K} z_j^{(1)} \sum_{i=1}^{K} e_i \\
&\leq \frac{1}{K} z_j^{(1)} \sum_{i=1}^{K} |e_i| \\
&\leq z_j^{(1)} |e_{\max}|.
\end{aligned}$$

Ahora, tomando la norma de $\dfrac{\partial l}{\partial w_{1j}^{(2)}}$ y asumiendo el mismo análisis planteado para la estructura del perceptron se tiene:

$$\begin{aligned}
\left| \frac{\partial l}{\partial w_{1j}^{(2)}} \right| &\leq |z_j^{(1)}||e_{\max}| \\
&\leq |z_j^{(1)}| \epsilon_{\max} LB.
\end{aligned}$$

Proposición 3.1.6

Si se cuenta con una estructura de red neuronal residual con una capa oculta como se presenta en la Figura **3-4** y con función de activación sigmoide, la magnitud de la derivada parcial de la función de costo respecto al peso $w_{jj}^{(1)}$ esta acotada por la constante LB. La notación del súper índice denota la capa en la cual esta presente el peso, en este caso es la capa interna.

La derivada parcial de la función de costo respecto al peso $w_{jj}^{(1)}$ corresponde a

$$\begin{aligned}\frac{\partial l}{\partial w_{jj}^{(1)}} &= \frac{\partial e_1^{(3)}}{\partial z_1^{(2)}}\frac{\partial z_1^{(2)}}{\partial a_1^{(2)}}\frac{\partial a_1^{(2)}}{\partial z_j^{(1)}}\frac{\partial z_j^{(1)}}{\partial a_j^{(1)}}\frac{\partial a_j^{(1)}}{\partial w_{jj}^{(1)}} \\ &= \frac{1}{K}w_{1j}^{(2)}\sigma(a_j^{(1)})(1-\sigma(a_j^{(1)}))\sum_{i=1}^{K}e_i x_{ij}\end{aligned}$$

Tomando la magnitud de $\dfrac{\partial l}{\partial w_{jj}^{(1)}}$, se tiene:

$$\begin{aligned}\left|\frac{\partial l}{\partial w_{jj}^{(1)}}\right| &= \left|\frac{1}{K}w_{1j}^{(2)}\sigma(a_j^1)(1-\sigma(a_j^1))\sum_{i=1}^{K}e_i x_{ij}\right| \\ &= \left|\frac{1}{K}\right||w_{1j}^{(2)}||\sigma'(a_j^{(1)})|\left|\sum_{i=1}^{K}e_i x_{ij}\right| \\ &< \left|\frac{1}{K}\right||w_{1j}^{(2)}||\sigma'(a_j^{(1)})|\left(\sum_{i=1}^{K}e_i^2\right)^{\frac{1}{2}}\left(\sum_{i=1}^{K}x_{ij}^2\right)^{\frac{1}{2}}.\end{aligned}$$

Dado que $|\sigma'(a_j^{(1)})| \leq 1/4$ entonces $|\sigma'(a_j^{(1)})| < 1$, y

$$\left|\frac{\partial l}{\partial w_{jj}^{(1)}}\right| < \left|\frac{1}{K}\right||w_{1j}^{(2)}|\left(\sum_{i=1}^{K}e_i^2\right)^{\frac{1}{2}}\left(\sum_{i=1}^{K}x_{ij}^2\right)^{\frac{1}{2}}.$$

Realizando el mismo análisis que se llevo a cabo para el Perceptrón se tiene:

$$\left|\frac{\partial l}{\partial w_{jj}^{(1)}}\right| \leq |w_{1j}^{(2)}||x_{j\max}|\epsilon_{\max}LB.$$

Sin pérdida de generalidad, considerando que los datos están normalizados, se tiene que:

$$\left|\frac{\partial l}{\partial w_{jj}^{(1)}}\right| \leq |w_{1j}^{(2)}|\epsilon_{\max}LB.$$

Este análisis permite considerar que la magnitud de la función de costo para algún peso de la capa interna de la arquitectura propuesta queda acota por la cota de Lipschitz y factores de los pesos de la siguiente capa. Esta arquitectura y el Perceptrón puede generalizarse en redes de mayor profundidad, de manera que, en general se puede considerar que la magnitud de todas las derivadas parciales de los pesos frente a la función de costo se acotan por LB.

Relación de la cota de Lipschitz con normas equivalentes

En los procesos de regularización generalmente se usan expresiones relacionadas con normas $||\cdot||$, por ejemplo al considerar la regularización L2, sobre las matrices de pesos por capa, esta regularización es el cuadrado de la norma de Frobenius. Dado que en $\mathbb{R}^{n\times n}$ todas las normas son equivalentes (ver Anexo Sección C.3), entonces, las normas matriciales de $\mathcal{M}_{n,n}(\mathbb{R})$ también lo son. Esto es consecuencia de que el espacio $\mathbb{R}^{n\times n}$ es isomorfo a $\mathcal{M}_{n,n}(\mathbb{R})$. En particular se puede ver que la norma de Frobenius $||\cdot||_F$, definida como se presenta en la Ecuación 3-9 es una cota superior de la norma espectral $||\cdot||_2$.

$$||\mathbf{W}||_F = \left(\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}^2\right)^{\frac{1}{2}} \tag{3-9}$$

Proposición 3.1.7

Si se tiene una matriz de entradas reales $\mathbf{W}$ y tamaño $n \times n$, es decir, $\mathbf{W} \in \mathcal{M}_{n,n}(\mathbb{R})$, entonces su norma de Frobenius, defina en la Ecuación 3-9, es una cota superior de su norma espectral.

$$\begin{aligned}
||\mathbf{W}||_F &= \left(\sum_{i=1}^{n}\sum_{j}^{n} w_{ij}^2\right)^{\frac{1}{2}} \\
&=< \mathbf{W}, \mathbf{W} >^{\frac{1}{2}} \\
&= \left[Tr(\mathbf{W}^T\mathbf{W})\right]^{\frac{1}{2}} \\
&= \left[\sum_{i=1}^{n}(\lambda_i(\mathbf{W}^T\mathbf{W}))\right]^{\frac{1}{2}} \\
&\geq \left[\max_{1\leq\ i\ \leq n} \lambda_i(\mathbf{W}^T\mathbf{W})\right]^{\frac{1}{2}} \\
&= ||\mathbf{W}||_2
\end{aligned}$$

Proposición 3.1.8

Si se tiene una matriz de entradas reales $\mathbf{W}$ y tamaño $n \times n$, es decir, $\mathbf{W} \in \mathcal{M}_{n,n}(\mathbb{R})$, entonces la norma de Frobenius, definida en la Ecuación 3-9, es una cota inferior del producto $\sqrt{n}||\mathbf{W}||_2$.

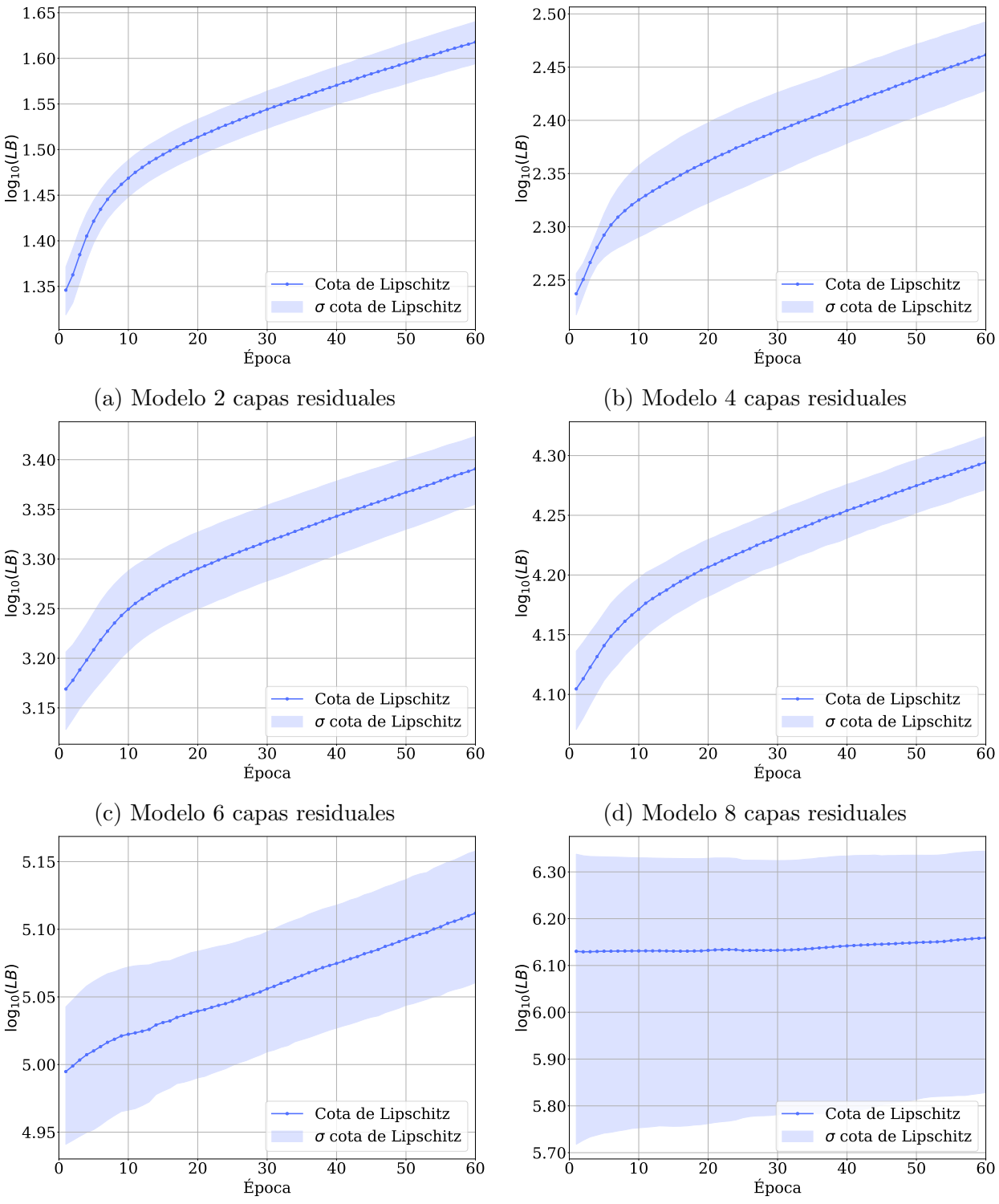
$$
\begin{aligned}
||\mathbf{W}||_F &= \left[\sum_{i=1}^{n}(\lambda_i(\mathbf{W}^T\mathbf{W}))\right]^{\frac{1}{2}} \\
&\leq \left[\sum_{i=1}^{n}(\lambda_{\max}(\mathbf{W}^T\mathbf{W}))\right]^{\frac{1}{2}} \\
&= \sqrt{n}[(\lambda_{\max}(\mathbf{W}^T\mathbf{W}))]^{\frac{1}{2}} \\
&= \sqrt{n}||\mathbf{W}||_2
\end{aligned}
$$

Entonces, se tiene que

$$||\mathbf{W}||_2 \leq ||\mathbf{W}||_F \leq \sqrt{n}||\mathbf{W}||_2. \tag{3-10}$$

Como se ha establecido la norma espectral $||\mathbf{W}||_2$ de una matriz de pesos, en una arquitectura neuronal, corresponde a la constante de Lipschitz particular de una capa. Al aplicar técnicas de regularización L2 se puede observar por la Desigualdad 3-10 que se restringe la cota de la $||\mathbf{W}||_2$ y si la regularización disminuye significativamente el valor de $||\mathbf{W}||_F$ se reducirá la cota de Lipschitz también. Este hecho puede explicar la tendencia a la reducción de la cota de Lipschitz en los experimentos computacionales de este Capítulo.

Los análisis teóricos realizados en este Capítulo permiten concluir que la propiedad de la continuidad de Lipschitz en redes neuronales, vistas como funciones, se relaciona con la capacidad de tolerancia a las variaciones en las entrada. Además, la norma del gradiente y la variación de los pesos esta acotado por la cota de Lispchitz, y en un sentido más estricto, por la constante. Estos dos elementos señalan como buen candidato de la medida del sobreajuste y robustez a la constante o una cota de Lipschitz. A continuación se presentan algunas simulaciones computacionales en las que se estimaron las cotas de Lipschitz para los modelos neuronales en los escenarios de los Experimentos A y B realizados en el capítulo anterior. En ambos experimentos se forzaron estructuras sobre-ajustadas, y se concluyó que características como la insuficiencia de datos para el entrenamiento y la complejidad del modelo, determinada en estos experimentos por el número de capas, inciden en escenarios con mayor tendencia al sobreajuste.



(a) Modelo 2 capas residuales

(b) Modelo 4 capas residuales

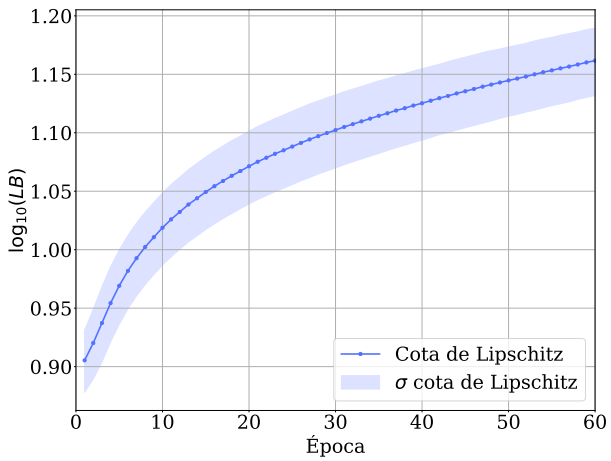
(c) Modelo 6 capas residuales

(d) Modelo 8 capas residuales

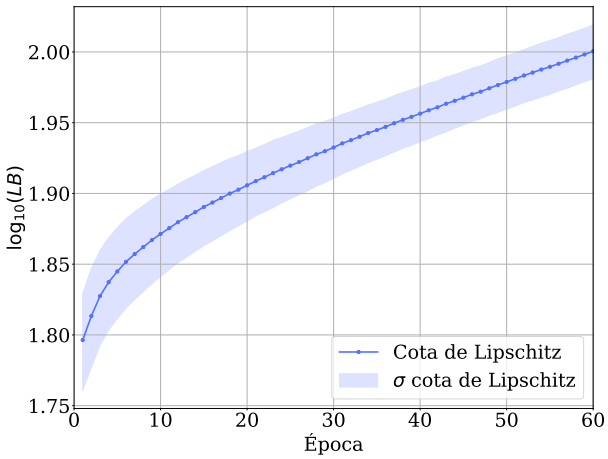
(e) Modelo 10 capas residuales

(f) Modelo 12 capas residuales

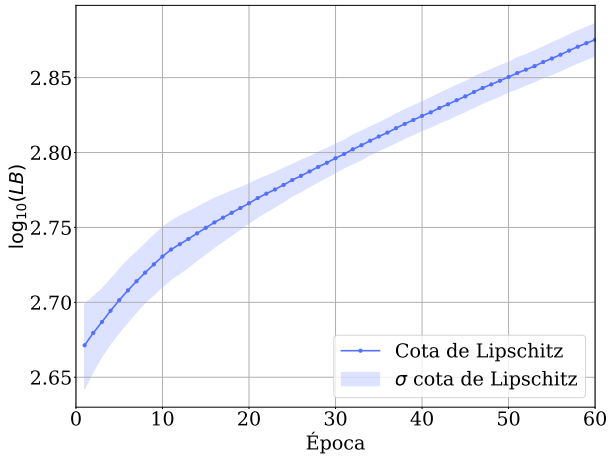
Figura 3-5.: $\log_{10}(LB)$ Cota de Lipschitz para modelos neuronales con diferente cantidad de capas residuales. Experimento A Subsección 2.3.2.



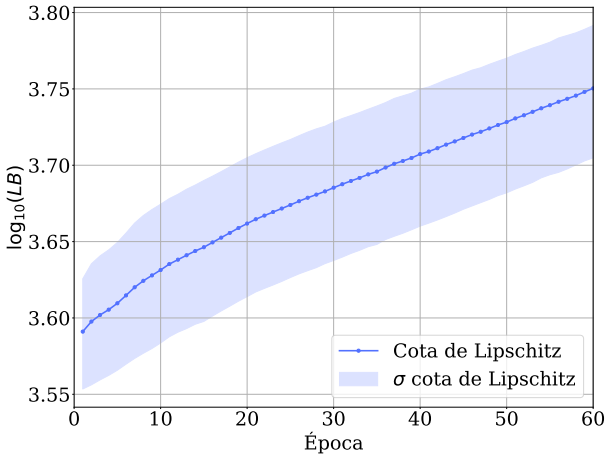
(a) Modelo 1 capas residuales



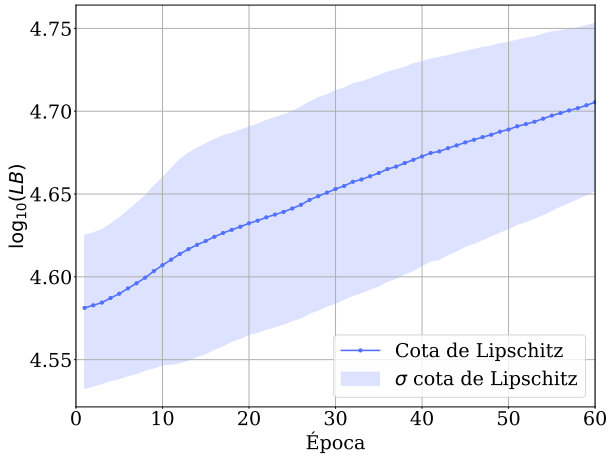
(b) Modelo 3 capas residuales



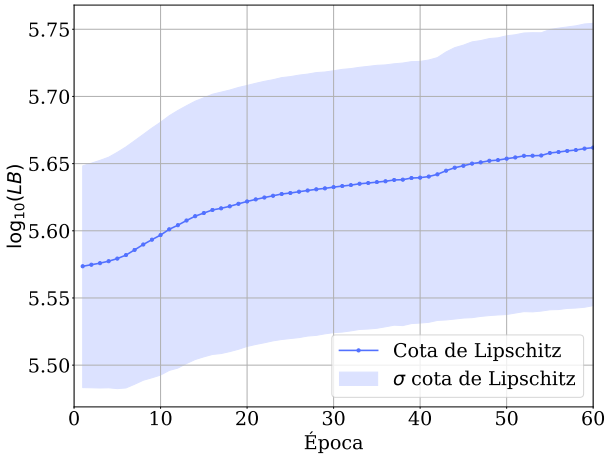
(c) Modelo 5 capas residuales



(d) Modelo 7 capas residuales



(e) Modelo 9 capas residuales



(f) Modelo 11 capas residuales

Figura 3-6.: $\log_{10}(LB)$ Cota de Lipschitz para modelos neuronales con diferente cantidad de capas residuales. Experimento B Subsección 2.3.2.

Los resultados para la constante de Lipschitz refuerzan la conclusión y permiten constatar que esta medida puede servir como guía para evaluar el sobreajuste en un modelo. Las Figuras **3-5** y **3-6** presentan la evolución de la cota de Lipschitz a través de las épocas y para cada uno de los modelos. Se puede analizar que en los modelos que resultaron con mayor sobreajuste (Ver Subsección 2.3.2) la cota alcanza un valor más elevado, que incrementa su orden al añadir capas al modelo. Lo anterior tiene sentido aritmético, ya que, la cota es el producto de las normas espectrales de todas las capas. También se observa que la cota de Lipschitz presenta una elevada variabilidad en escenarios repetitivos como el entrenamiento por validación cruzada, esta variabilidad es más acentuada en modelos complejos por ejemplo se puede notar en Figura **3-5**f y en Figura **3-6**f que la franja de la desviación entandar abarca más área.

La discusión anterior y lo desarrollado en la propuesta de restricciones aleatorias se sintetiza en la segunda modificación del algoritmo. Se propone usar como valor λ de penalización de la restricción la norma espectral de la capa aleatoria elegida en cada iteración. Cuando la capa tenga un aporte significativo a la cota de Lipschitz se penalizará fuertemente la norma de Frobenious de los pesos, en caso contrario la penalización será débil. Si en una capa los pesos tienden a cero, el algoritmo y la estructura residual recuperaran su aporte al proceso de entrenamiento, o se considerará que la capa se apagó, esto no afectará la propagación hacia adelante dado que es una estructura residual. Formalmente se presenta el algoritmo propuesto como sigue:

Modificación 2. Regularización adaptativa de Lipschitz con restricciones aleatorias

Considere el algoritmo con la Modificación 1 en el cual las regularizaciones individuales por cada capa se trabajan de forma aleatoria. Esta metodología está propuesta para algoritmos de aprendizaje con optimización por lotes, en particular para el Gradiente Estocástico y en la estructura de red residual (ResNet). Ahora, trabajando con la norma de Frobenius como función de penalización, se modifica la Ecuación 3-1 por la Ecuación 3-11. La elección del parámetro λ_a en cada iteración corresponde a la norma espectral de la capa a, denotada como $\sqrt{\rho(\mathbf{W}^{(a)T}\mathbf{W}^{(a)})}$. En el algoritmo 3.2 se presenta el pseudocódigo de esta propuesta.

$$l_r(y,\hat{y}) = l(y,\hat{y}) + \sqrt{\rho(\mathbf{W}^{(a)T}\mathbf{W}^{(a)})}||\mathbf{W}^{(a)}||_F \tag{3-11}$$

Algoritmo 3.2: Entrenamiento regularización adaptativa de Lipschitz con restricciones aleatorias

Entrada:
$\mathbf{X}$ conjunto de datos de entrenamiento,
η tasa de aprendizaje,
N_e número de épocas,
N_b número de lotes,
$f(\mathbf{X};\theta)$ modelo red neuronal residual
Salida: $\theta = \{\mathbf{W}, \mathbf{b}\}$ parámetros del modelo, pesos y bias
$\theta_0 \leftarrow$ Aleatorio;
para $epoch = 1 : N_e$ **hacer**
  Barajar datos de entrenamiento $\mathbf{X}_s \leftarrow \mathbf{X}$;
  Generar N_b lotes del conjunto de entrenamiento barajado $\mathbf{X}_s$;
  $i \leftarrow 0$;
  para $batch = 1 : N_b$ **hacer**
    Generar índice capa residual aleatoria $a \leftarrow$ Aleatorio;
    Calcular la norma espectral de la capa a $\lambda_a \leftarrow \sqrt{\rho(\mathbf{W}^{(a)T}\mathbf{W}^{(a)})}$;
    Calcular gradiente función de costo
      $\hat{y} = f(\mathbf{X}[batch];\theta_i)$;
      $\nabla_{l_r} = \nabla_{\theta_i}(l(y,\hat{y}) + \lambda_a||\mathbf{W}^{(a)}||_F)$;
    Encontrar $\theta_{i+1} \leftarrow \theta_i - \eta\nabla_{l_r}$;
    $i \leftarrow i + 1$;
  $\theta_0 \leftarrow \theta_{i+1}$;
devolver *[θ_0]*

3.2. Experimentos computacionales

En esta sección se presentan algunos resultados comparativos del desempeño del método propuesto. En la Subsección 3.2.1 se proponen algunos experimentos evaluados con conjuntos estándar de datos y en la Subsección 3.2.2 se evalúa la robustez del clasificador al enfrentarse a imágenes adversarias.

3.2.1. Validación en conjuntos estándar de datos

Los experimentos de este apartado se enfocan en comparar el desempeño del método propuesto con técnicas convencionales para la reducción del sobreajuste. En los experimentos se contrastan los resultados del entrenamiento con los métodos de regularización L1, L2 y la técnica de *Dropout*, además se construyeron diferentes arquitecturas variando el número

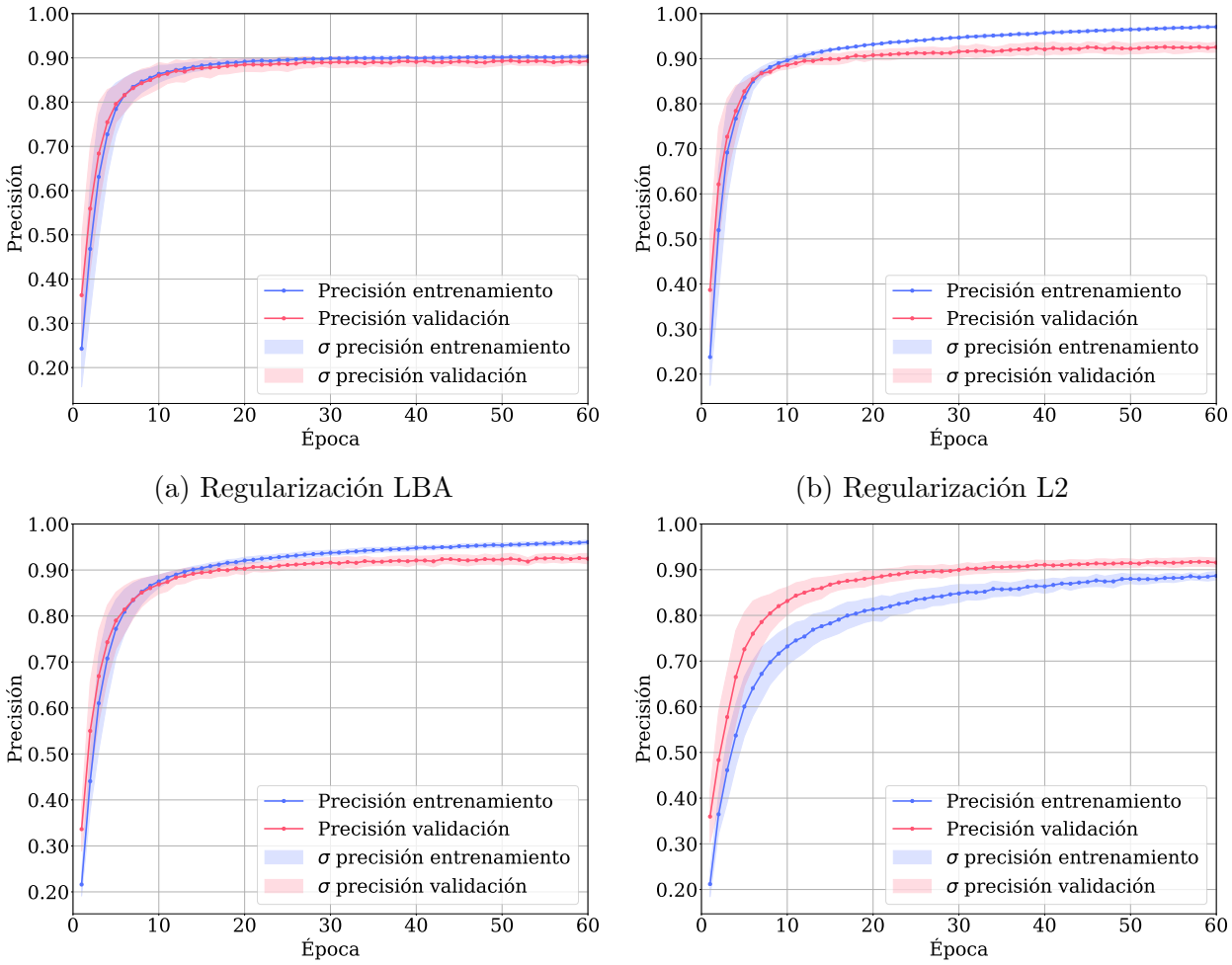
de capas residuales del modelo. Es importante señalar que para los métodos L1 y L2 se realizó una sintonización previa del parámetro λ utilizando una búsqueda de grilla para cada arquitectura neuronal, en la selección se optó por el valor que mejor resultado promedio presentara en la evaluación con el conjunto de validación. Metodología similar se aplicó en el caso del método de *Dropout*, la probabilidad de desconexión, aplicada en todas las capas de la arquitectura, se eligió después de una búsqueda de grilla con validación cruzada. Las arquitecturas de red para los Experimentos A y B coinciden con la propuesta en la Figura **2-7** y evaluados con diferente numero de capas residuales. En el caso de los experimentos C y D se fijó un número de capas residuales y se acondicionó la capa de entrada de acuerdo con el número de características del conjunto de datos.

Experimento A

Para este experimento se empleó el conjunto de datos MNIST de imágenes de escritura de los dígitos. De forma similar al escenario del **Experimento A** de la Subsección 2.3.2 se emplearon 10.000 imágenes de entrenamiento y 10.000 de prueba, con validación cruzada de 5 pliegues. Los demás parámetros de configuración se fijaron igual que en el caso del **Experimento A** (Subsección 2.3.2). Las pruebas se realizaron para un número de capas residuales de 4, 6 y 8.

En la Figura **3-7** se presenta la curva de aprendizaje para el modelo neuronal de 4 capas residuales. El método propuesto el cual se denominó **Regularización adaptativa de Lipschitz con restricciones aleatorias**, se notará como **Regularización LBA**. Para esta arquitectura los resultados indican que en el método de regularización LBA las curvas de aprendizaje mantiene un comportamiento similar tanto en entrenamiento como en validación, esta característica resalta la factibilidad del método para contrarrestar el sobreajuste. La regularización L1 y L2 generan curvas más distanciadas para los datos de entrenamiento y prueba. En el método de *Dropout* se puede observar que la curva de validación se ubica por encima que la de entrenamiento.

La Figura **3-7** también indica que la variabilidad de los modelos con diferentes técnicas de entrenamiento cambia para cada una. En la regularización LBA se observa en las primeras épocas una variación mayor, sin embargo en la etapa de convergencia esta variación se reduce. En el caso de la regularización L2, también se observa una zona con variación más pronunciada al inicio del proceso de aprendizaje, similar a los resultados de la regularización L1. En el caso del método de *Dropout* las franjas de la desviación estándar se mantienen constantes alrededor del valor promedio de la precisión, estas franjas disminuyen a medida que se incrementan las épocas.



(a) Regularización LBA

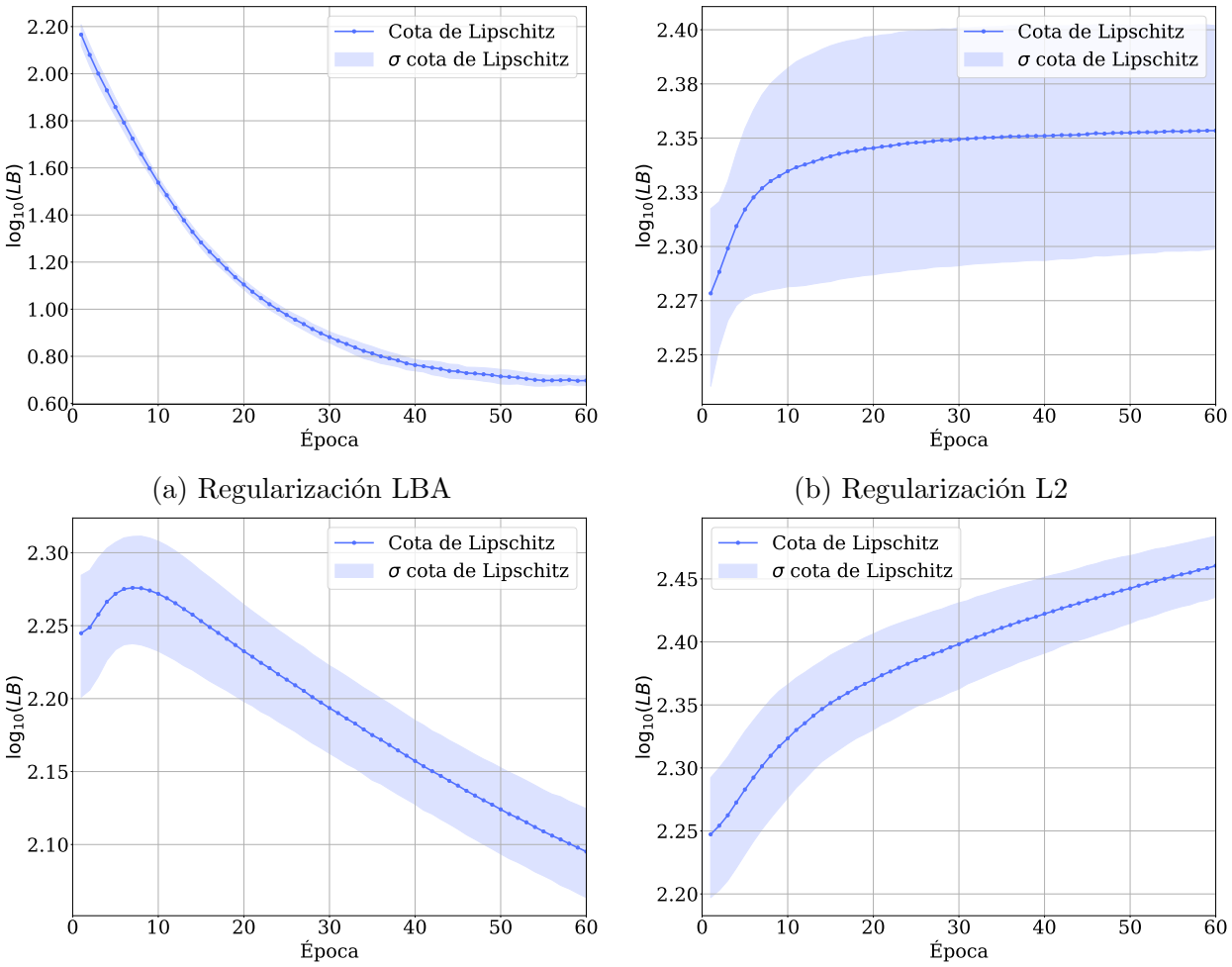
(b) Regularización L2

(c) Regularización L1

(d) Regularización *Dropout*

Figura 3-7.: Curvas de aprendizaje modelo de 4 capas residuales. Experimento A.

La Figura **3-8** presenta el comportamiento de la cota de Lipschitz, en escala logarítmica, para el modelo de 4 capas residuales. Como se puede observar, al emplear el método de la regularización LBA la curva decrece, en la escala logarítmica preserva un decaimiento exponencial. La regularización L1 también indica un comportamiento decreciente para la cota de Lipschitz, sin embargo, presenta franjas de desviación estándar acentuadas. El método de regularización L2 establece una cota de carácter incremental, que converge en su valor medio a partir de la época 20. En el caso de *Dropout* también se tiene un comportamiento creciente, pero no se evidencia un valor de convergencia, en este caso también son pronunciadas las franjas que ilustran la desviación estándar. Para esta arquitectura el método que menor variabilidad presenta en la cota de Lipschitz es la regularización LBA, también es la curva que menor valor alcanza.



(a) Regularización LBA

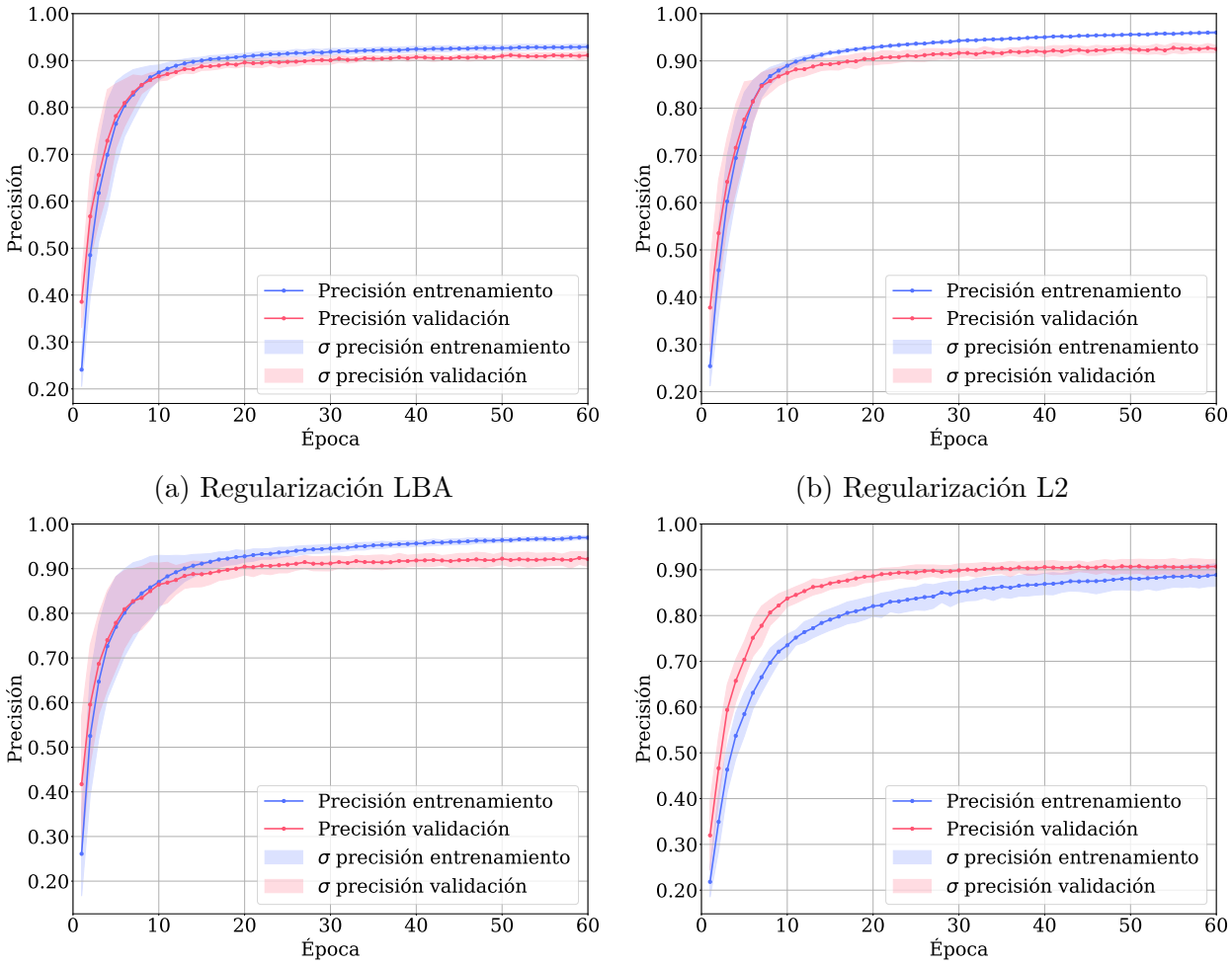
(b) Regularización L2

(c) Regularización L1

(d) Regularización *Dropout*

Figura 3-8.: $\log_{10}(LB)$ Cota de Lipschitz modelo de 4 capas residuales. Experimento A.

Los resultados para el modelo neuronal de 6 capas se presentan en la Figura **3-9**. Para este modelo los resultados son bastante similares a los expuesto en el modelo de 4 capas, con la diferencia que todas las curvas de aprendizaje tienden a incrementar la diferencia entre entrenamiento y validación. De los métodos de entrenamiento presentados la regularización LBA continua siendo la que menor diferencia presenta entre los resultados de entrenamiento y validación. En el caso de la regularización L1 se observa una área de alta variabilidad en la cual las franjas de desviación estándar están pronunciadas, este fenómeno ocurre en las primeras capas. El método de regularización LBA también presenta esta característica, aunque menos acentuada que en el caso L1. En el caso de la regularización por *Dropout* se continua la tendencia a tener una curva de validación por encima de la de aprendizaje.



(a) Regularización LBA

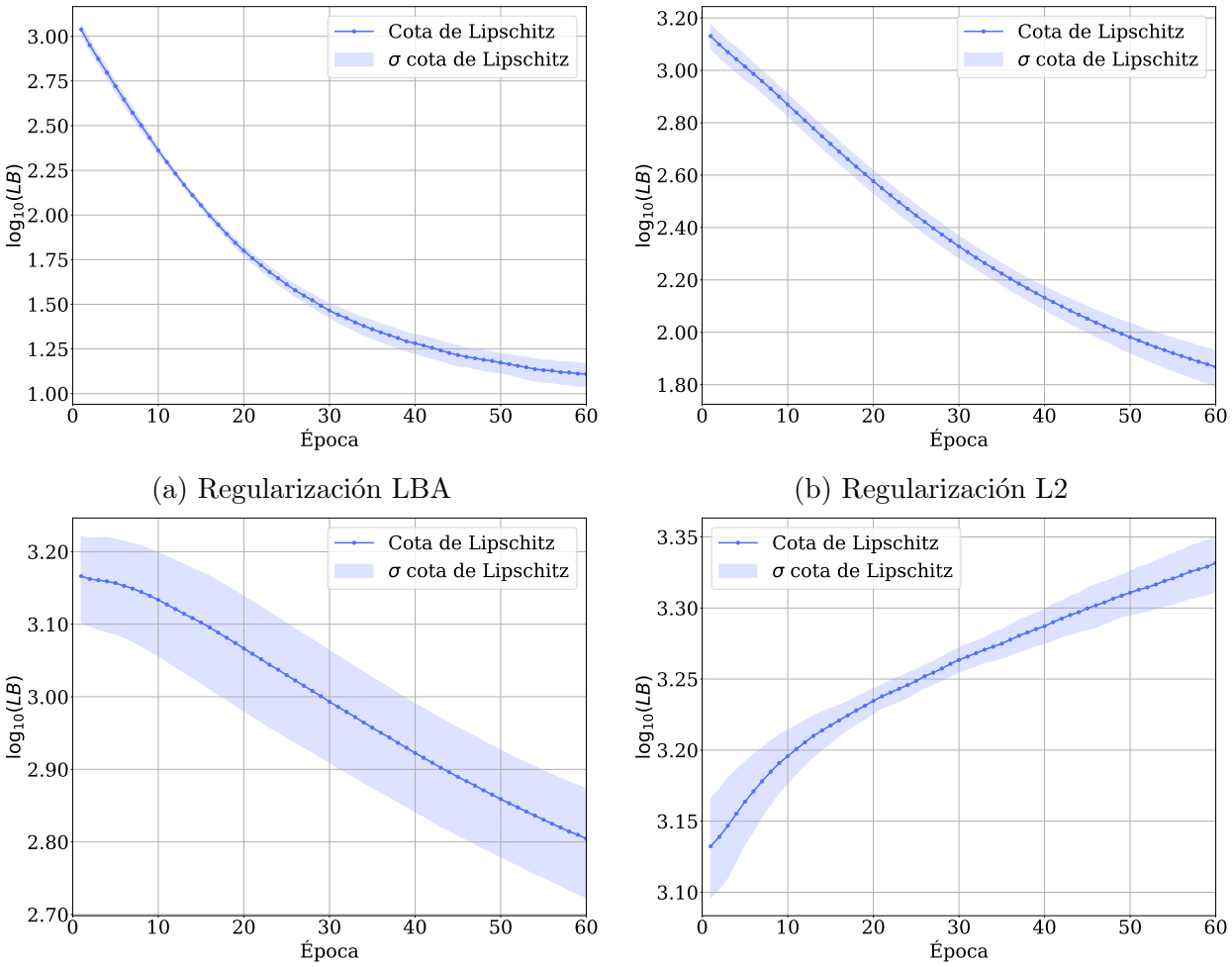
(b) Regularización L2

(c) Regularización L1

(d) Regularización *Dropout*

Figura 3-9.: Curvas de aprendizaje modelo de 6 capas residuales. Experimento A.

Para la arquitectura de 6 capas residuales la cota de Lipschitz se ilustra en la Figura **3-10**. En el método de regularización LBA se puede observar el decaimiento en la cota de Lipschitz, además la curva mantiene franjas de desviación estándar que abarcan menor área en comparación con los otros métodos. La regularización L1 y L2 también generan comportamientos decrecientes en la cota de Lipschitz, pero ambos presentan mayor variabilidad en comparación con la regularización LBA. A diferencia de estos métodos, la cota de Lipschitz tiene un comportamiento incremental para la estrategia de regularización *Dropout*.



(a) Regularización LBA

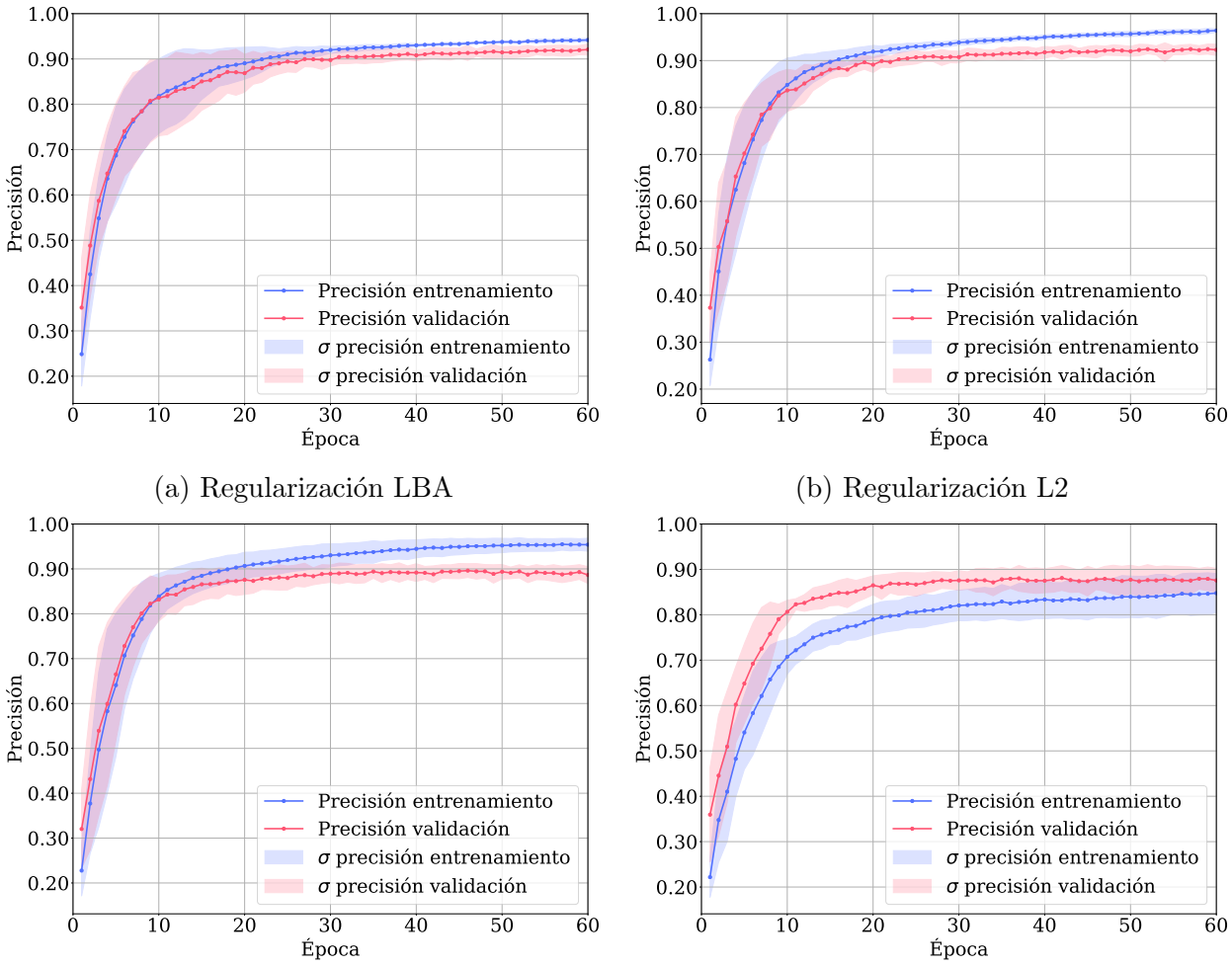
(b) Regularización L2

(c) Regularización L1

(d) Regularización *Dropout*

Figura 3-10.: $\log_{10}(LB)$ Cota de Lipschitz modelo de 6 capas residuales. Experimento A.

En la ultima arquitectura los resultados de las curvas de aprendizaje presentados en la Figura **3-11** indican que se conserva la tendencia a mantener comportamientos similares entre validación entrenamiento para el método de regularización LBA. En las curvas de aprendizaje para las regularizaciones L1 y L2 se puede evidenciar mayor diferencia entre los resultados de validación y entrenamiento, además, de una elevada variabilidad reflejadas en el aumento de las franjas de desviación estándar. En el caso del método de regularización LBA en las primeras épocas se observar una área marcada de variabilidad, sin embargo, cuando se aproxima a la zona de convergencia esta variabilidad disminuye. El método de *Dropout* conserva la tendencia de generar curvas de validación que se ubican por encima de las de aprendizaje, sin embargo, para esta arquitectura con más capas, la variabilidad aumenta, incrementando la dependencia de los resultados con el conjunto de entrenamiento.



(a) Regularización LBA

(b) Regularización L2

(c) Regularización L1

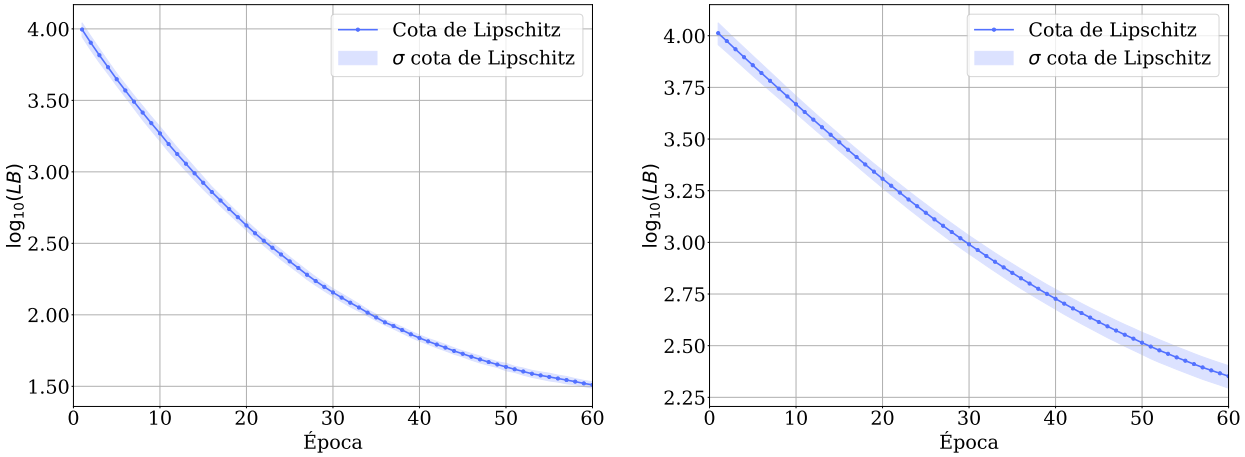
(d) Regularización *Dropout*

Figura 3-11.: Curvas de aprendizaje modelo de 8 capas residuales. Experimento A.

En cuanto a la variabilidad general de las curvas de aprendizaje de los modelos al cambiar de conjuntos de entrenamiento se observa que al incrementar el número de capas se tiene un incremento en la variación que se sufre. En especial los modelos presentan altas variaciones en las épocas previas a alcanzar convergencia. Los métodos que más variabilidad exponen son la regularización L1 y LBA, no obstante, en la etapa de convergencia LBA disminuye la variabilidad, tanto para el conjunto de prueba como de validación.

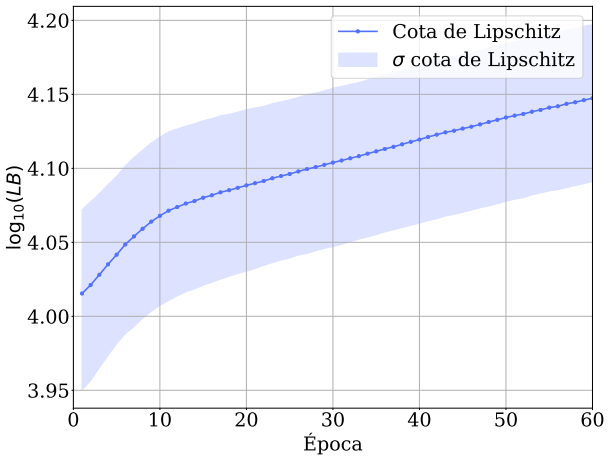
La evolución de la cota de Lipschitz para la arquitectura de 8 capas residuales es presentada en escala logarítmica en la Figura **3-12**. Se puede evidenciar que en el método de regularización LBA se mantienen los resultados alcanzados en las otras arquitecturas, la cota de Lipschitz presenta un comportamiento decreciente. Estas dinámicas también se replican en la regularización L2, aunque, las curvas muestran mayor variabilidad. En el caso de la

regularización L1 y *Dropout* las curvas tienden a incrementar conforme avanza el proceso de aprendizaje. En ambos métodos se puede observar una variabilidad alta, ilustrada por las franjas de desviación estándar.

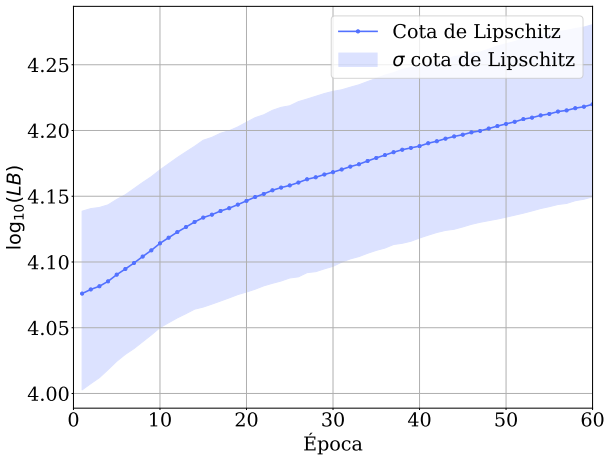


(a) Regularización LBA

(b) Regularización L2



(c) Regularización L1



(d) Regularización *Dropout*

Figura 3-12.: $\log_{10}(LB)$ Cota de Lipschitz modelo de 8 capas residuales. Experimento A.

En general, el comportamiento de la cota de Lipschitz es distinto al método convencional presentado en la Figura **3-5**, en el método LBA se observa que independiente de la arquitectura se disminuye el valor de la cota de Lipschitz. Un comportamiento similar ocurre para los métodos L1 y L2, sin embargo, no es constante a través del número de capas residuales de la arquitectura. Por ejemplo, en Figura **3-8**b la cota crece hasta un punto y se mantiene constante, mientras que en Figura **3-10**b y Figura **3-12**b disminuye su valor. En el caso del método *Dropout* se observa un crecimiento de la cota que se compara con el entrenamiento convencional, pero no es exactamente igual.

Finalizado el entrenamiento de las estructuras residuales se realizó la evaluación con el conjunto de prueba que dispone de 10.000 imágenes. Los resultados de esta evaluación se presenta en la Tabla **3-1**. Las métricas $\%Train\bar{A}_{cc}$ y $\%Test\bar{A}_{cc}$ corresponden al porcentaje de la precisión promedio al evaluar el modelo con los conjuntos de entrenamiento y prueba, respectivamente. Los valores de $std(Train\bar{A}_{cc})$ y $std(Test\bar{A}_{cc})$ indican la desviación estándar a través de los pliegues de la validación cruzada para los valores de la precisión en entrenamiento y prueba. En la Tabla **3-1**, también se indica el tiempo empleado en el entrenamiento y la diferencia entre la precisión promedio de entrenamiento y prueba, denotado como $\%|Train\bar{A}_{cc} - Test\bar{A}_{cc}|$. Estas mismas convenciones se utilizarán para las tablas de evaluación de los experimentos de esta sección. En cada tabla se resalta en negrita los valores máximos para las precisiones promedios y los mínimos para las desviaciones, la diferencia entre los promedios de precisión en entrenamiento y prueba, y para el tiempo.

En los resultados de evaluación se puede observar, para el modelo de 4 capas residuales, que el método de regularización LBA presenta una diferencia menor entre lo logrado en entrenamiento y prueba. El mejor resultado en precisión se logra al aplicar regularización L2, sin embargo, es el método con mayor diferencia entre el entrenamiento y prueba. El método con menor tiempo de entrenamiento es la regularización L1, la cual también presenta la menor desviación estándar para los datos de prueba. El método de *Dropout* resulta ser el menos conveniente, presenta altos valores para la desviación estándar y bajos valores de precisión.

La Tabla **3-1** también nos indica los resultados para la arquitectura de 6 capas, se puede observar que la menor diferencia de la precisión promedio con los datos de entrenamiento y prueba es para el método *Dropout*, no obstante, es el que menor precisión alcanza. Situación similar sucede en el modelo de 8 capas, *Dropout* señala la menor diferencia pero es el método de mas bajo desempeño para la métrica de precisión. Las regularizaciones L1 y L2 alcanzan valores altos de precisión pero son los método en los cuales más se acentúa la diferencia entre el conjunto de prueba y entrenamiento.

Independientemente de la arquitectura el método de regularización LBA mantiene las desviaciones estándar alrededor de un valor, contrario al comportamiento de los demás métodos que cambian estos valores al agregar más capas residuales. El tiempo de entrenamiento señala que el método de regularización LBA requiere más tiempo, no obstante, este valor no difiere significativamente de los otros métodos, incluso para la arquitectura de 8 capas el método L2 resulta en un tiempo mayor.

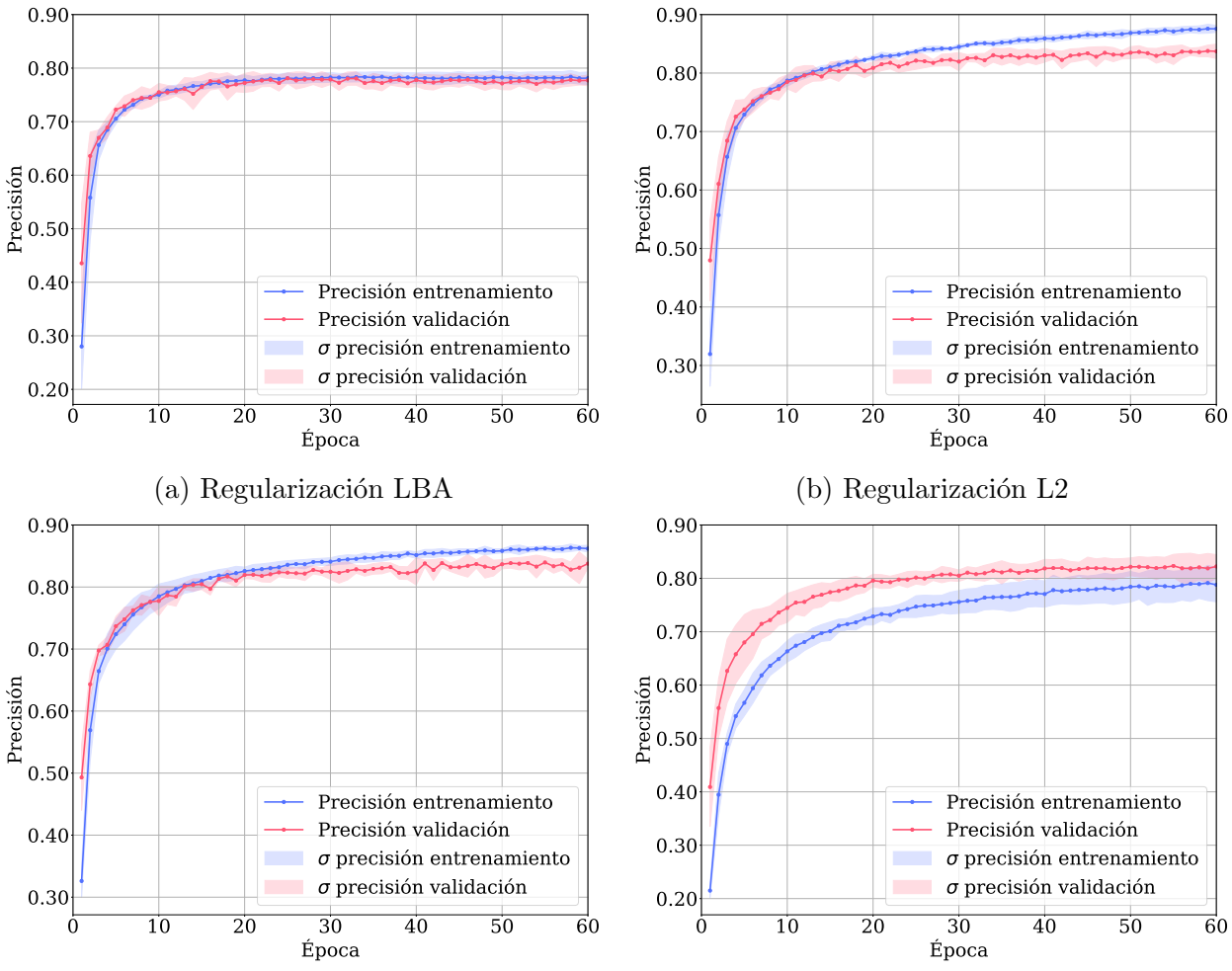
Tabla 3-1.: Resultados de la precisión en el entrenamiento y la prueba de los diferentes modelos de redes neuronales residuales para el Experimento A de la Subsección 3.2.1.

# de capas residuales	Métricas	LBA	L2	L1	*Dropout*
4	% $Train\bar{A}_{cc}$	90.29	**97.06**	95.92	88.68
	% $Test\bar{A}_{cc}$	88.67	**92.20**	92.19	91.22
	% $\lvert Train\bar{A}_{cc} - Test\bar{A}_{cc}\rvert$	**1.62**	4.86	3.73	2.54
	$\pm$ $std(Train\bar{A}_{cc})$	0.377	**0.211**	0.382	3.865
	$\pm$ $std(Test\bar{A}_{cc})$	0.356	0.257	**0.215**	0.477
	Tiempo de entrenamiento [s]	144	136	**116**	122
6	% $Train\bar{A}_{cc}$	92.93	96.01	**96.96**	88.87
	% $Test\bar{A}_{cc}$	90.63	**92.32**	92.05	90.09
	% $\lvert Train\bar{A}_{cc} - Test\bar{A}_{cc}\rvert$	2.30	3.69	4.91	**1.22**
	$\pm$ $std(Train\bar{A}_{cc})$	0.353	0.260	**0.207**	2.221
	$\pm$ $std(Test\bar{A}_{cc})$	0.394	0.231	**0.187**	1.374
	Tiempo de entrenamiento [s]	159	157	132	**132**
8	% $Train\bar{A}_{cc}$	94.19	96.41	**97.34**	84.77
	% $Test\bar{A}_{cc}$	91.17	**91.99**	89.17	87.38
	% $\lvert Train\bar{A}_{cc} - Test\bar{A}_{cc}\rvert$	3.02	4.42	8.17	**2.61**
	$\pm$ $std(Train\bar{A}_{cc})$	0.356	**0.345**	1.736	3.865
	$\pm$ $std(Test\bar{A}_{cc})$	0**.300**	0.750	1.788	2.404
	Tiempo de entrenamiento [s]	175	183	**148**	**148**

Experimento B

En este experimento se empleó el conjunto de datos *Fashion* MNIST utilizado en el **Experimento B** de la Subsección 3.2.1. Se usaron 10.000 imágenes en el conjunto de entrenamiento y el mismo número para prueba, los demás parámetros se fijaron igual que la configuración del Experimento B (Subsección 3.2.1). El experimento se realizó para arquitecturas neuronales de 3, 5 y 7 capas residuales con la estructura que presenta la Figura **2-7**.

La Figura **3-13** presenta las curvas de aprendizaje contrastando el resultado logrado con el método propuesto de Regularización LBA y los métodos L1, L2 y *Dropout*. En las gráficas se puede observar un comportamiento análogo al Experimento A de esta sección, el método de regularización LBA es, entre los métodos que se probaron, el que mejor mantienen la relación de las curvas con los datos de entrenamiento y validación. La regularización L2 y L1 generan curvas con diferencias en los resultados de entrenamiento y validación. Por su parte el método de *Dropout* presenta resultado de validación por encima que los alcanzados en entrenamiento. La variabilidad no es muy alta para los método de regularización LBA, L1 y L2, pero si se evidencia mayor tendencia a variaciones en *Dropout*.



(a) Regularización LBA

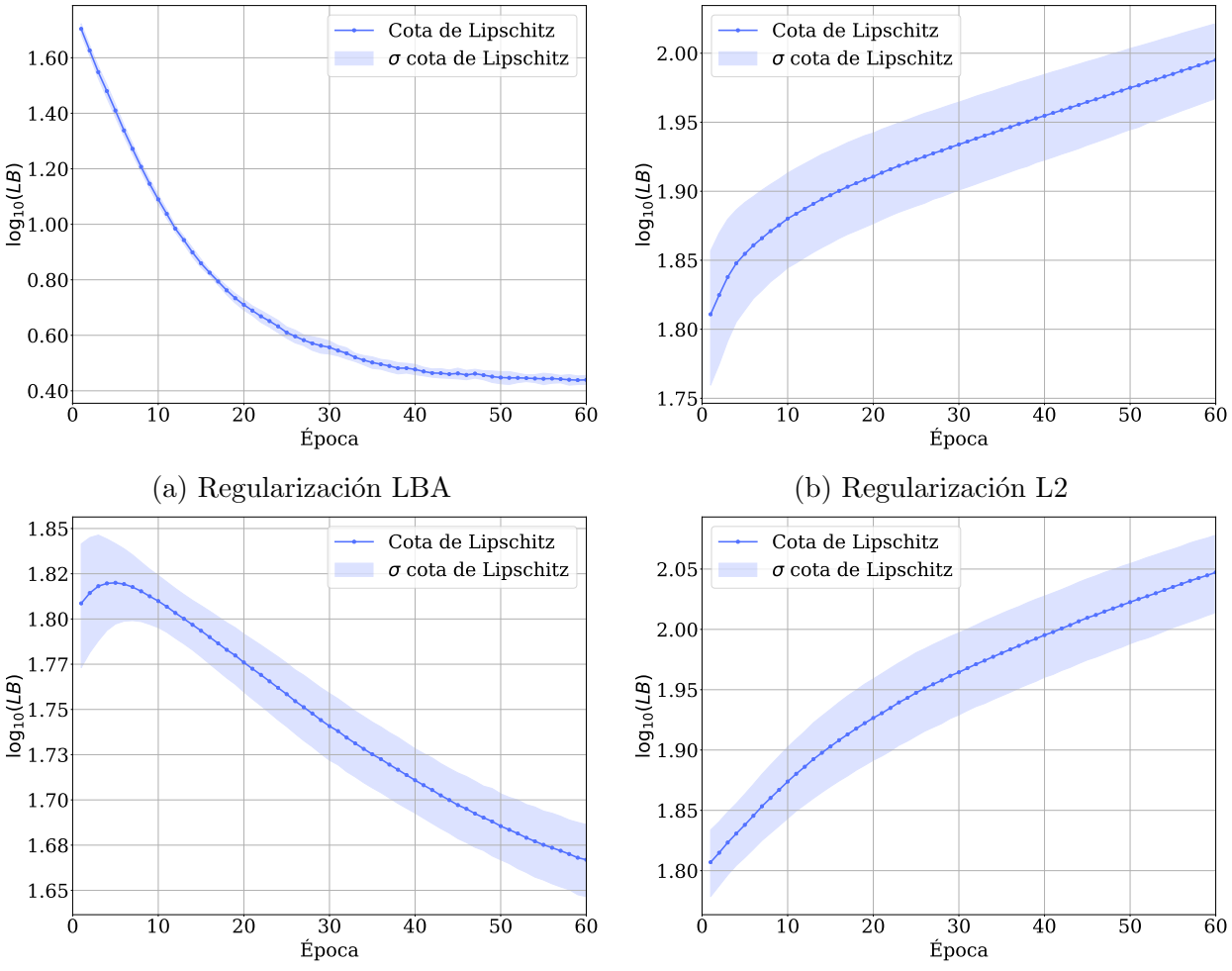
(b) Regularización L2

(c) Regularización L1

(d) Regularización *Dropout*

Figura 3-13.: Curvas de aprendizaje modelo de 3 capas residuales. Experimento B.

En la Figura **3-14** se muestran las cotas de Lipschitz en escala logarítmica. Para esta arquitectura se puede observar que el comportamiento de la cota se reduce a través del proceso de aprendizaje al emplear el método de regularización LBA. Una situación similar se presenta en la regularización L1, sin embargo, el decaimiento es menos pronunciado y se observa en las primeras épocas una tendencia incremental que luego decae. En la regularización L1 también se observan marcadas franjas de desviación estándar, señalando una dependencia mayor del conjunto de entrenamiento. En el caso de regularización L2 y *Dropout* el comportamiento de la cota es incremental, en ambas técnicas la cota aumenta sin convergencia a ningún valor. Además, se reflejan franjas de desviación estándar ampliamente marcadas.



(a) Regularización LBA

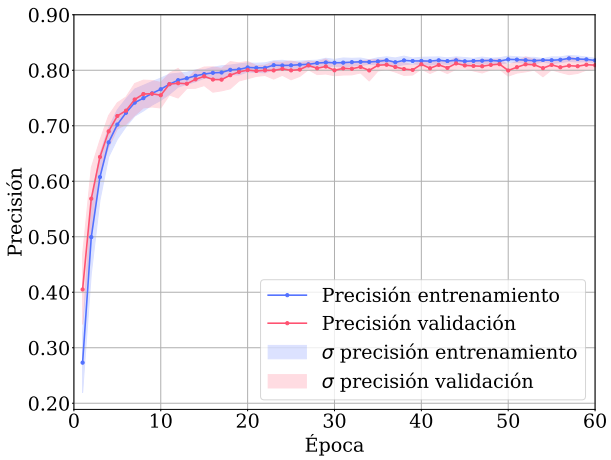
(b) Regularización L2

(c) Regularización L1

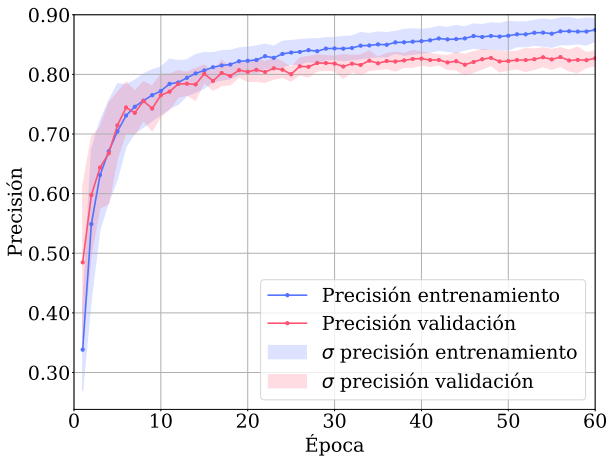
(d) Regularización *Dropout*

Figura 3-14.: $\log_{10}(LB)$ Cota de Lipschitz modelo de 3 capas residuales. Experimento B.

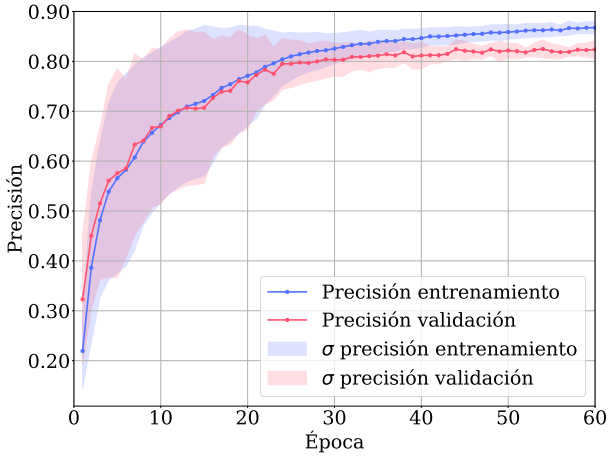
La Figura **3-15** presenta las curvas de aprendizaje para la estructura de 5 capas residuales. En las figuras se puede observar un incremento de la variabilidad, en comparación con la arquitectura anterior, por ejemplo en el caso de la regularización L1, durante la mitad del entrenamiento el área que representa la desviación estándar está ampliamente acentuada. En el caso de la regularización L2 se observa una variabilidad considerable, además, las curvas de validación y entrenamiento tiene una separación notable. El método de *Dropout* también incrementa sus variaciones, estas franjas no reducen incluso en la zona donde se esperaría el método converja. De los método presentados la regularización LBA conserva la similitud entre entrenamiento y validación, así mismo, es el método que menor variabilidad expone.



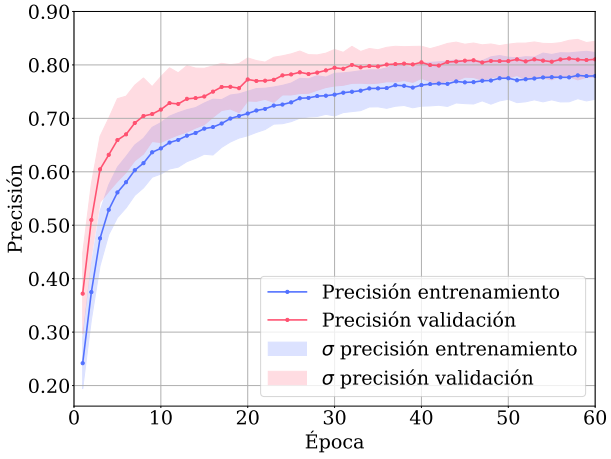
(a) Regularización LBA



(b) Regularización L2



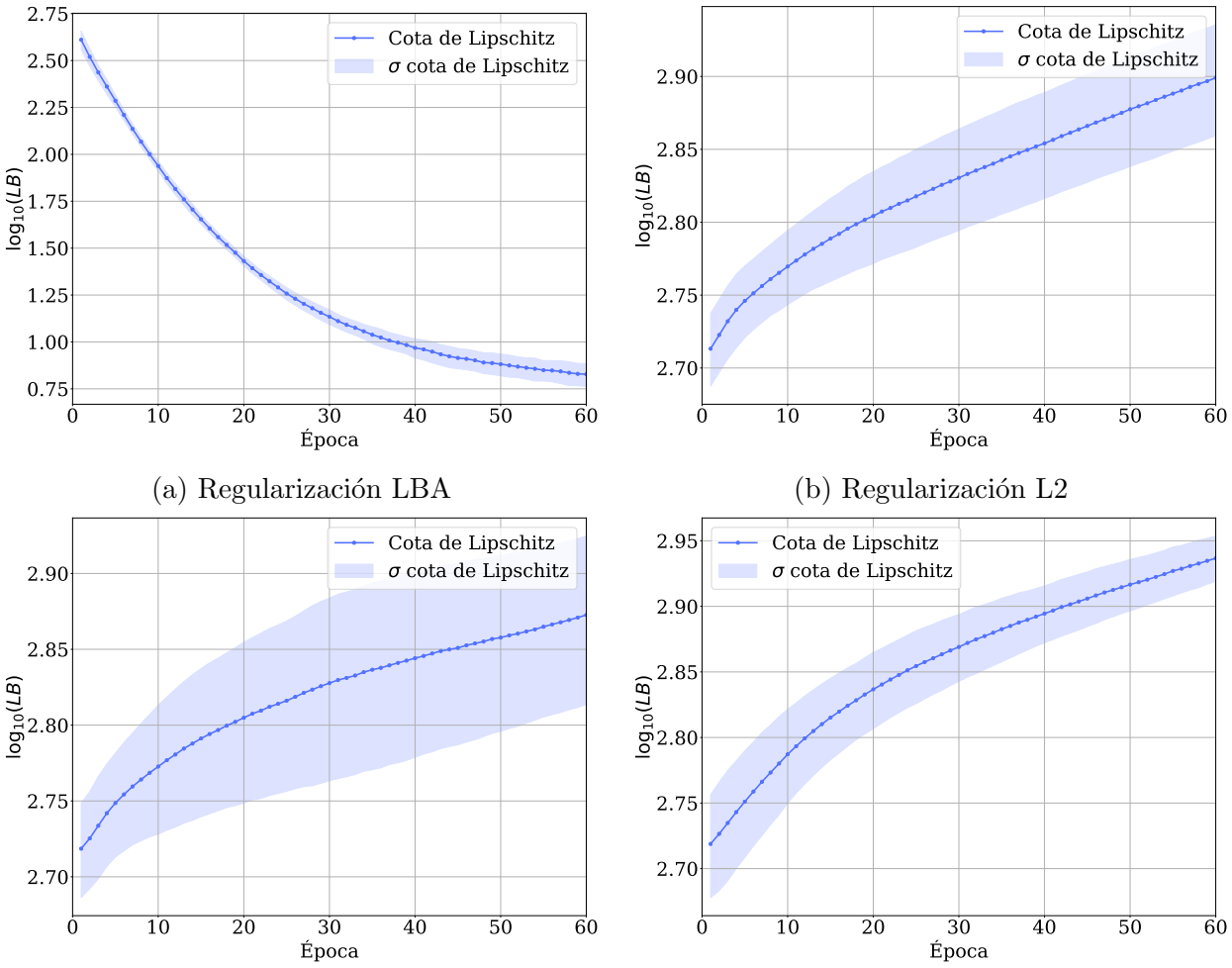
(c) Regularización L1



(d) Regularización *Dropout*

Figura 3-15.: Curvas de aprendizaje modelo de 5 capas residuales. Experimento B.

Las cotas de Lipschitz para la arquitectura de 5 capas residuales se encuentran ilustradas en la Figura **3-16**. En el método de regularización LBA se puede notar una clara disminución de la cota, además de tener una desviación estándar menor. En el caso de las regularizaciones L1, L2 y *Dropout* la tendencia de la cota es de tipo incremental, en ninguno de estos métodos la cota logra estabilizarse cerca de algún valor. El crecimiento mantiene una forma similar en los tres casos, pero la variación es notablemente mayor en la técnica L1. Los demás métodos también presentan franjas de desviación estándar más altas que la Regularización LBA. A modo de comparación, usando la regularización LBA la medida de la cota de Lipschitz es robusta a la elección del conjunto de entrenamiento con el cual se entrene el modelo.



(a) Regularización LBA

(b) Regularización L2

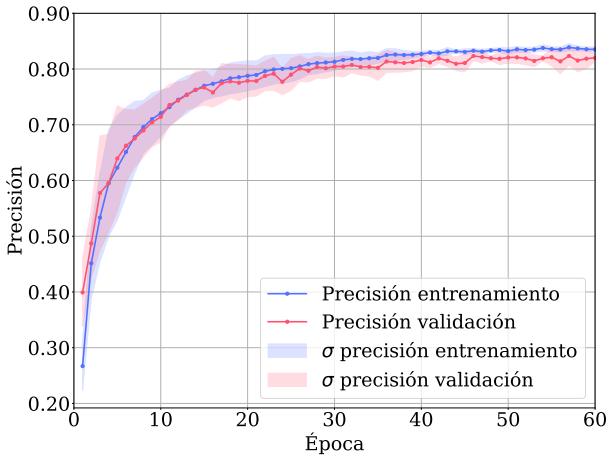
(c) Regularización L1

(d) Regularización *Dropout*

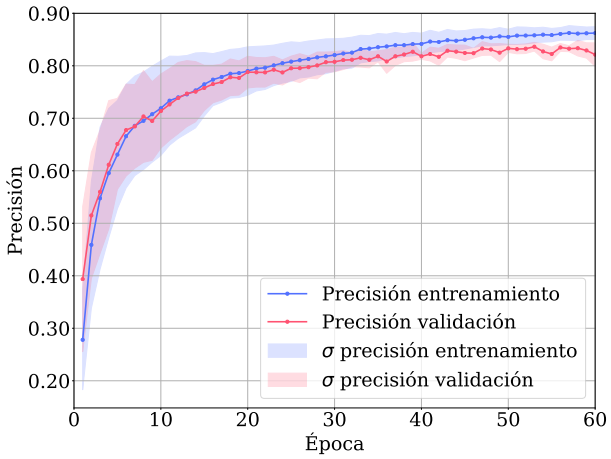
Figura 3-16.: $\log_{10}(LB)$ Cota de Lipschitz modelo de 5 capas residuales. Experimento B.

En la arquitectura final con 7 capas residuales la complejidad del modelo se incrementa, y esto impacta la variabilidad de las curvas de aprendizaje presentadas en al Figura **3-17**. Como se puede notar los métodos han incrementado las franjas que señalan la desviación

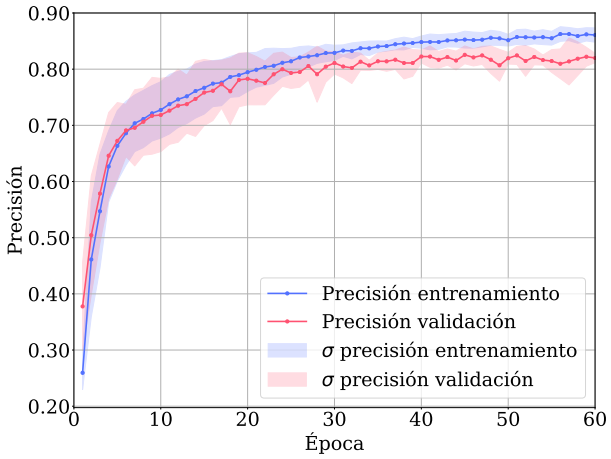
estándar, en particular para la regularización L2 y *Dropout*. La similitud entre los resultados de entrenamiento y validación se conserva para el método de regularización LBA, además, es el que menor variabilidad presenta. Otro aspecto a resaltar es que en esta arquitectura la técnica de *Dropout* no logra alcanzar una precisión adecuada disminuyendo la capacidad de aprendizaje del clasificador.



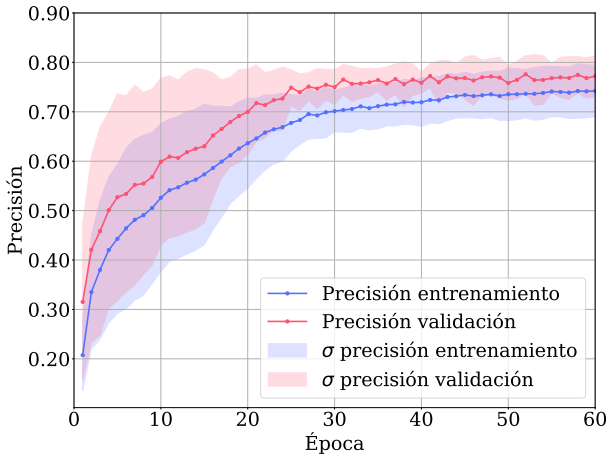
(a) Regularización LBA



(b) Regularización L2



(c) Regularización L1

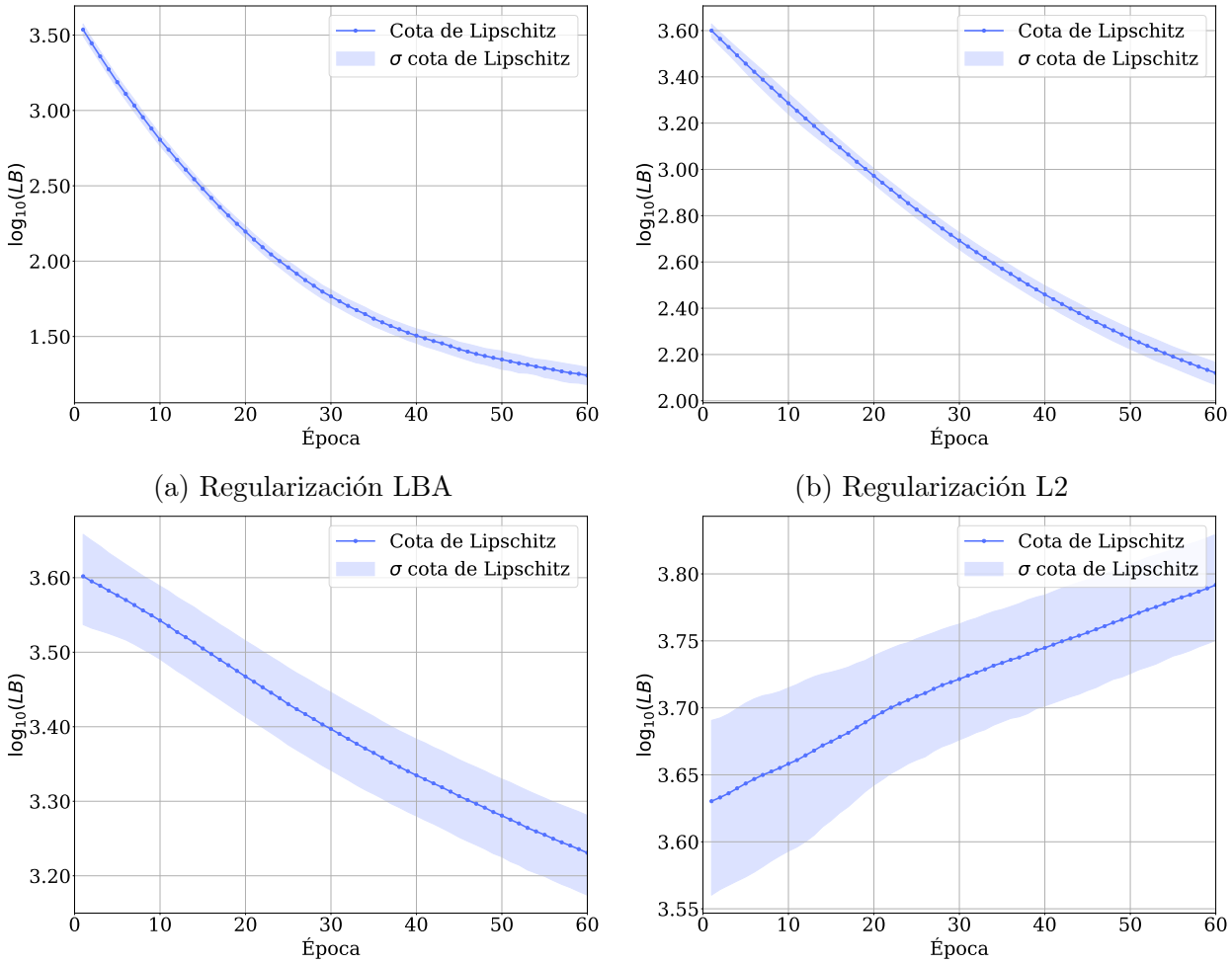


(d) Regularización *Dropout*

Figura 3-17.: Curvas de aprendizaje modelo de 7 capas residuales. Experimento B.

En general, el método de regularización LBA es, entre los métodos que se probaron, el que mejor mantienen la relación de las curvas con los datos de entrenamiento y validación. Similar a lo observado en el caso anterior, el método de *Dropout* en todas las arquitecturas presenta una curva de validación que se ubica por encima de la curva con los datos de entrenamiento. Las regularizaciones L1 y L2 alcanzan el más alto valor final con los datos de validación, pero es muy cercano al logrado en LBA, además, en ambos método se puede observar una diferencia notable entre los conjuntos de validación y prueba. Otro aspecto a resaltar es la

convergencia, el método LBA requiere menos épocas para estabilizar su comportamiento, contrario a los demás métodos que logran estabilizarse en un número mayor de épocas.



(a) Regularización LBA

(b) Regularización L2

(c) Regularización L1

(d) Regularización *Dropout*

Figura 3-18.: $\log_{10}(LB)$ Cota de Lipschitz modelo de 7 capas residuales. Experimento B.

En la Figura **3-18** se muestran las curvas de la evolución de la cota de Lipschitz para la arquitectura de 7 capas residuales. Como ocurrió las arquitecturas previas, el método de regularización LBA genera un decaimiento de la cota. Por el contrario en *Dropout* se encuentra un crecimiento sin evidente valor que permita analizar algún tipo de estabilización. Los métodos L1 y L2 para esta arquitectura también tiene cotas que decrecen, sin embargo ambos métodos tienen mayor variabilidad que la regularización LBA.

En las tres arquitecturas el método regularización LBA genera una cota de Lipschitz que disminuye a través de las épocas, este comportamiento también se obtiene para la cota de la regularización L2, únicamente en la arquitectura de 7 capas Figura **3-18**b y para L1 en las

arquitecturas de 3 y 7 capas como se muestra en la Figura **3-14**c y Figura **3-18**c. Además, el método de regularización LBA se destaca por presentar una menor variación para las curvas de la cota de Lipschitz.

Tabla 3-2.: Resultados de la precisión en el entrenamiento y la prueba de los diferentes modelos de redes neuronales residuales para el Experimento B de la Subsección 3.2.1.

# de capas residuales	**Métricas**	**LBA**	**L2**	**L1**	***Dropout***
3	% $Train\bar{A}_{cc}$	78.13	**87.56**	86.19	78.80
	% $Test\bar{A}_{cc}$	76.79	**83.03**	82.80	81.07
	% $\lvert Train\bar{A}_{cc} - Test\bar{A}_{cc}\rvert$	**1.34**	4.53	3.39	2.27
	$\pm$ $std(Train\bar{A}_{cc})$	1.039	0.539	**0.486**	2.855
	$\pm$ $std(Test\bar{A}_{cc})$	0.620	**0.391**	0.576	2.247
	Tiempo de entrenamiento [s]	121	78	117	88
5	% $Train\bar{A}_{cc}$	81.77	**87.47**	86.76	77.95
	% $Test\bar{A}_{cc}$	79.87	**81.84**	81.73	79.63
	% $\lvert Train\bar{A}_{cc} - Test\bar{A}_{cc}\rvert$	1.90	5.63	5.03	**1.68**
	$\pm$ $std(Train\bar{A}_{cc})$	**0.357**	1.741	1.080	3.920
	$\pm$ $std(Test\bar{A}_{cc})$	**0.509**	1.635	0.614	3.241
	Tiempo de entrenamiento [s]	153	**90**	135	102
7	% $Train\bar{A}_{cc}$	83.51	**86.23**	86.08	74.23
	% $Test\bar{A}_{cc}$	80.66	81.05	**81.44**	76.35
	% $\lvert Train\bar{A}_{cc} - Test\bar{A}_{cc}\rvert$	2.85	5.18	4.64	**2.12**
	$\pm$ $std(Train\bar{A}_{cc})$	**0.467**	1.094	1.134	4.644
	$\pm$ $std(Test\bar{A}_{cc})$	**0.500**	1.481	0.844	4.215
	Tiempo de entrenamiento [s]	208	**104**	153	117

Los resultados de la evaluación con el conjunto de prueba se resumen en la Tabla **3-2**. Para la arquitectura de 3 capas residuales el método de regularización LBA presenta la menor diferencia del promedio de precisión con el conjunto de entrenamiento y prueba, sin embargo, también es el método que menor precisión logra. En las arquitectura de 5 y 7 capas la regularización LBA presenta el más bajo nivel de desviación estándar, lo cual indica

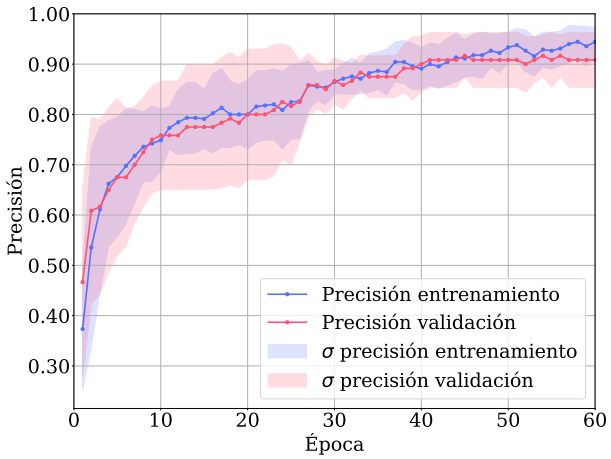
una menor dependencia del conjunto que se emplea para entrenar el modelo. Los método de regularización L1 y L2 son lo que mejor desempeño alcanzan en precisión, pero también son los que mas acentúan la diferencia entre los conjuntos de entrenamiento y prueba. Para esta métrica el método de *Dropout* logra la menor diferencia en las arquitecturas de 5 y 7 capas, pero es el método que menor precisión alcanza. En cuanto al tiempo, la regularización LBA es el método que más tarda, pero no se diferencia mucho de los otros métodos.

Experimento C

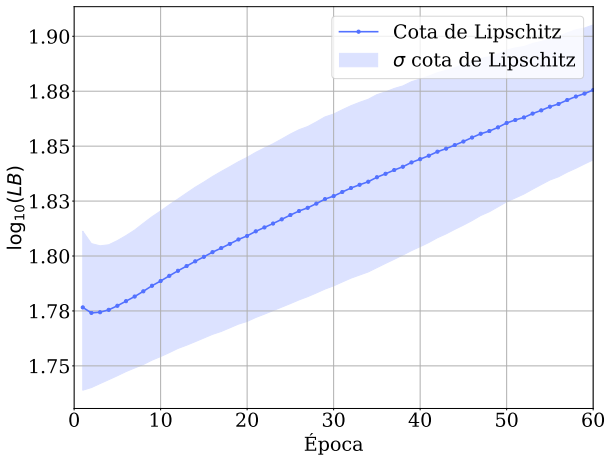
En este experimento se implementó el método de regularización LBA, L1, L2 y *Dropout* para los conjuntos de datos Flor Iris [1] [116], *Wine Dataset*[2] [117] y *Breast Cancer Wisconsin*[3] [117]. *Iris Dataset* cuenta con 120 muestras distribuidas en proporciones iguales en 3 clases, cada observación es un vector de 4 posiciones. Esta base de datos clasifica la especie de flor Iris de acuerdo con los parámetros relacionados con las medidas del sépalo y pétalo. El conjunto de datos *Wine Dataset* esta conformado por un total de 178 observaciones, identificadas en tres clases. Cada observación contiene 13 medidas resultado de un análisis químico de vinos y su clasificación establece el tipo de vino. En el ultimo conjunto de datos *Breast Cancer Wisconsin* la clasificación es binaria, discriminando entre pacientes con tumores benignos y malignos. Cada observación consta de 30 características, extraídas de una imagen mamaria, en total se cuenta con 569 muestras.

Para cada uno de los conjuntos de datos se acondicionó la arquitectura presentada en Figura **2-7** adaptando la primera capa de acuerdo con la dimensión de los datos de entrada. Para los 3 conjuntos de datos se utilizaron 3 capas residuales ocultas con 10 neuronas. En el caso de la clasificación binaria (*Breast Cancer Dataset*) en la capa de salida se consideró la función sigmoide como activación. De forma similar a los experimentos anteriores, y para todos los conjuntos de datos, se consideró un escenario repetitivo con validación cruzada en 5 pliegues. Para todos los conjuntos de datos también se realizó el entrenamiento convencional (denotado *Base Line*), sin usar ningún método de reducción de sobreajuste. Para este entrenamiento las curvas de precisión y la evolución de la cota de Lipschitz se presentan en la Figura **3-19**.

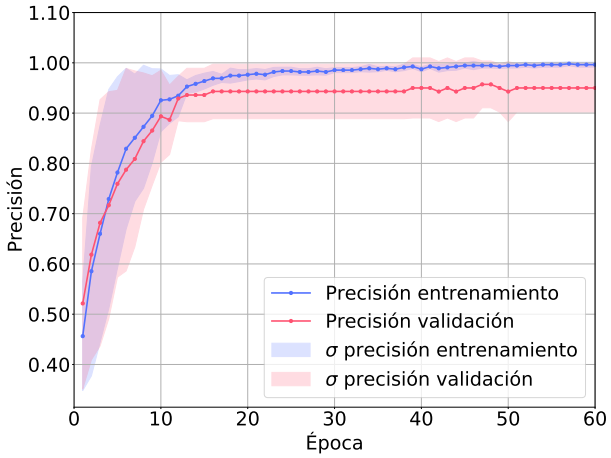
[1] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html
[2] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html
[3] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html



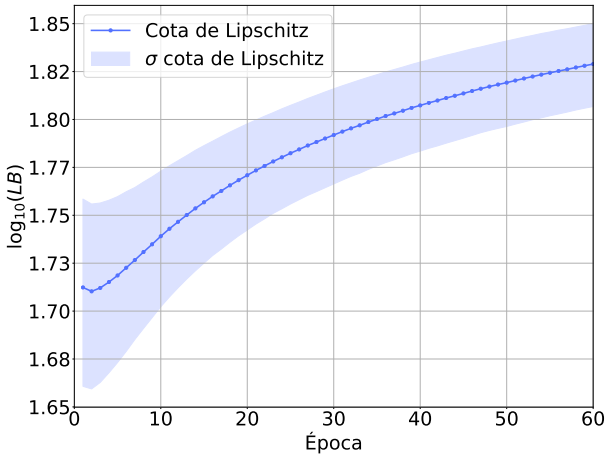
(a) Conjunto de datos Iris



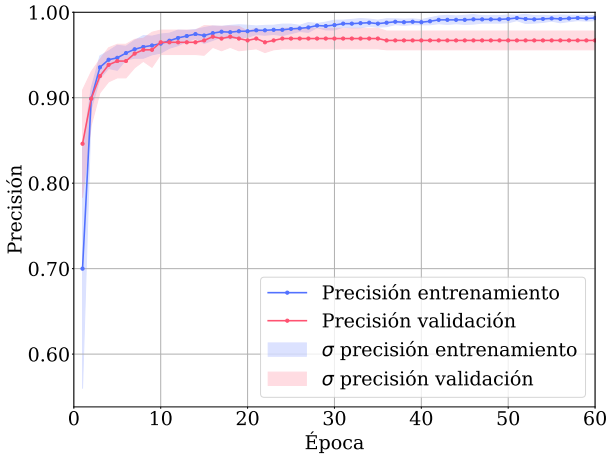
(b) Conjunto de datos Iris



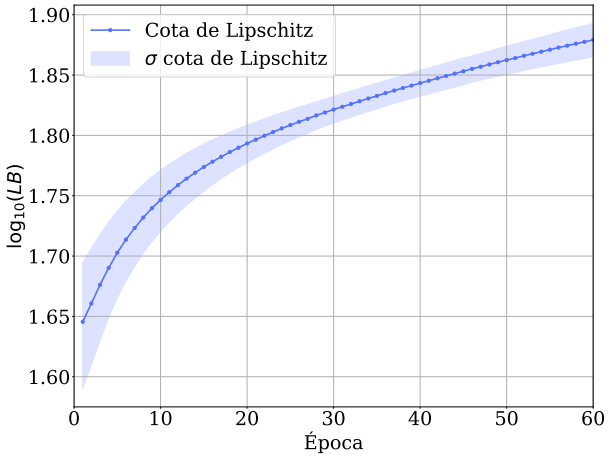
(c) Conjunto de datos *Wine*



(d) Conjunto de datos *Wine*

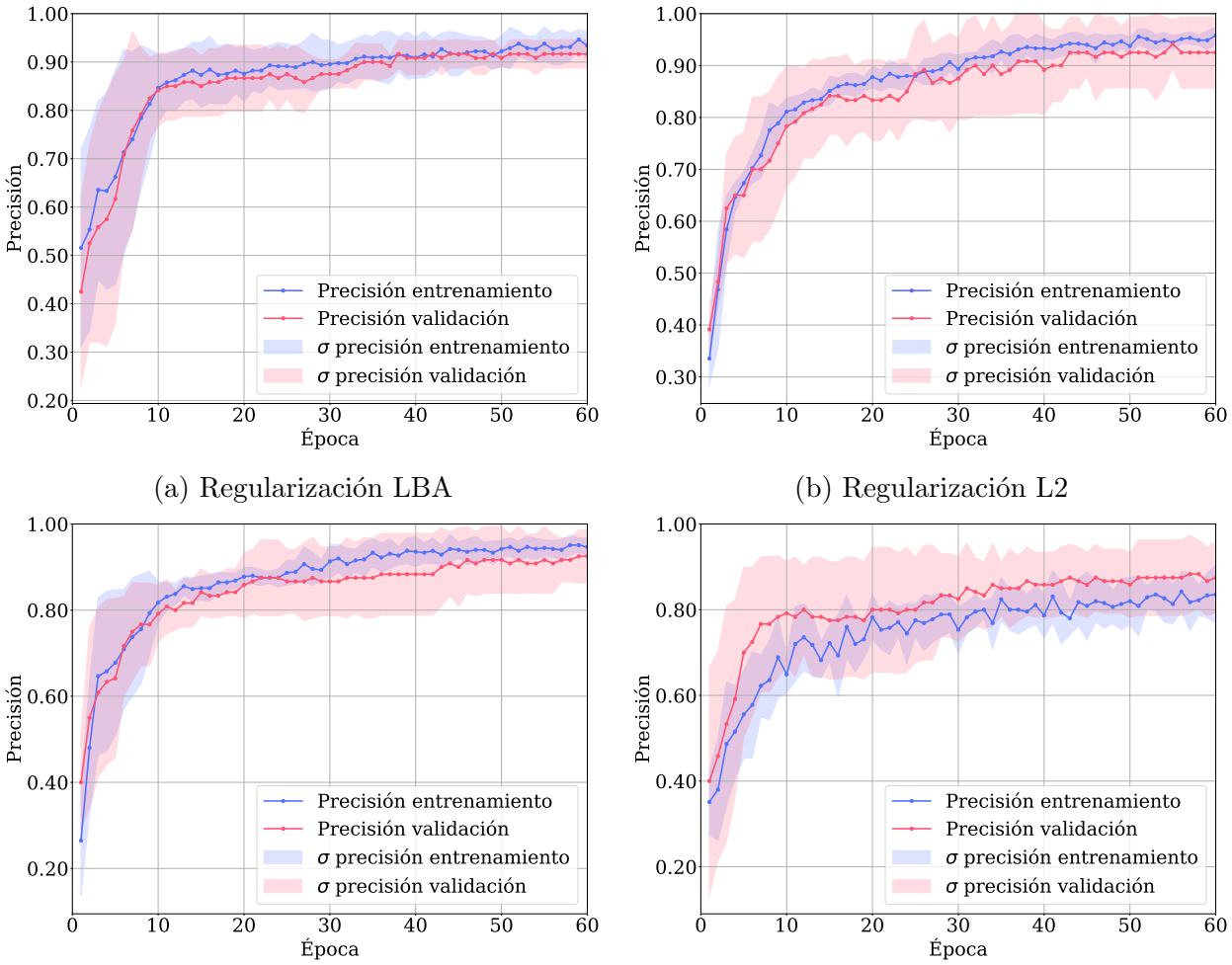


(e) Conjunto de datos *Breast Cancer*



(f) Conjunto de datos *Breast Cancer*

Figura 3-19.: Curvas de aprendizaje y $\log_{10}(LB)$ cota de Lipschitz del entrenamiento convencional. Experimento C.



(a) Regularización LBA

(b) Regularización L2

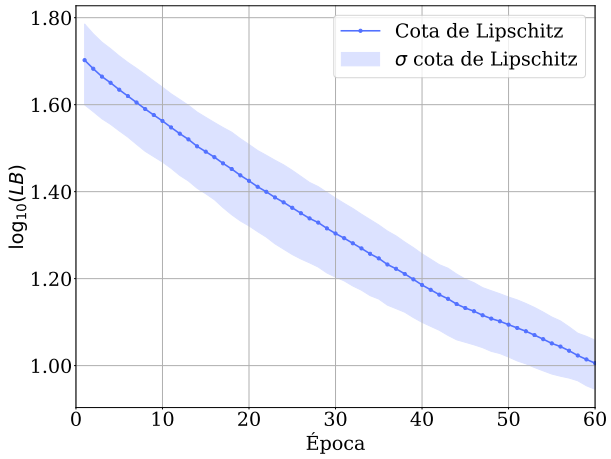
(c) Regularización L1

(d) Regularización *Dropout*

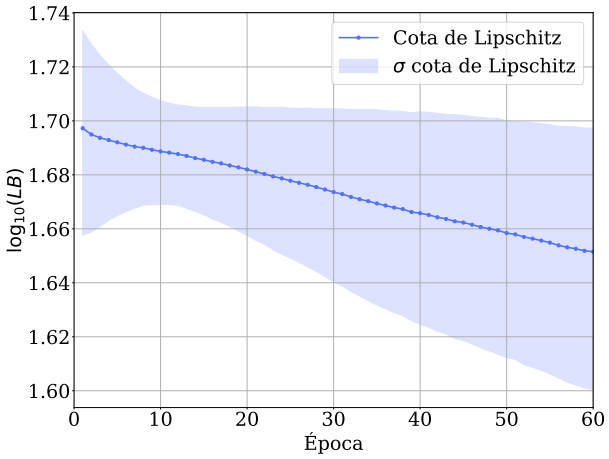
Figura 3-20.: Curvas de aprendizaje modelo neuronal con 3 capas residuales y conjunto de datos *Iris Dataset*. Experimento C.

La Figura **3-20** presenta las curvas de aprendizaje para el conjunto de datos Iris. Se puede observar que el método regularización LBA presenta curvas de precisión promedio muy cercanas entre el comportamiento de validación y prueba, como se indica en la Figura **3-20**a. Además la desviación estándar tiene franjas menos acentuadas. Los método L1 y L2 alcanzan mayor precisión en entrenamiento, pero indican mayor separación con el conjunto de prueba, también se observan franjas de desviación estándar con mayor área Figuras **3-20**c y **3-20**b. El método de *Dropout* presenta un comportamiento contrario a los demás métodos, la curva para la prueba se ubica por encima de la lograda en entrenamiento **3-20**d, este método también indica una pronunciada variabilidad. La cota de Lipschitz para este conjunto de datos se presenta en la Figura **3-21**. Se puede notar que en el método de regularización LBA (Figura **3-21**a) la cota decrece a través de las épocas, distinto al

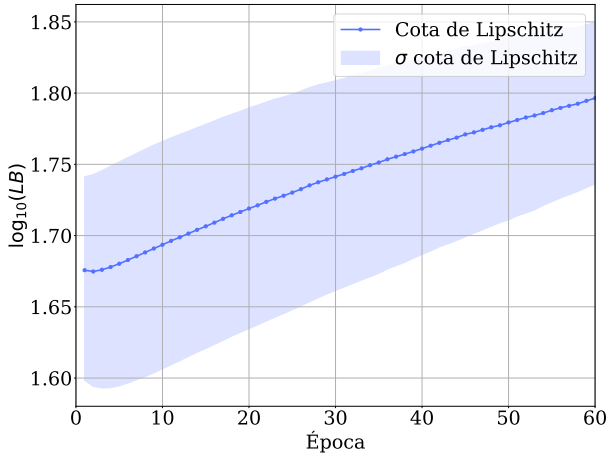
comportamiento de los otros métodos en los cuales se observa un crecimiento. Las franjas de desviación estándar son menos pronunciadas en la regularización LBA, en los otros métodos la desviación de la cota es considerable.



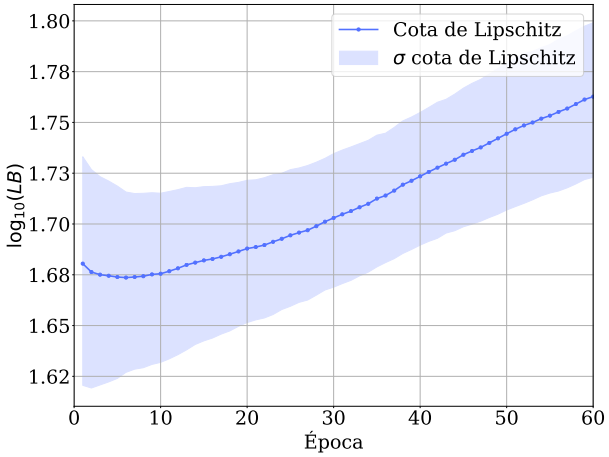
(a) Regularización LBA



(b) Regularización L2



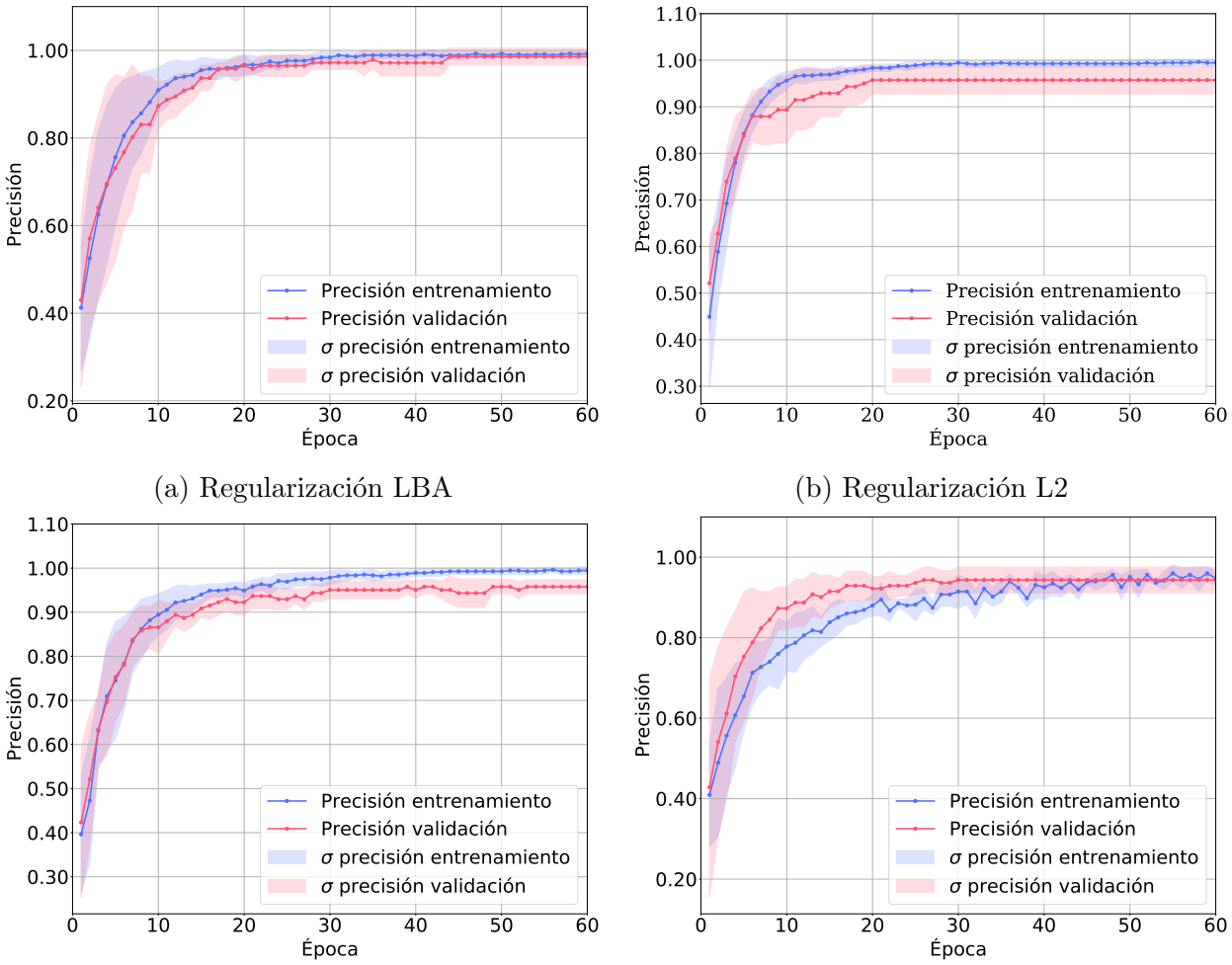
(c) Regularización L1



(d) Regularización *Dropout*

Figura 3-21.: $\log_{10}(LB)$ Cota de Lipschitz modelo neuronal con 3 capas residuales y conjunto de datos *Iris Dataset*. Experimento C.

La Figura **3-22** indica las curvas de precisión para el conjunto de datos *Wine Dataset*. En el entrenamiento realizado con el método de regularización LBA se puede notar una convergencia en un número de epocas menor, y un comportamiento similar entre los datos de entrenamiento y prueba. Los métodos de regularización L1 y L2 preservan una separación entre los resultados de los conjuntos de entrenamiento y prueba, además, muestran una variabilidad mayor. Por su parte en el método de regularización *Dropout* las curvas del conjunto de prueba supera a la del conjunto de entrenamiento, pero finalmente convergen.



(a) Regularización LBA

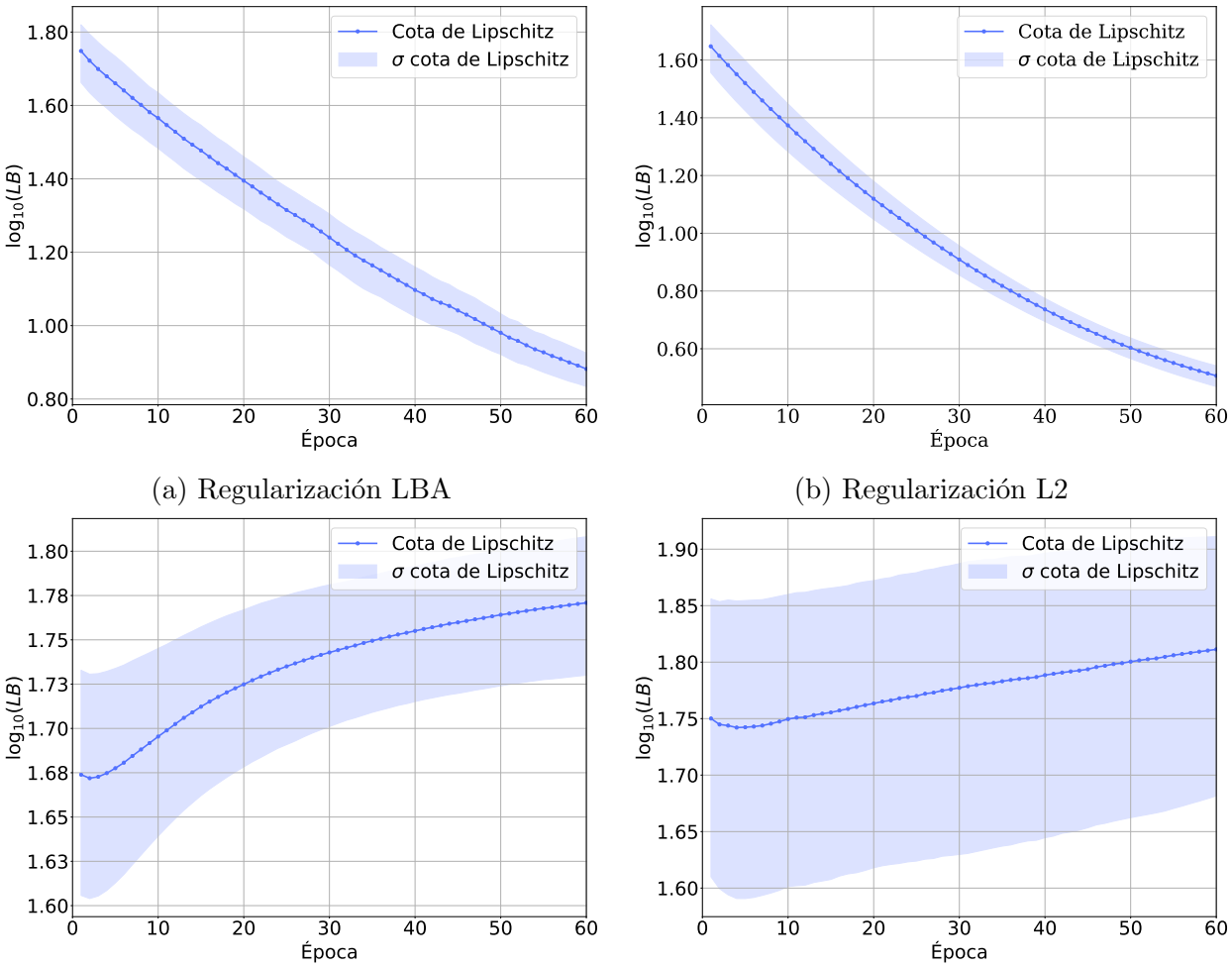
(b) Regularización L2

(c) Regularización L1

(d) Regularización *Dropout*

Figura 3-22.: Curvas de aprendizaje modelo neuronal 3 capas residuales y conjunto de datos *Wine Dataset*. Experimento C.

En las Figuras de la cota de Lipschitz **3-23** para el conjunto de datos *Wine Dataset* se observa un comportamiento decreciente en la regularización LBA y L2. Los métodos L1 y *Dropout* presentan la tendencia a incrementar el valor de la cota a través de las capas. Las franjas de desviación estándar son menos pronunciadas en el método L2, seguido por la regularización LBA. Un comportamiento contrario se presenta en L1 y *Dropout* ambos métodos indican alta variabilidad.



(a) Regularización LBA

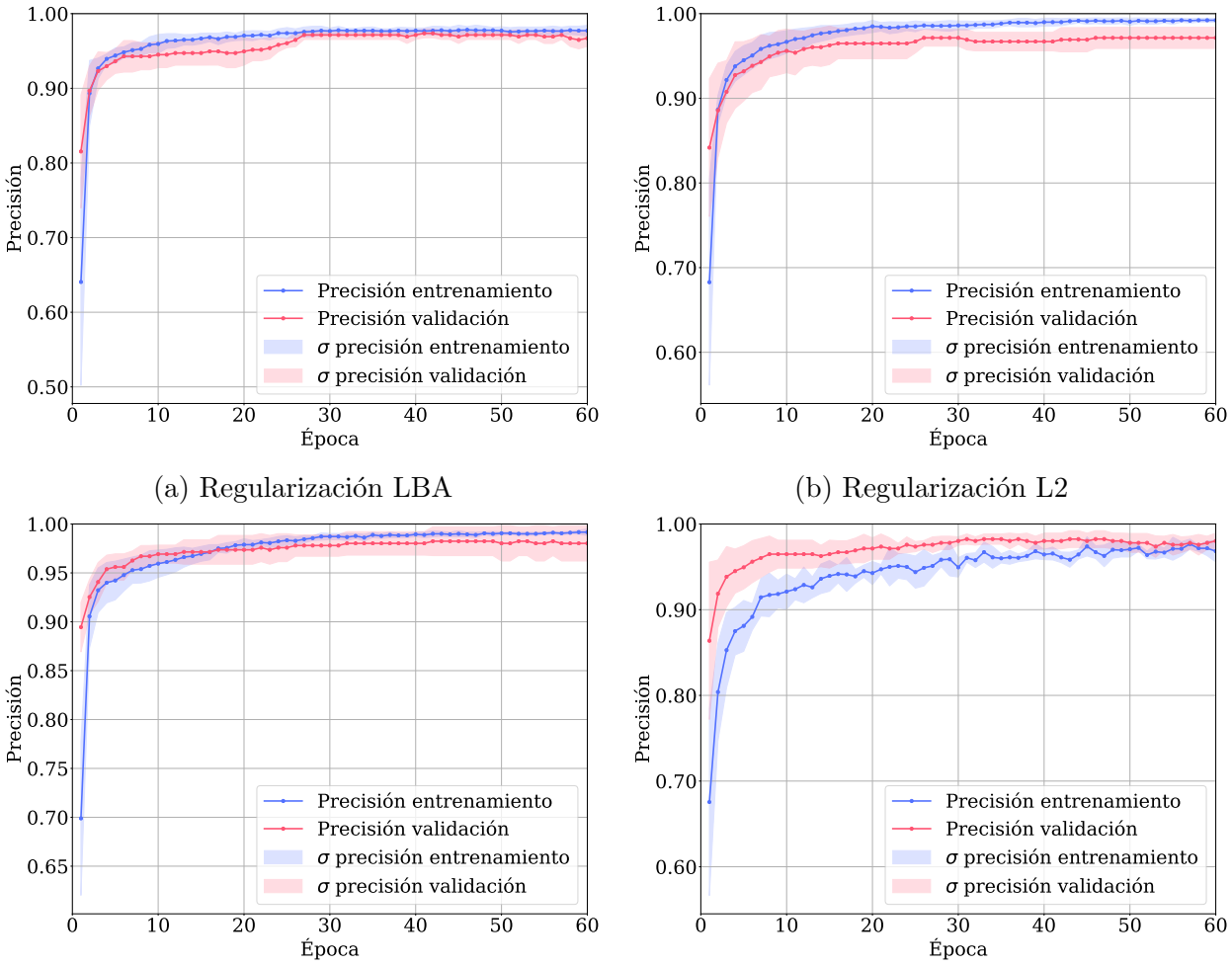
(b) Regularización L2

(c) Regularización L1

(d) Regularización *Dropout*

Figura 3-23.: $\log_{10}(LB)$ Cota de Lipschitz modelo neuronal con 3 capas residuales y conjunto de datos *Wine Dataset*. Experimento C.

Las curvas de precisión mostrada en la Figura **3-24** corresponden al desempeño de clasificación con el conjunto de datos *Breast Cancer Wisconsin*. En las figuras se destaca el método LBA por presentar un comportamiento muy similar para la precisión del entrenamiento y prueba, así como una variabilidad pequeña. Un comportamiento similar se observa para la regularización L1, logrando incluso mejor precisión final, sin embargo, en comparación con la regularización LBA presenta una separación entre entrenamiento y prueba considerable. En el método de regularización *Dropout* no se logra una precisión alta, además, la curva de prueba se sitúa por encima de la entrenamiento y ambas presentan una variación alta.



(a) Regularización LBA

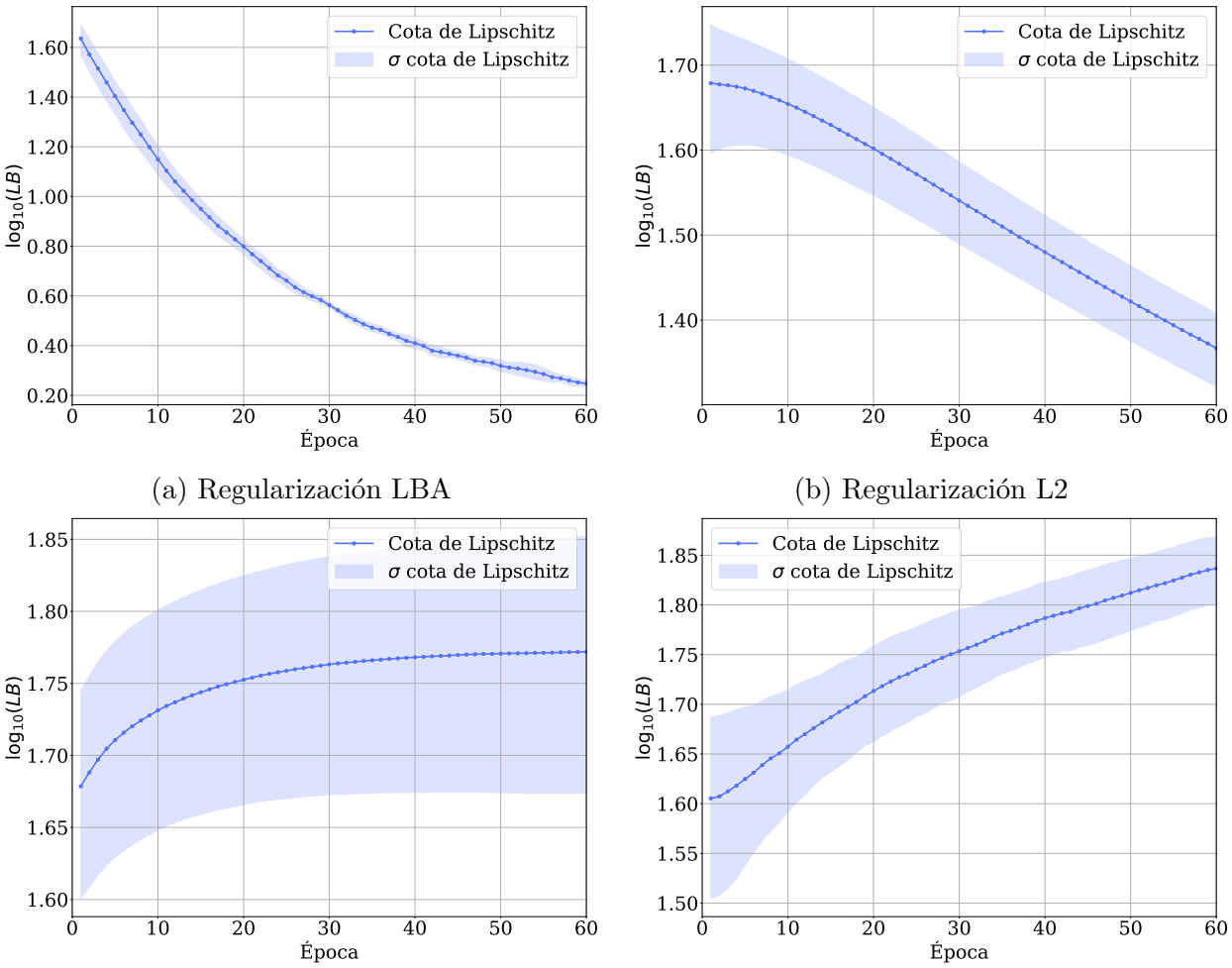
(b) Regularización L2

(c) Regularización L1

(d) Regularización *Dropout*

Figura 3-24.: Curvas de aprendizaje modelo neuronal 3 capas residuales y conjunto de datos *Breast Cancer Dataset*. Experimento C.

El comportamiento a través de las épocas para la cota de Lipschitz se ilustra en la Figura **3-25**. El método de regularización LBA muestra una disminución de este valor, con una variabilidad pequeña. Comportamiento similar se presenta para la regularización L2, sin embargo, presenta una desviación estándar mayor, como se indica en las franjas de color azul. Los método de regularización L1 y *Dropout* fomentan el incremento de la cota, ambos presentan franjas de desviación altamente pronunciadas.



(a) Regularización LBA

(b) Regularización L2

(c) Regularización L1

(d) Regularización *Dropout*

Figura 3-25.: $\log_{10}(LB)$ Cota de Lipschitz modelo neuronal con 3 capas residuales y conjunto de datos *Breast Cancer Dataset*. Experimento C.

La Tabla **3-3** indica los resultados de la evaluación de los modelos, para esto se empleó el 20 % de las observaciones del conjunto de datos. El método de regularización LBA se destaca por tener la mayor precisión en el conjunto de prueba, la menor diferencia entre entrenamiento y prueba y la menor desviación estándar en entrenamiento, esto para los tres *datasets*. En cuanto al tiempo de entrenamiento, la regularización LBA no difiere significativamente del modelo convencional (*Base line*). La regularización L1 y L2 alcanzan valores altos para la precisión promedio del conjunto de entrenamiento, sin embargo, presentan diferencias al compararse con la precisión del conjunto de prueba. En cuanto al método *Dropout* se evidencia que la precisión alcanzada en entrenamiento es menor que la lograda en prueba, además la desviación estándar para los primeros dos *dataset* es la más elevada.

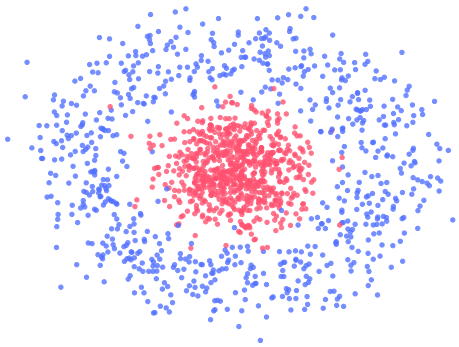
Tabla 3-3.: Resultados de la precisión en el entrenamiento y la prueba de los diferentes modelos de redes neuronales residuales para el Experimento C de la Subsección 3.2.1.

Dataset	**Métricas**	***Base Line***	**LBA**	**L2**	**L1**	***Dropout***
Flor Iris	% $Train\bar{A}_{cc}$	94.44	93.33	**95.78**	94.67	83.56
	% $Test\bar{A}_{cc}$	89.33	**92.67**	91.33	91.33	91.33
	% $\lvert Train\bar{A}_{cc} - Test\bar{A}_{cc}\rvert$	5.11	**0.66**	4.45	4.23	7.77
	$\pm$ $std(Train\bar{A}_{cc})$	2.534	2.629	1.474	1.912	6.222
	$\pm$ $std(Test\bar{A}_{cc})$	6.799	**3.887**	5.416	4.522	5.416
	Tiempo de entrenamiento [s]	**10**	11	14	14	11
Wine Dataset	% $Train\bar{A}_{cc}$	**99.64**	99.27	99.45	99.45	94.73
	% $Test\bar{A}_{cc}$	96.11	**98.89**	98.33	97.22	97.78
	% $\lvert Train\bar{A}_{cc} - Test\bar{A}_{cc}\rvert$	3.53	**0.38**	1.12	2.23	3.05
	$\pm$ $std(Train\bar{A}_{cc})$	**0.357**	0.455	0.727	0.727	1.763
	$\pm$ $std(Test\bar{A}_{cc})$	2.833	**1.361**	**1.361**	3.043	2.079
	Tiempo de entrenamiento [s]	**9**	14	14	15	17
Breast Cancer Wisconsin	% $Train\bar{A}_{cc}$	**99.33**	97.72	98.83	99.22	96.83
	% $Test\bar{A}_{cc}$	96.67	**97.37**	96.67	**97.37**	97.19
	% $\lvert Train\bar{A}_{cc} - Test\bar{A}_{cc}\rvert$	2.66	**0.35**	2.16	1.87	0.36
	$\pm$ $std(Train\bar{A}_{cc})$	**0.283**	0.643	0.656	0.304	1.133
	$\pm$ $std(Test\bar{A}_{cc})$	1.404	**0.961**	1.023	**0.961**	0.859
	Tiempo de entrenamiento [s]	**30**	33	26	41	35

Experimento D

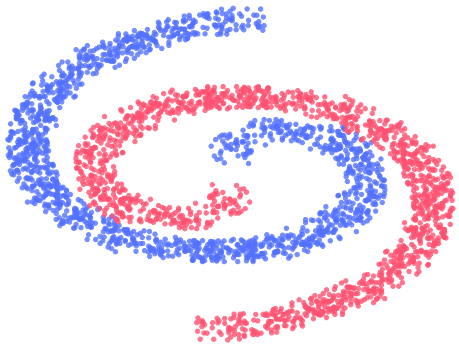
En este experimento se verificó el desempeño del método de regularización LBA, L1, L2 y Dropout con conjunto de datos sintéticos. Los datos sintéticos empleados son estructuras comúnmente usadas para verificar algoritmos de separación de clases. La arquitectura empleada es similar a la propuesta en Figura **2-7**, adaptando la capa de entrada para recibir dos componentes y utilizando la función sigmoide en la capa de salida. El número de capas empleado para cada *dataset* ser se explica a continuación.

El primer conjunto de datos corresponde a una clasificación binaria de 1500 observaciones diferenciando distribuciones de datos en círculos concéntricos, Figura **3-26**a, para este conjunto la arquitectura empleada fue de 5 capas residuales, con 30 épocas y lotes de 10 observaciones. El segundo *dataset* con el mismo número de observaciones corresponde a dos medios círculos intercalados Figura **3-26**b, se denominara el conjunto de datos medias lunas. Para este *dataset* se utilizaron 10 capas residuales, 30 épocas y lotes de 10 observaciones. Para el tercer conjunto de datos se generaron 3000 puntos que tiene una estructura de espirales concéntricos como se presentan en la Figura **3-26**c. Dada la complejidad de este *dataset* se consideró usar una arquitectura de 15 capas residuales, 60 épocas y lotes de 100 observaciones.



(a) Círculos concéntricos

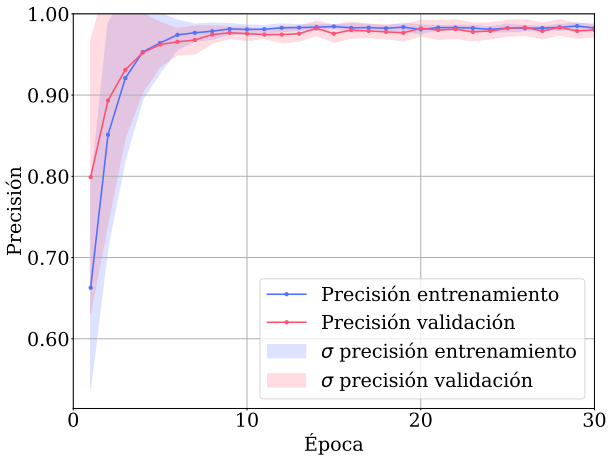
(b) Medias lunas



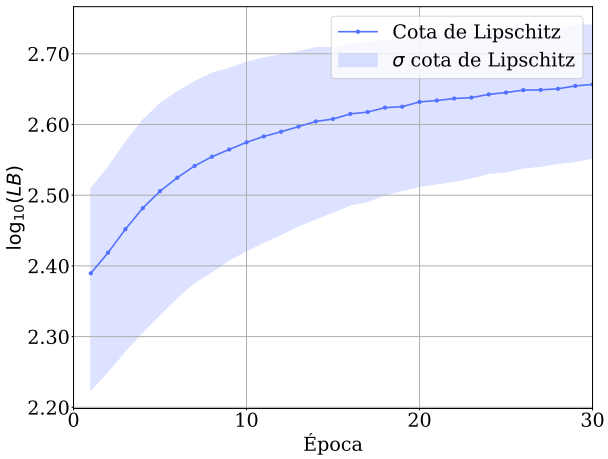
(c) Espirales concéntricos

Figura 3-26.: Conjunto de datos empleados en el Experimento D.

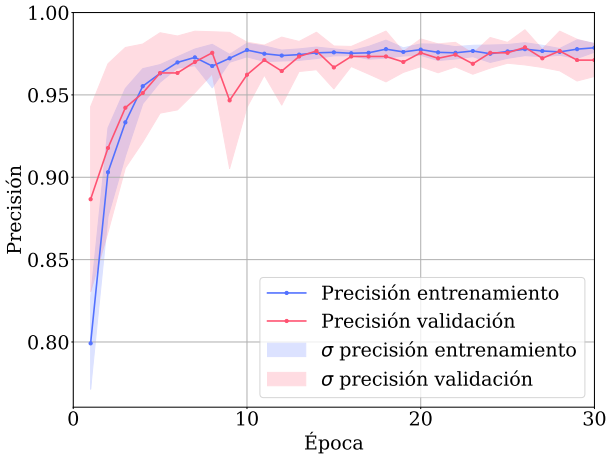
Para tener una referencia del comportamiento de un modelo convencional se realizó el entrenamiento para los 3 conjuntos de datos sin emplear ningún método de reducción de sobreajuste. La Figura **3-27** presenta los resultados para las curvas de precisión y de la evolución de la cota de Lipschitz para este entrenamiento convencional.



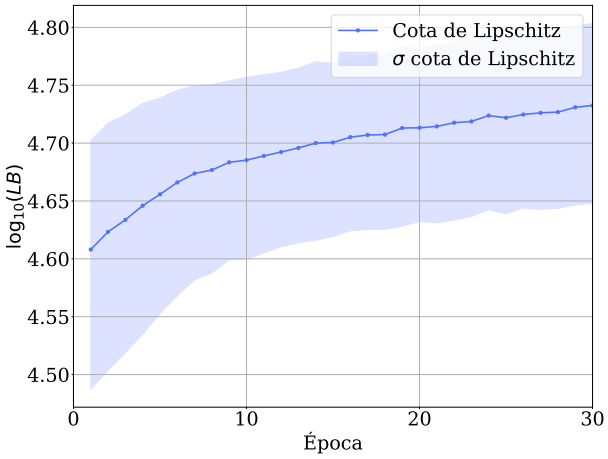
(a) Conjunto de datos círculos concéntricos



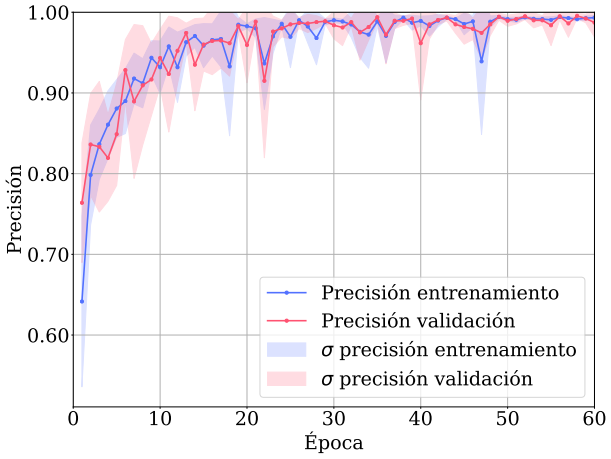
(b) Conjunto de datos círculos concéntricos



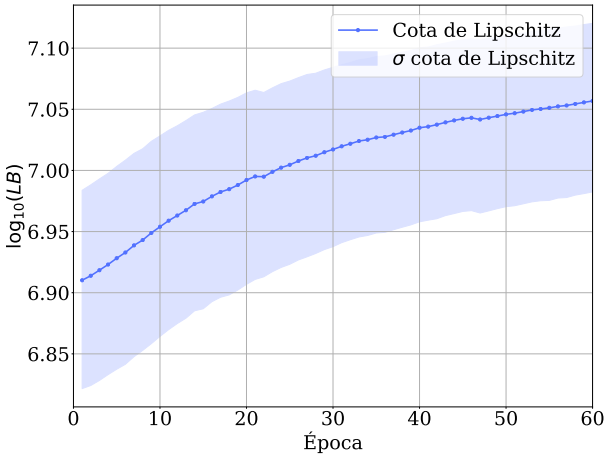
(c) Conjunto de datos medias lunas



(d) Conjunto de datos medias lunas



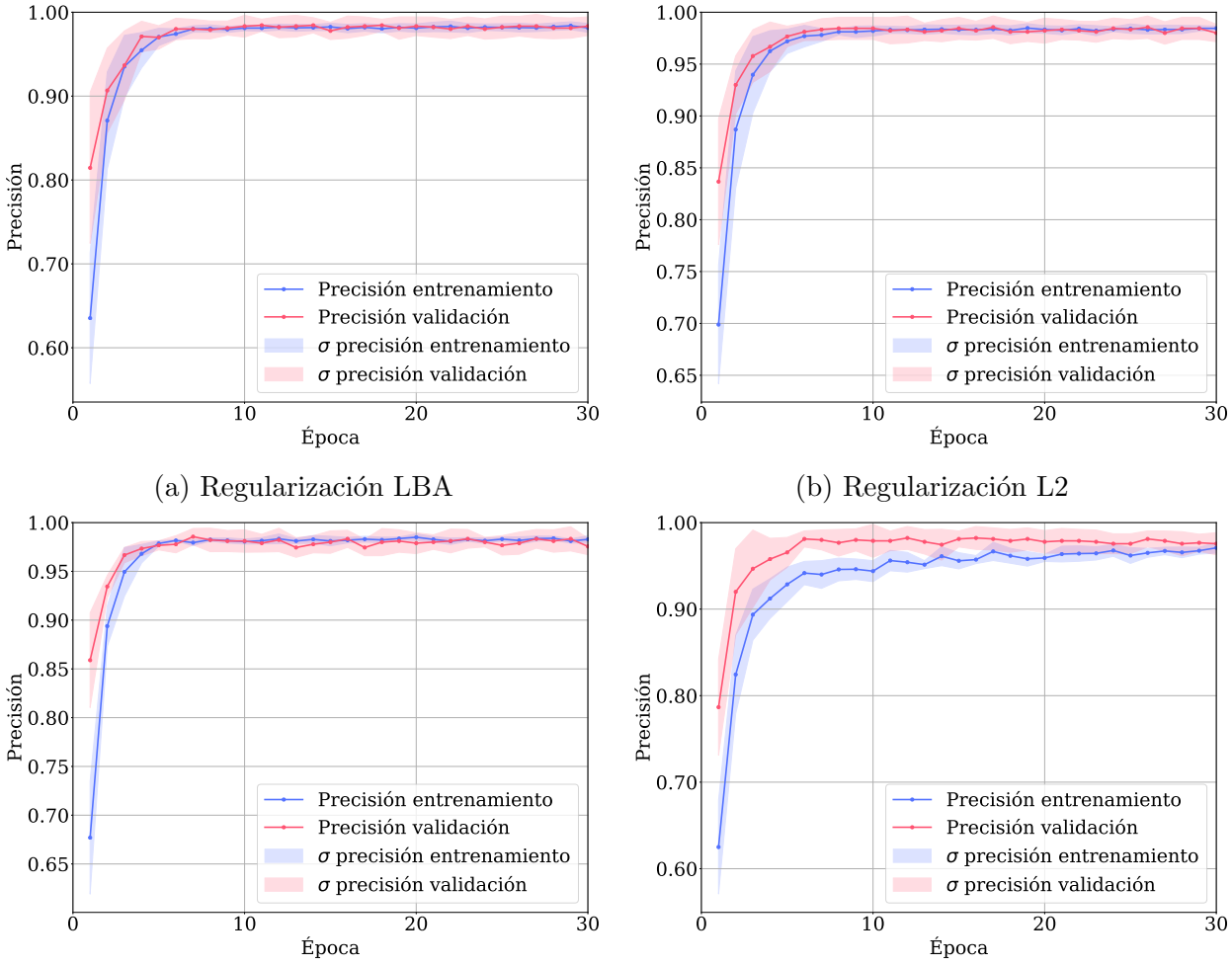
(e) Conjunto de datos espirales concéntricos



(f) Conjunto de datos espirales concéntricos

Figura 3-27.: Curvas de aprendizaje y $\log_{10}(LB)$ cota de Lipschitz del entrenamiento convencional. Experimento D.

La Figura **3-28** presenta las curvas de aprendizaje para la métrica precisión con el conjunto de datos de los círculos concéntricos. En las 30 épocas todos los métodos alcanzan un desempeño adecuado, en particular la regularización LBA. La regularización *Dropout* indica que la curva de prueba supera la de entrenamiento, y es el que menor precisión alcanza. Las franjas de al desviación estándar son similares para todos los métodos. La evolución de la cota de Lipschitz para este *dataset* de los círculos concéntricos se presenta en la Figura **3-29**. El método de regularización LBA indica un decrecimiento, contrario al método L1 y *Dropout*, en los cuales la tendencia es incremental. La regularización L2 también indica un decrecimiento, pero es más pronunciado en LBA. Las franjas que ilustran la desviación estándar son más pronunciadas en los métodos L2, L1 y *Dropout*.



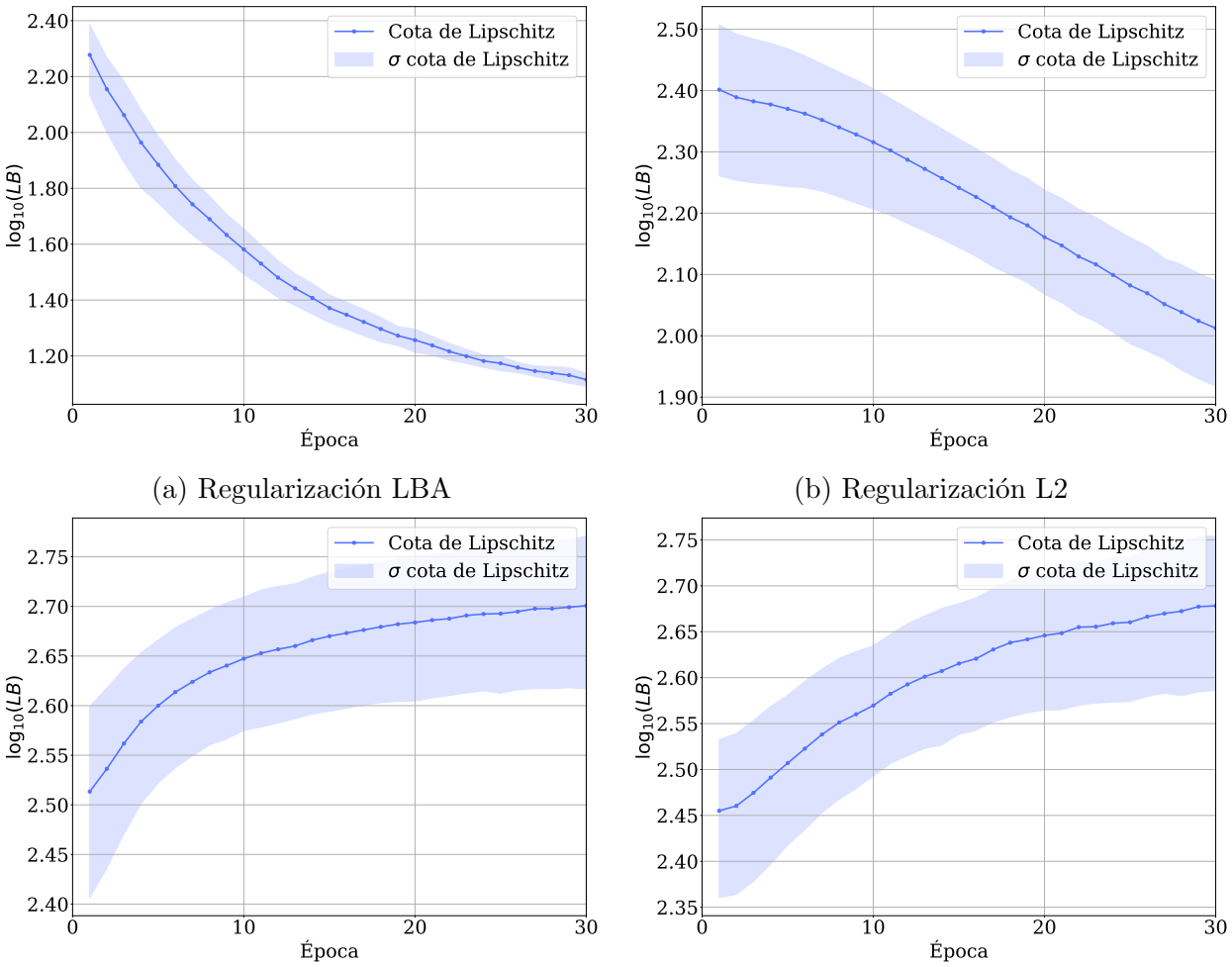
(a) Regularización LBA

(b) Regularización L2

(c) Regularización L1

(d) Regularización *Dropout*

Figura 3-28.: Curvas de aprendizaje modelo neuronal con 5 capas residuales y conjunto de datos círculos concéntricos conjunto de datos sintéticos círculos concéntricos. Experimento D.



(a) Regularización LBA

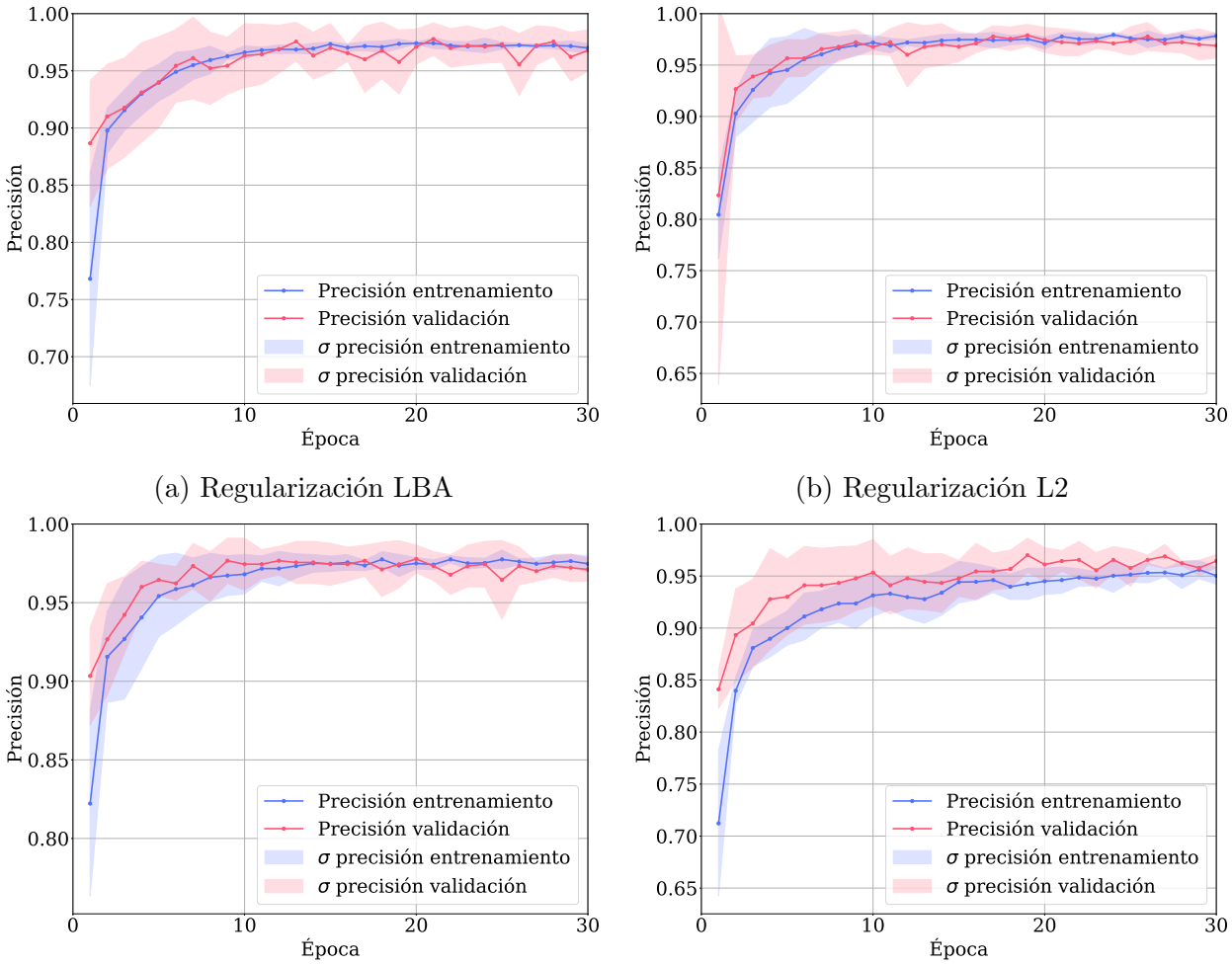
(b) Regularización L2

(c) Regularización L1

(d) Regularización *Dropout*

Figura 3-29.: $\log_{10}(LB)$ Cota de Lipschitz modelo neuronal con 5 capas residuales y conjunto de datos sintéticos círculos concéntricos. Experimento D.

La Figura **3-30** muestra las curvas de aprendizaje para el segundo *dataset*. Los métodos de entrenamiento tienen un desempeño similar. En la regularización *Dropout* la precisión es menor y se mantiene la tendencia a tener una curva de validación por encima de la lograda en entrenamiento. La desviación estándar es un poco mayor en el método de regularización LBA y se presentan algunos picos en la regularización L2.



(a) Regularización LBA

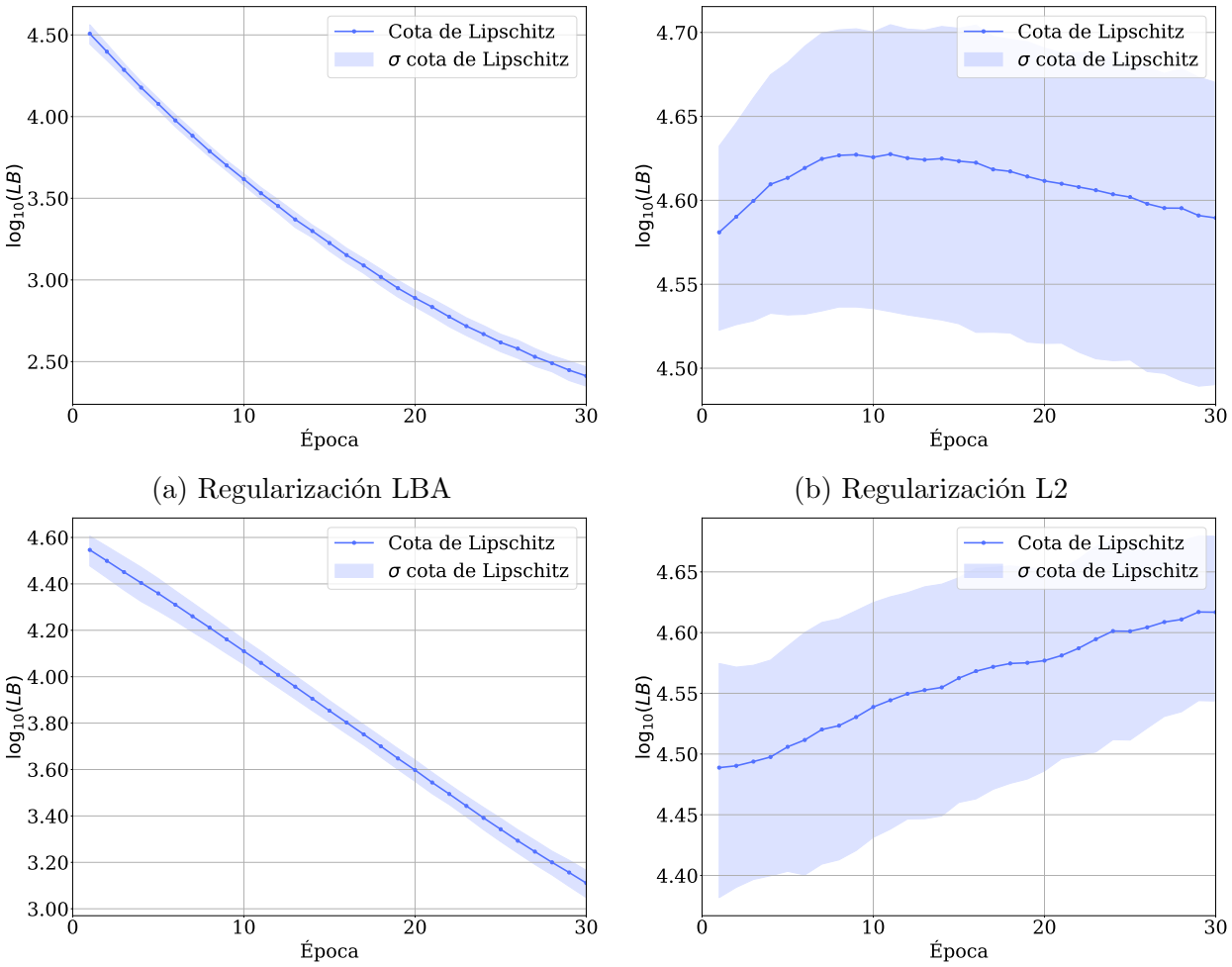
(b) Regularización L2

(c) Regularización L1

(d) Regularización *Dropout*

Figura 3-30.: Curvas de aprendizaje modelo neuronal 10 capas residuales y conjunto de datos sintéticos medias lunas. Experimento D.

En la Figura **3-31** se presenta la evolución de la cota de Lipschitz para el segundo *data-set*. Pese a que las curvas de aprendizaje no presentaban diferenciar notables, las cotas de Lipschitz si. En el método de regularización LBA la cota disminuye con el tiempo, similar a la regularización L1, en ambos casos la franja que ilustra la desviación estándar es reducida. Los métodos L2 y *Dropout* tienen alta variabilidad, en el primero la cota crece y luego disminuye, en *Dropout* se observa un incremento.



(a) Regularización LBA

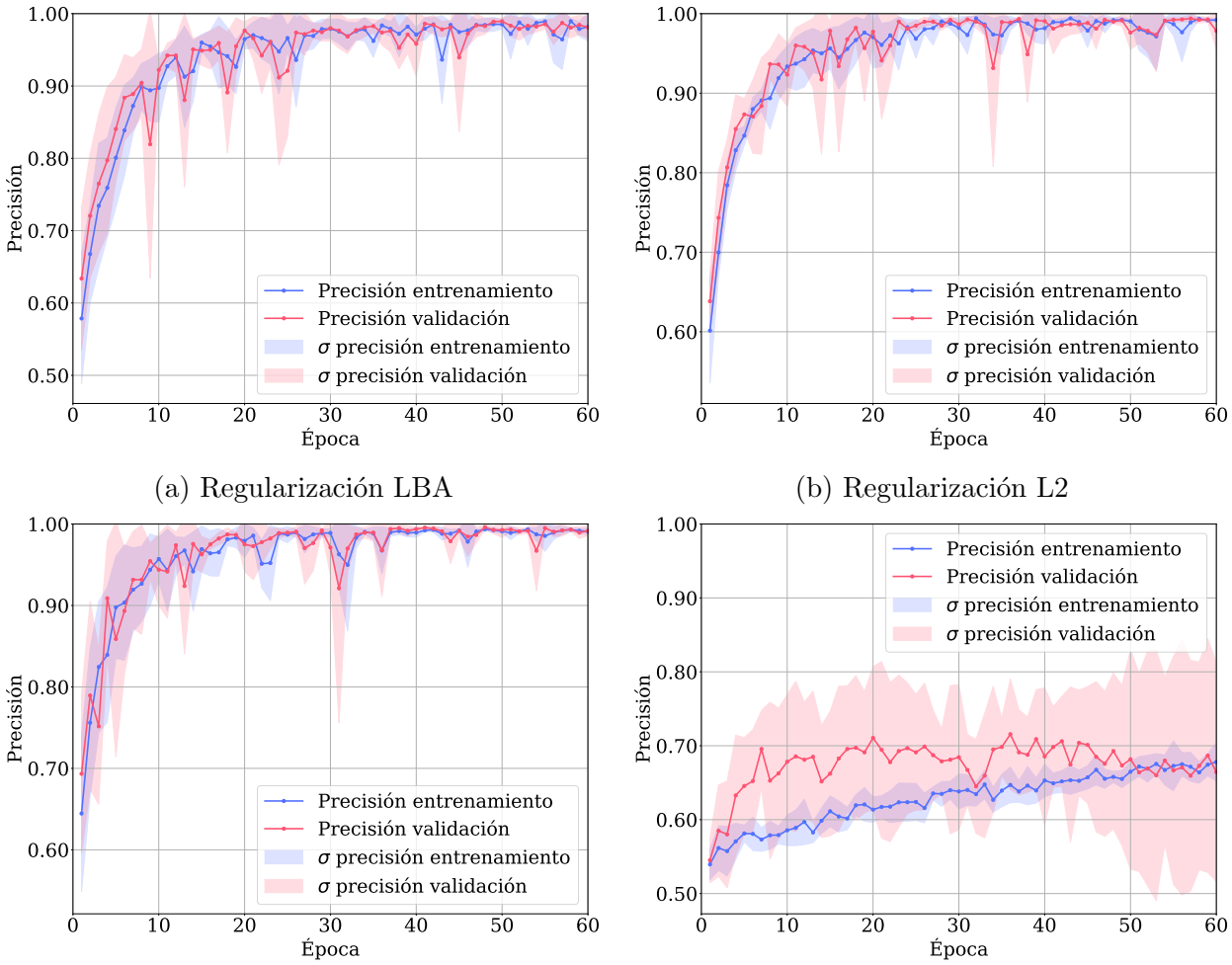
(b) Regularización L2

(c) Regularización L1

(d) Regularización *Dropout*

Figura 3-31.: $\log_{10}(LB)$ Cota de Lipschitz modelo neuronal con 10 capas residuales y conjunto de datos sintéticos medias lunas. Experimento D.

El conjunto de datos de espirales concéntricos exhibe una mayor complejidad. Durante las 60 épocas de entrenamiento se pueden observar en la Figura **3-32** curvas de aprendizaje ruidosas de un comportamiento similar para los métodos regularización LBA, L2 y L1. El método de regularización *Dropout* no logra un buen desempeño manteniendo la precisión por debajo del % 80, además también es el método con franjas de variabilidad muy marcadas.



(a) Regularización LBA

(b) Regularización L2

(c) Regularización L1

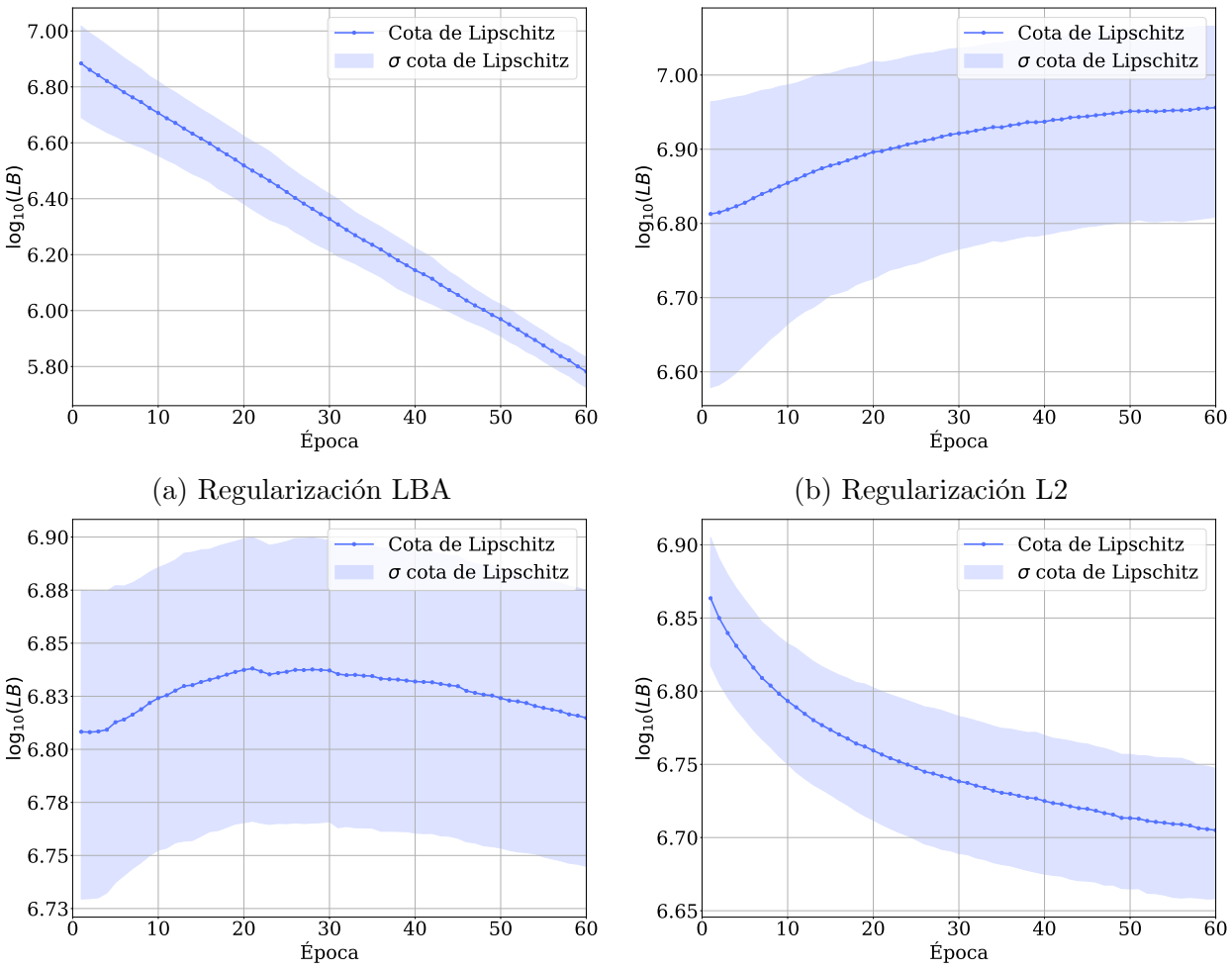
(d) Regularización *Dropout*

Figura 3-32.: Curvas de aprendizaje modelo neuronal 15 capas residuales y conjunto de datos sintéticos espirales concéntricos. Experimento D.

La cota de Lipschitz para los espirales concéntricos se muestra en Figura **3-33**. El método L2 muestra un comportamiento incremental, en la regularización L1 se observa un crecimiento y posterior decrecimiento de la cota. En el método de *Dropout* la constante decrece, igual a como ocurre con LBA, sin embargo en este ultimo la variación es menor.

El proceso de validación para los 3 conjuntos de datos se realizó con el 20 % de las observaciones. Para los círculos concéntricos el método LBA alcanza el mejor resultado para la precisión con el conjunto de prueba. Los resultados para el segundo *dataset* indican nuevamente que la mayor precisión en test se logra con el método LBA, también es el método que presenta menor diferencia entre la precisión de entrenamiento y prueba. En los espirales concéntricos el método L2 es el que mejor se desempeña logrando la mayor precisión en

entrenamiento y prueba, y la menor diferencia entre estos dos conjuntos. El método de regularización LBA obtiene unos resultados bastantes similares pero inferiores a L2 y L1. Otro aspecto a considerar es que el método LBA requiere mayor tiempo para la ejecución, pero este valor no dista considerablemente de los demás métodos. Estos resultados se sintetizan en la Tabla **3-4**.



(a) Regularización LBA

(b) Regularización L2

(c) Regularización L1

(d) Regularización *Dropout*

Figura 3-33.: $\log_{10}(LB)$ Cota de Lipschitz modelo neuronal con 15 capas residuales y conjunto de datos sintéticos espirales concéntricos. Experimento D.

Tabla 3-4.: Resultados de la precisión en el entrenamiento y la prueba de los diferentes modelos de redes neuronales residuales para el Experimento D de la Subsección 3.2.1.

Dataset	**Métricas**	***Base Line***	**LBA**	**L2**	**L1**	***Dropout***
Círculos concéntricos (Modelo 5 capa residuales)	% $Train\bar{A}_{cc}$	98.25	98.14	**98.44**	98.28	97.08
	% $Test\bar{A}_{cc}$	97.20	**97.40**	97.23	97.13	96.73
	% $\lvert Train\bar{A}_{cc} - Test\bar{A}_{cc}\rvert$	1.05	0.74	1.21	1.15	**0.35**
	$\pm$ $std(Train\bar{A}_{cc})$	0.324	0.470	**0.184**	0.336	0.541
	$\pm$ $std(Test\bar{A}_{cc})$	**0.245**	0.271	0.309	0.618	0.602
	Tiempo de entrenamiento [s]	30	45	38	37	31
Medias lunas (Modelo 10 capas residuales)	% $Train\bar{A}_{cc}$	**97.83**	97.00	**97.83**	97.47	95.03
	% $Test\bar{A}_{cc}$	95.67	**96.10**	95.63	95.50	96.00
	% $\lvert Train\bar{A}_{cc} - Test\bar{A}_{cc}\rvert$	2.16	**0.90**	2.20	1.97	0.97
	$\pm$ $std(Train\bar{A}_{cc})$	**0.242**	0.336	0.286	0.468	0.726
	$\pm$ $std(Test\bar{A}_{cc})$	0.435	**0.389**	0.542	0.650	0.408
	Tiempo de entrenamiento [s]	**49**	69	60	61	51
Espirales concéntricos (Modelo 15 capas residuales)	% $Train\bar{A}_{cc}$	99.33	98.19	**99.83**	99.10	67.80
	% $Test\bar{A}_{cc}$	**98.83**	98.80	99.80	99.53	66.40
	% $\lvert Train\bar{A}_{cc} - Test\bar{A}_{cc}\rvert$	0.50	0.61	**0.03**	0.43	1.40
	$\pm$ $std(Train\bar{A}_{cc})$	**0.205**	1.742	0.242	0.334	2.290
	$\pm$ $std(Test\bar{A}_{cc})$	1.304	0.620	**0.155**	0.239	13.332
	Tiempo de entrenamiento [s]	**26**	39	34	34	31

3.2.2. Validación con imágenes adversarias

Una imagen adversaria[4] es el resultado de modificar sutilmente una imagen original, sin embargo, pese a que la modificación es imperceptible para el ojo humano, estas imágenes tienden a "confundir" al algoritmo de clasificación y en consecuencia afectan su rendimiento [118, 119]. Las imágenes adversarias se utilizan para evaluar la robustez de los algoritmo de aprendizaje automático y analizar su desempeño ante pequeñas variaciones en los datos de entrada. Por lo anterior, las imágenes adversarias con adecuadas para comprobar la capacidad de generalización de algoritmos de clasificación.

[4]También se podrían considerar datos adversarios que representen necesariamente una imagen, pero en general se ha abordado la idea de imágenes adversarias por la facilidad de identificación y prueba del ojo humano.

Matemáticamente, consideremos una imagen x de algún problema de clasificación etiquetada con y y la imagen adversaria $x^{(ad)}$, resultado de la corrupción de x. Entonces, para que $x^{(ad)}$ sea un cambio sutil de x de manera que $x \approx x^{(ad)}$ se espera que $||x^{(ad)} - x|| < \epsilon$, con $\epsilon > 0$ un valor arbitrariamente pequeño. Además, la imagen adversaria debe ser capaz de perturbar la respuesta del clasificador, entonces

$$\max_{x^{(ad)}} l(f(x^{(ad)}; \theta), y).$$

Asumiendo las dos condiciones impuestas a $x^{(ad)}$, el cambio sutil y la perturbación del clasificador entonces $x^{(ad)}$ se puede ver como

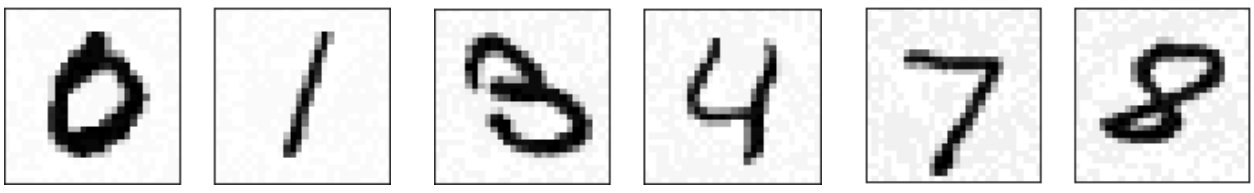
$$\min_{x^{(ad)}} - l(f(x^{(ad)}; \theta), y) + \gamma||x^{(ad)} - x||,$$

el factor γ pondera el factor relacionado con la distorsión. Este problema típicamente se puede resolver con las condiciones de optimalidad, en particular tomando $\nabla(-l(f(x^{(ad)}; \theta), y) + \gamma||x^{(ad)} - x||) = \mathbf{0}$. Siguiendo estas hipótesis una forma de resolver el problema y encontrar la imagen $x^{(ad)}$, corresponde a

$$x^{(ad)} = x + \Gamma \text{sign}(\nabla_x(l(f(x; \theta), y)),$$

el la expresión Γ se relaciona con el factor de distorcion que sufrirá la imagen original, por su parte sign es la función que regresa el signo del gradiente, en este caso, de la función de costo respecto a x $\nabla_x(l(f(x; \theta), y)$. Esta forma de generar imágenes adversarias se conoce como FGSM: Fast Gradient Sing Method [118], en esta sección se emplea este método con diferentes niveles de perturbación Γ para generar imágenes adversarias de los conjuntos de datos MNIST y Fashion MNIST. Estas imágenes se evalúan en modelos neuronales entrenados con los métodos de regularización LBA, L1, L2 y *Dropout*.

Experimento A



(a) $\Gamma = 0{,}01$ (b) $\Gamma = 0{,}02$ (c) $\Gamma = 0{,}03$

Figura 3-34.: Imágenes adversarias del conjunto *Digits MNIST* con diferentes niveles de perturbación Γ. Fuente: elaborado por el autor y a partir de [112].

Usando la arquitectura presentada en la Figura **2-7** para modelos de 4, 6 y 8 capas residuales se realizó el entrenamiento con el conjunto de datos de digitos MNIST. Los hiper-parámetros se configurarón igual que el Experimento A de la Subsección 3.2.1. Finalizado el entrenamiento para cada modelo se generaron 1000 imágenes adversarias con factores de distorsión Γ de 0.01, 0.02 y 0.03, un ejemplo de como se observan estas imágenes se presenta en la Figura **3-34**. Con estos conjuntos se evaluó el clasificador linea base (*Base Line*), y los entrenados con los métodos LBA, L2, L1 y *Dropout*.

Tabla 3-5.: Resultados de la precisión al evaluar diferentes arquitecturas neuronales residuales con imágenes adversarias de *Digits MNIST*. Experimento A Subsección 3.2.2.

# de capas residuales	***Dataset* de validación**	***Base Line***	**LBA**	**L2**	**L1**	***Dropout***
4	*TrainSet*	96.42	90.22	96.84	**97.30**	88.86
	TestSet	92.42	89.21	**92.99**	92.68	90.80
	Imágenes adversarias $\Gamma = 0{,}01$	74.80	**83.40**	78.10	73.90	76.10
	Imágenes adversarias $\Gamma = 0{,}02$	46.90	**77.70**	50.90	46.90	56.80
	Imágenes adversarias $\Gamma = 0{,}03$	24.40	**70.20**	23.90	21.50	41.40
6	*TrainSet*	**97.80**	93.04	97.21	97.16	90.83
	TestSet	91.93	90.95	91.70	**92.62**	91.73
	Imágenes adversarias $\Gamma = 0{,}01$	70.40	**84.00**	73.40	81.90	77.50
	Imágenes adversarias $\Gamma = 0{,}02$	44.20	**77.50**	50.00	66.10	60.00
	Imágenes adversarias $\Gamma = 0{,}03$	23.40	**69.90**	29.30	45.10	44.40
8	*TrainSet*	**97.10**	94.99	96.22	95.81	88.88
	TestSet	91.20	92.12	**92.39**	91.73	90.68
	Imágenes adversarias $\Gamma = 0{,}01$	64.70	**84.12**	82.10	80.00	68.20
	Imágenes adversarias $\Gamma = 0{,}02$	34.30	**75.00**	64.50	61.10	45.90
	Imágenes adversarias $\Gamma = 0{,}03$	18.20	**63.70**	45.20	39.00	29.40

La Tabla **3-5** presenta los resultados de la evaluación usando la métrica de precisión. En los resultados se puede observar que independientemente de la arquitectura empleada el método de regularización LBA logra la mayor precisión al ser evaluado con el conjunto de imágenes adversarias. La diferencia de este método respecto a los demás es considerable, tolerando la contaminación presente en las imágenes incluso para el factor $\Gamma = 0{,}03$.

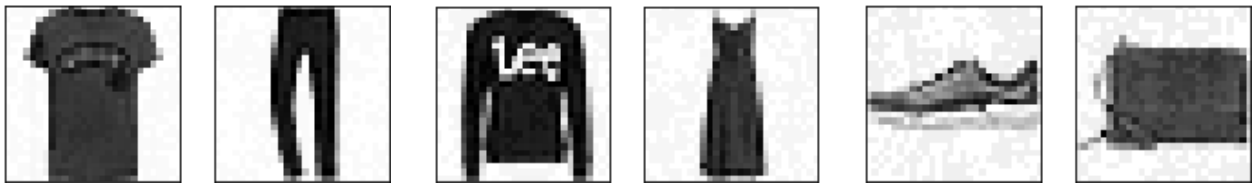
Experimento B

Similar al experimento anterior, en esta sección se presentan los resultados de la validación con conjuntos de imágenes adversarias contaminadas con perturbaciones Γ de 0.01, 0.02 y 0.03. En la Figura **3-35** se detallan algunas imágenes adversarias generadas. Para este experimento se empleó el conjunto de datos *Fashion* MNIST y se fijaron los hiper-parámetros de forma similar al Experimento B de la Subsección 3.2.1. Las arquitecturas de este experimento se construyeron siguiendo la Figura **2-7** con 3, 5 y 7 capas residuales.

En la Tabla **3-6** se exponen los resultados de la precisión para los entrenamientos con los métodos LBA, L2, L1 y *Dropout*, además de incluir el modelo convencional *Base Line*. Los resultados indican que en la mayoría de los casos el método de regularización LBA presenta mejor desempeño con las imágenes adversarias que los demás métodos.

Tabla 3-6.: Resultados de la precisión al evaluar diferentes arquitecturas neuronales residuales con imágenes adversarias de *Fashion MNIST*. Experimento B Subsección 3.2.2.

# de capas residuales	***Dataset* de validación**	***Base Line***	**LBA**	**L2**	**L1**	***Dropout***
3	*TrainSet*	88.48	79.13	87.88	**88.70**	81.41
	TestSet	83.12	77.68	83.48	**83.61**	83.06
	Imágenes adversarias $\Gamma = 0{,}01$	63.60	**73.70**	69.10	68.30	71.70
	Imágenes adversarias $\Gamma = 0{,}02$	40.50	**67.30**	49.00	46.60	58.10
	Imágenes adversarias $\Gamma = 0{,}03$	19.80	**61.30**	29.80	26.30	45.80
5	*TrainSet*	87.49	81.46	87.52	**86.74**	80.02
	TestSet	79.92	80.28	82.84	**83.50**	80.06
	Imágenes adversarias $\Gamma = 0{,}01$	60.70	74.30	66.80	**74.40**	66.20
	Imágenes adversarias $\Gamma = 0{,}02$	32.70	**66.80**	43.10	60.10	52.40
	Imágenes adversarias $\Gamma = 0{,}03$	16.20	**54.60**	26.60	46.90	42.30
7	*TrainSet*	89.76	83.81	**88.85**	84.94	77.31
	TestSet	82.82	80.28	**83.29**	80.67	78.19
	Imágenes adversarias $\Gamma = 0{,}01$	63.30	**75.30**	65.20	61.20	60.30
	Imágenes adversarias $\Gamma = 0{,}02$	38.70	**65.20**	42.20	38.40	46.80
	Imágenes adversarias $\Gamma = 0{,}03$	23.50	**58.30**	25.30	25.90	36.70



(a) $\Gamma = 0{,}01$ (b) $\Gamma = 0{,}02$ (c) $\Gamma = 0{,}03$

Figura 3-35.: Imágenes adversarias del conjunto *Fashion MNIST* con diferentes niveles de perturbación Γ. Fuente: elaborado por el autor y a partir de [113].

3.3. Discusión de resultados

3.3.1. Discusión del análisis teórico

El análisis teórico realizado como parte de la propuesta del método de regularización LBA motiva los siguientes tópicos de discusión:

- Como se demostró en la Subsección 3.1.2 las redes neuronales que incluyen capas completamente conectadas preservan la propiedad de continuidad de Lipschitz si sus funciones de activación lo son. Este análisis también se extiende a versiones residuales convencionales con una sola capa completamente conectada dentro del bloque residual. Dependiendo de las funciones de activación la cota de Lipschitz de una red neural depende del producto de la norma espectral de las matrices de pesos de las conexiones.

- Según lo expuesto en la Sección 3.1.2 las perturbaciones presentes en la entrada de la red se propaga a la salida (*forward*) de forma proporcional a la cota de Lipschitz de la red.

- En la Sección 3.1.2 se estableció que para las arquitecturas de Perceptrón y red neuronal residual de una capa oculta la magnitud de las componentes del gradiente de la función de pérdida está acotada por la cota de Lipschitz. Si bien el análisis fue realizado para estas arquitecturas se puede generalizar a redes con más capas.

- Los dos puntos de análisis previamente mencionados muestran una relación directa entre la cota de Lipschitz y el proceso de aprendizaje. Como se explica en Subsección 2.1.1, este algoritmo se utiliza para calcular el gradiente de la función de pérdida con respecto a los parámetros de la red neuronal. En su funcionamiento, se propaga la salida de la red hacia adelante y se calculan las derivadas parciales de los pesos de las conexiones al propagarse hacia atrás. Por la discusión previa se puede decir que la cota de Lipschitz limita la propagación de ruido en el proceso, tanto hacia adelante como hacia atrás. En otras palabras, establece un límite máximo para la magnitud de los

cambios que pueden ocurrir en la salida de la red debido a perturbaciones en la entrada y limita la propagación de ruido en el proceso del cálculo de las derivadas parciales.

- La equivalencia de la norma espectral con la norma de Frobenius permite asumir la regularización utilizando esta ultima para acotar la norma espectral, siempre y cuando se considere un factor de escalamiento apropiado (Sección 3.1.2). Por otra parte, el usar la norma de Frobenious como parte de la función a optimizar también representa una ventaja computacional, ya que su minimización requiere menos operaciones.

3.3.2. Discusión de los experimentos

Los experimentos realizados permiten establecer algunos puntos de discusión:

- En los experimentos A y B, independientemente del método de entrenamiento empleado (regularización LBA, L1, L2 o *Dropout*), al comparar las curvas de aprendizaje con el modelo que carece de una estrategia de reducción del sobreajuste, presentado en la Subsección 2.3.2, se evidencian mejoras en la capacidad de generalización. Lo anterior se puede analizar al observar la distancia entre las curvas de aprendizaje con los datos de entrenamiento y los de validación.

- En los experimentos A y B, el método de regularización LBA presentó un desempeño adecuado, destacándose entre todos los métodos como el que mantiene la menor diferencia entre las curvas de aprendizaje realizadas con los datos de entrenamiento y prueba. Esto implica que los resultados logrados durante el entrenamiento fueron replicables al conjunto de validación, es decir, se mejoró la capacidad de generalización.

- En el experimento B también se observa, en las curvas de aprendizaje, una menor variabilidad lograda al aplicar el método LBA, este resultado es adecuado ya que indica una robustez al cambio de conjunto de entrenamiento, esto se relaciona con la estimación de parámetros que pueden generalizar mejor a la población. En el experimento A también se logran variaciones pequeñas en la regularización LBA, en especial en la zona de convergencia. No obstante, en las primeras épocas el método tiende a variar. Pese a que esta condición no es deseable el algoritmo se recupera adecuadamente reduciendo su variabilidad y convergiendo de forma estable.

- En las tablas de evaluación de resultados de los experimentos A y B se puede notar que el algoritmo de *Dropout* y regularización LBA son los que menor diferencias presentan entre los valores de precisión de entrenamiento y prueba. Sin embargo, *Dropout* tiende a tener menores valores de precisión con los datos de prueba. Además, en el experimento B la regularización LBA se destaca en las arquitecturas de 5 y 7 capas por tener la menor desviación estándar en la evaluación con los conjuntos de entrenamiento y prueba.

- En el experimento C, realizado con conjuntos de datos de aplicaciones reales, se observa para el primer conjunto de datos un comportamiento similar de las curvas de aprendizaje de los métodos empleados. Sin embargo, en el método de regularización LBA el área de las franjas que indican la desviación estándar de la precisión a través de los pliegues de la validación cruzada es significativamente menor. En el segundo conjunto de datos el método de regularización LBA resuelve adecuadamente la tarea de clasificación, logrando una convergencia tanto en entrenamiento como prueba muy cercana al 100 %. En el tercer conjunto de datos, las curvas de aprendizaje indican una diferencia menor entre entrenamiento y prueba para la regularización LBA.

- Los resultados de validación del experimento C indican que para todos los conjuntos de datos de este experimento la mejor precisión, al evaluar el modelo con los datos de prueba, se logra en el método de regularización LBA. Además, este método es el que menor diferencia presenta entre la precisión de entrenamiento y prueba, y también indica la menor desviación estándar de la precisión en el conjunto de prueba.

- El experimento D se realizó con el propósito de evaluar el método de regularización LBA con datos sintéticos que generen geometrías espaciales no separables linealmente. En todos los casos la regularización LBA logró resolver el problema de clasificación, de acuerdo con estos resultados se puede inferir que el método es robusto a las condiciones de distribución geométricas, círculos concéntricos, medios círculos intercalados y espirales concéntricos. En el caso de esta ultima estructura geométrica, el método de *Dropout* no fue capaz de resolver el problema.

- En los experimentos de la Subsección 3.2.1 se puede apreciar que en el método *Dropout* las curvas de precisión con el conjunto de validación quedan por encima de las de entrenamiento. Este fenómeno se puede explicar debido a que durante el entrenamiento el método desconecta aleatoriamente neuronas y pierde tanto su complejidad como la capacidad de aprendizaje, por el contrario, en el momento de la validación se emplean todas las conexiones y se logra una estructura que evidencia resultados superiores.

- En los experimentos de la Subsección 3.2.1 las gráficas que presentan la evolución de la cota de Lipschitz señalan que al emplear el método de regularización LBA el valor de la cota decae. Este es el único método que mantiene durante todos los experimentos la tendencia a disminuir de la cota de Lipschitz. De acuerdo con esto se puede afirmar que para los experimentos realizados se cumple el propósito planteado en el análisis teórico, reducir la cota de Lipschitz. En algunas ocasiones la regularización L1 y L2 también lo hacen, sin embargo, sus comportamientos son impredecibles ya que en algunos escenarios tienen tendencias opuestas. También se considera relevante resaltar que, en comparación con los demás métodos, la desviación estándar de la cota de Lipschitz es menor en el método de regularización LBA.

- En la Subsección 3.2.2 los resultados logrados en la validación con imágenes adversarias indican que la regularización LBA logró valores de precisión adecuados para los tres niveles Γ de perturbación. En ambos experimentos se puede destacar la capacidad del método el cual supera, en el Experimento A, en hasta 46 puntos porcentuales la precisión lograda en el entrenamiento convencional y en el Experimento B, hasta un 41 %, esto para el nivel más alto de perturbación $\Gamma = 0{,}03$. En estos experimentos se puede inferir que el método de regularización LBA es adecuado para tolerar ataques con imágenes adversarias.

- En general, el método de regularización LBA requirió mas tiempo de ejecución para realizar el mismo número de épocas que los métodos convencionales. El tiempo requerido en el método se ve afectado por el tamaño de las matrices de pesos de la red, al incrementar el número de neuronas el tamaño de las matrices también lo hace y resulta computacionalmente costoso calcular la norma espectral de estas matrices.

4. Conclusiones y trabajo futuro

4.1. Conclusiones

En el presente trabajo se realizó un estudio del problema de sobreajuste, y cumpliendo el objetivo general, se determinó un método para reducir esta problemática en arquitecturas de redes neuronales residuales (ResNet) en un escenario de clasificación de patrones. De acuerdo con la metodología empleada, enfocada en determinar las propiedades matemáticas subyacentes en la función que representa a la totalidad de la arquitectura de una red neuronal, se puede concluir que la continuidad de Lipschitz de la función está relacionada con el fenómeno de sobreajuste. En esta investigación se demostró que, usando funciones de activación Lipschitz continuas, las arquitecturas de red neuronal convencional y residual, compuestas por composiciones de funciones de activación y transformación afines, son continuamente Lipschitz, si las funciones de activación lo son. Para este tipo de arquitecturas se estableció una cota de Lipschitz y se encontró que acotar su valor permitió reducir, en los experimentos realizados, el problema de sobreajuste.

A partir del análisis teórico se desarrolló el método Regularización adaptativa de Lipschitz con restricciones aleatorias (Regularización LBA), este método reduce la cota de Lipschitz a través de una regularización aleatoria, generando restricciones únicamente en alguna capa en cada iteración, y con parámetro de ponderación adaptativo. La adaptación del parámetro esta relacionada con el aporte a la cota de Lipschitz de la capa escogida aleatoriamente en cada iteración. El método fue evaluado a través de diferentes experimentos y variando algunos hiper-parámetros de las arquitecturas neuronales. Los resultados permiten concluir que el método regularización LBA presenta un desempeño destacable logrando: disminución significativa de la diferencia entre las curvas de aprendizaje y validación, menor variabilidad del modelo al cambiar el conjunto de entrenamiento y dinámica descendente del comportamiento de la cota de Lipschitz a través del proceso de aprendizaje.

A través de los experimentos realizados, que incluyeron la evaluación en: tres arquitecturas de red residual para los conjuntos de datos *Digits* MNIST y *Fashion* MNIST, tres conjuntos de datos sintéticos y tres de aplicaciones de contextos reales, se pudo verificar que el método desarrollado se destaca por disminuir la diferencia entre las curvas de aprendizaje y validación y mantener, en todos los experimentos, la tendencia a disminuir la cota de

Lipschitz. Además, en los experimentos de validación con imágenes adversarias generadas para los conjuntos *Digits* MNIST y *Fashion* MNIST se logró un desempeño favorable de la regularización LBA. Al validar las imágenes adversarias de *Digits* MNIST se superó en 46 puntos porcentuales la precisión lograda en el entrenamiento convencional y con las imágenes adversarias de *Fashion* MNIST, hasta un 41 %, esto para el nivel más alto de perturbación $\Gamma = 0{,}03$.

Los buenos resultados del método al ser validado en conjuntos con imágenes adversarias se puede explicar por la evidente reducción de la cota de Lipschitz. Dado que las imágenes adversarias son generadas agregando ruido a las originales y como se demostró en el apartado de análisis teórico, el ruido es acotado por la cota de Lipschitz, preservando que "datos similares se ubiquen en etiquetas similares". Entonces, debido a que el método disminuye la cota, también se disminuye la propagación del ruido presente en las imágenes adversarias. Por lo anterior, se puede concluir que para experimentos similares, el método propuesto puede ser empleado y escalado para generar arquitecturas robustas a ataques adversarios.

A nivel teórico se destaca que en esta investigación se empleó la idea de analizar la red neuronal como una función, en el sentido matemático, que mapea del espacio de los datos a las etiquetas. Esta propuesta brinda un marco conceptual que permite continuar estudiando propiedades y comportamientos de la función para relacionarlos con el desempeño de una red neuronal. Otra conclusión relevante radica en que se estableció que las técnicas de regularización que emplean funciones relacionadas con las normas de los pesos de la red son equivalentes entre ellas. En particular, el método propuesto emplea la norma de Frobenius de la matrices de pesos de alguna capa aleatoria, se encontró que esta norma es una cota superior de la norma espectral, función relacionada con el aporte a la cota de Lipschitz de la capa aleatoria. Esto permite concluir que al elegir parámetros de regularización adecuados se podría reducir la norma espectral optimizando la norma de Frobenius. Dado que en todos los experimentos la cota de Lipschitz disminuyó al usar la regularización LBA, se puede establecer que el parámetro de regularización adaptativo propuesto es adecuado para que la optimización de la norma de Frobenius incida en la disminución de la norma espectral, logrando disminuir la cota de Lipschitz.

Otro aspecto teórico relevante resulta en el acotamiento de la norma del gradiente de la función de pérdida por la cota de Lipschitz. Este hallazgo, realizado para el Perceptrón y una arquitectura simple de red residual, permite concluir que fenómenos como la explosión del gradiente se evitaría al restringir la cota de Lipschitz. De acuerdo con lo expuesto se puede concluir que la cota de Lipschitz es un factor crucial en el proceso de aprendizaje, ya que afecta directamente el funcionamiento del algoritmo de *Backpropagation*. Al utilizar este algoritmo, las perturbaciones en la entrada se propagan a través de la red, escaladas por la

cota de Lipschitz. Este efecto también se produce en la propagación hacia atrás, donde la magnitud de las derivadas parciales del error depende de la cota de Lipschitz. Por lo tanto, la cota puede tener un impacto significativo en la precisión y la estabilidad del proceso de aprendizaje.

Pese a las bondades del método propuesto también es importante mencionar que requiere el cálculo, en cada iteración, de la norma espectral de alguna matriz, este proceso es computacionalmente costoso y se dificulta a medida que aumenta la dimensión de la matriz. No obstante, se pueden implementar algoritmos que mejoren la eficiencia en el cálculo de la norma espectral. Por otra parte, la constante disminución de la cota de Lipschitz, limita en cierta medida, la posibilidad de alcanzar buenos desempeños en clasificación y al mismo tiempo satisfacer las restricciones impuestas sobre la cota de Lipschitz. En el método de regularización LBA se fortalece la capacidad de generalización, pero se reduce la precisión del aprendizaje, sin embargo, este valor es similar al logrado con los otros métodos.

Por otra parte, debido a que el método requiere cálculos iterativos para cada lote de observaciones es necesario establecer una relación entre el número de lotes y las capas del modelo. En arquitecturas con elevado número de capas se debe considerar un número de lotes comparable con el de capas, y de la misma manera en el caso de estructuras con pocas capas. Si bien el método desarrollado elimina la necesidad de sintonizar el parámetro de regularización, si requiere fijar de manera adecuada la relación de capas y el número de iteraciones en cada época. Esta sintonización puede resultar beneficiosa y mejorar los resultados del método. Sin embargo, en otros escenarios y debido al número de observaciones podrían no encontrarse un hiper-parámetro adecuado para la arquitectura deseada.

4.2. Perspectiva en tiempo continuo

En esta investigación se abordó la red neuronal residual desde el enfoque de una función que mapea del espacio de los datos al de las etiquetas. Otro enfoque relevante es el análisis en tiempo continuo, esta perspectiva se puede aplicar a redes de carácter residual y se analiza desde el enfoque de las ecuaciones diferenciales ordinarias. Este paradigma aparece en la literatura en el año 2018 cuando los investigadores Chen y otros acuñaron el termino Ecuaciones Diferenciales Ordinarias Neuronales ODENet [13]. En su investigación se propone considerar que la red neuronal residual ResNet corresponde a la discretización de una ecuación diferencial ordinaria (*Ordinary Differential Equation* - ODE). Si bien el término ODENet aparece referenciado en la literatura desde la publicación del artículo, ya se conocían algunos trabajos que siguen el enfoque teórico presentado por Chen y otros. Las investigaciones alrededor de este planteamiento están en continua exploración, ya que con el nuevo enfoque se brinda un marco teórico novedoso y flexible que puede ser empelado como escenario estudiar diversas

propiedades matemáticas, computacionales y aplicativas de la ResNet.

El estado del arte de este enfoque teórico ha generado diversos tipos de investigaciones. Por una parte se encuentran los estudios que consideran el problema de entrenamiento, definido por la Ecuación 2-10 y restringido con la Ecuación 2-11, en el modelo de tiempo continuo como un problema de control óptimo o control simultaneo [17, 120, 16, 121, 18, 50, 19]. Otras investigaciones se orientan a modificar el proceso de entrenamiento incluyendo bloques que resuelven numéricamente ecuaciones diferenciales al interior de la red neuronal, cada bloque corresponde a la propagación por alguna red residual de s capas [13, 122, 123].

Los estudios de la ResNet desde en enfoque del modelamiento en tiempo continuo también incluyen investigaciones que relacionan la robustes de la arquitectura con las propiedades matemáticas de la ecuación diferencial subyacente [20, 17, 21, 22]. En esta sección se presenta una breve síntesis de este enfoque y su posible relación con las condiciones que generan el problema de sobreajuste en redes de tipo residual.

4.2.1. Modelo matemático

Teniendo en cuenta el modelo que define la propagación a través de la red neuronal residual ResNet, formalizado en la Ecuación 2-10, se realiza la siguiente consideración $h = \delta f$, entonces se puede expresar el modelo como se describe a continuación

$$\mathbf{x}^{(j+1)} = \mathbf{x}^{(j)} + \delta f(\mathbf{x}^{(j)},\ \theta^{(j)}).$$

En la Ecuación anterior, δ es un valor suficientemente pequeño de manera que la propagación pueda ser considerada como la discretización de Euler para la ecuación diferencial ordinaria de primer orden presentada en la expresión 4-1.

$$\dot{\mathbf{x}} = f(\mathbf{x}(t),\ \theta(t)) \tag{4-1}$$

Este modelo es la perspectiva general abordada en la literatura [13, 21], el parámetro δ corresponde al tamaño de paso de la discretización. Algunos autores mencionan que $\delta = 1$ y no realizan el cambio de notación [17, 15]. Sin embargo, sin perdida de generalidad, la Ecuación 4-1 es el modelo en tiempo continuo de las capas residuales. Es importante resaltar que el valor del estado final $\mathbf{x}(T)$ es el correspondiente en la versión discreta a $\mathbf{x}^{(s)}$. Lo anterior implica que cuando el tiempo t, de la ODE 4-1, es igual horizonte final $t = T$, entonces el valor del estado $\mathbf{x}$, es análogo al valor de la propagación en la capa residual final. En este sentido, el número de capas residuales s corresponde a $s = T/\delta$.

Otra consideración relevante es la dimensión de la Ecuación 4-1. El vector de estados $\dot{\mathbf{x}} \in \mathcal{X}$, con $\mathcal{X} \subset \mathbb{R}^n$. Consideremos que las operaciones al interior del bloque residual corresponden a la función de activación de la transformación lineal $\mathbf{W}\mathbf{x} + b$ definida como se expresó en la Ecuación 2-11. En este caso el modelo continuo equivalente corresponde al sistema de ecuaciones diferenciales presentado en la Ecuación 4-2

$$\begin{cases} \dot{x_1} &= \sigma(\mathbf{W}_1(t)\mathbf{x} + b_1(t)) \\ \dot{x_2} &= \sigma(\mathbf{W}_2(t)\mathbf{x} + b_2(t)) \\ \vdots &= \qquad\qquad \vdots \\ \dot{x_n} &= \sigma(\mathbf{W}_n(t)\mathbf{x} + b_n(t)) \end{cases} \tag{4-2}$$

En la expresión 4-2 $\mathbf{x} = [x_1, x_2, \cdots, x_n]^T$, con x_i funciones que evolucionan en el tiempo. También se puede observar que $\mathbf{W}_i$, con $i = 1 : n$, es una función vectorial de la forma $\mathbf{W}_i = [w_{i1}(t), w_{i2}(t), \cdots w_{in}(t)]$ con w_{ji} funciones $\mathbb{R} \to \mathbb{R}$ dependientes de t, al igual que $b_i(t)$.

El sistema dinámico planteado en la Ecuación 4-2 es, generalmente, no lineal, por la naturaleza de la función de activación σ. Además, es variante en el tiempo si se considera la dependencia temporal de las funciones $\mathbf{W}$ y b. El estado final $x(T)$ se encontrará en el espacio $\mathcal{X}$, sin embargo en el problema de aprendizaje la salida de la red, relacionada con la etiqueta se encuentra en $\mathcal{Y}$, por lo tanto se considera una función final $g : \mathcal{X} \to \mathcal{Y}$. Esta función se interpreta como la capa de conexión final del bloque residual, y dependerá exclusivamente del estado final alcanzado.

4.2.2. Proceso de entrenamiento

Bajo la perspectiva planteada diversos autores relacionan el problema de aprendizaje con un problema de control óptimo. En particular el estudio [17] describe que dadas las dinámicas descritas por la Ecuación 4-2 y la función $g(\mathbf{x}(T))$, entonces partiendo de un estado inicial $\mathbf{x}(0)$, correspondiente a una observación $\mathbf{x}$ de las parejas de datos de entrenamiento $\{\mathbf{x}, y\}$, se requiere que el valor de $g(\mathbf{x}(T))$ sea lo más cercano a y. Esto se realiza a través de una función que estima la proximidad entre los valores, análogo a lo descrito en Sección 2.1 se pueden emplear las mismas funciones de costo, de manera que, siendo $\Phi : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ como a función de proximidad entre $g(\mathbf{x}(T))$ y y, el aprendizaje requiere minimizar esta función, sujeto a sus dinámicas.

La minimización anteriormente propuesta se puede realizar al encontrar las funciones $\mathbf{W}$ y $\mathbf{b}$, en este análisis denominados los controles. Tanto $\mathbf{W}$ y $\mathbf{b}$ deben pertenecer al conjunto de los controles factibles. Además, en el problema se cuenta con K parejas de la forma $\{\mathbf{x}, y\}$, y se debe realizar la minimización para todas, por lo cual se considera minimizar

$\sum_{i=1}^{K} \Phi(\mathbf{x}_i, y_i)$. Sobre los controles $\{\mathbf{W}, \mathbf{b}\}$ se pueden también establecer ciertas condiciones, por ejemplo, se podría considerar que su "energía" se minimice también durante el proceso, este planteamiento es análogo al uso de regularización en la función de costo en la versión discreta. De acuerdo con lo anterior, de manera general se puede plantear el problema de control óptimo como sigue

$$\min_{\mathbf{W}(t),\mathbf{b}(t)} \sum_{i=1}^{K} \Phi(g(\mathbf{x}_i(T), y_i) + \int_0^T \mathbf{W}(t)dt \tag{4-3}$$

$$\text{s.a } \dot{\mathbf{x}}_i = f(\mathbf{x}(\mathbf{t})_i, \mathbf{W}(t), \mathbf{b}(t)), \tag{4-4}$$

$$\text{con } \mathbf{x}_0 = \mathbf{x}_i, \ \forall i = 1, \cdots, K, \ \mathbf{y}, 0 \leq t \leq T. \tag{4-5}$$

En [17] el uso del Principio del mínimo de Pontryagin es postulado como forma de solución al problema de la Ecuación 4-3, sujeto a las dinámicas expresadas en Ecuación 4-4 y para las condiciones iniciales descritas por la Ecuación 4-5 . En la metodología propuesta se busca establecer el máximo para la función relacionada con el Hamiltonianio del sistema dinámico y sus funcionales.

La formulación del problema de aprendizaje bajo el enfoque del control óptimo permite, entre otras ventajas, analizar las condiciones de optimalidad, controlabilidad y estabilidad del problema y el sistema de ecuaciones diferenciales que gobierna las dinámicas. Por otra parte, esta propuesta permite pensar en la construcción de estrategias de solución desde el enfoque continuo y posteriormente discretizarlas para ser aplicables a la solución del problema de aprendizaje, similar a estudios como [17, 18].

Otro enfoque se plantea en [50], el problema de aprendizaje se trabaja desde el control simultaneo. En esta perspectiva cada observación $\mathbf{x}_i$ del conjunto de duplas $\{\mathbf{x}_i, y_i\}$, $i = 1, \ldots, K$ corresponde a un dato inicial diferente que se deben de enviar al espacio correcto de etiquetas y_i. Todos los datos iniciales se deben mapear correctamente con un control único $\{\mathbf{W}(t), \mathbf{b}(t)\}$, y bajo las dinámicas del sistema de ecuaciones que representa la red neuronal. En este estudio se destaca la propiedad de las funciones de activación de mantener la mitad del espacio invariante, condición de no linealidad que típicamente no se encuentra en sistemas dinámicos de inspiración mecánica.

4.2.3. Condiciones que generan el sobreajuste

El modelamiento en tiempo continuo de las estructuras residuales también permite estudiar el problema de sobreajuste y generalización. En el estudio [20] se planeta la importancia de considerar redes neuronales residuales estables y relaciona la capacidad de generalización con la estabilidad de la ecuación diferencial que representa la arquitectura. Existen diversos

conceptos para la estabilidad de un sistema dinámico, en particular el concepto de estabilidad de *Lyapunov* establece que si se parte de condiciones iniciales cercanas a un equilibrio, a través del tiempo la evolución del sistema se mantendrá en una distancia limitada de ese punto de equilibrio. La idea anterior se complementa con la importancia de los problemas bien definidos, se espera que la salida del sistema sea continua respecto a la entrada, de hecho esta noción de continuidad es la continuidad de *Lipschtiz*.

La Ecuación 4-2 modela una estructura neuronal residual, sin embargo este sistema de ecuaciones es no lineal y no autónomo, características que dificultan el análisis de la estabilidad (en cualquier sentido). En sistemas no lineales pero autónomos el análisis de la estabilidad local alrededor de un punto de equilibrio se puede encontrar con el Teorema de *Hartman-Grobman* [124, 125]. Este Teorema establece que en sistemas dinámicos con puntos de equilibrio hiperbólicos, los valores propios del Jacobiano pueden caracterizar la estabilidad del sistema en una cierta vecindad y bajo ciertas condiciones (ver Sección C.4).

Asumiendo una versión invariante en el tiempo del sistema de ecuaciones diferenciales establecido en la Ecuación 4-2 al eliminar la dependencia temporal de $\mathbf{W}(t) \equiv \mathbf{W}$ y $\mathbf{b}(t) \equiv \mathbf{b}$. Esta acción equivale a "congelar los coeficientes" de la matriz relacionada con el Jacobiano, también se puede interpretar como el análisis en un determinado tiempo, lo que es análogo a analizar en una determinada capa. En este caso se tiene las ecuaciones con matrices constantes de pesos y bias, como se presenta en Ecuación 4-6. El Teorema de *Hartman-Grobman* puede ser aplicado para analizar las condiciones de estabilidad en este sistema de ecuaciones diferenciales no lineal e invariante en el tiempo.

$$\begin{cases} \dot{x_1} &= \sigma(\mathbf{W}_1\mathbf{x} + b_1) \\ \dot{x_2} &= \sigma(\mathbf{W}_2\mathbf{x} + b_2) \\ \vdots &= \vdots \\ \dot{x_n} &= \sigma(\mathbf{W}_n\mathbf{x} + b_n) \end{cases} \tag{4-6}$$

Encontrando el Jacobiano asociado al sistema de ecuaciones se tiene que

$$J = \begin{bmatrix} \sigma'(\mathbf{W}_1\mathbf{x} + b_1)w_{11} & \cdots & \sigma'(\mathbf{W}_1\mathbf{x} + b_1)w_{1n} \\ \vdots & \ddots & \vdots \\ \sigma'(\mathbf{W}_n\mathbf{x} + b_n)w_{n1} & \cdots & \sigma'(\mathbf{W}_n\mathbf{x} + b_n)w_{nn} \end{bmatrix}.$$

J se puede factorizar como

$$J = \underbrace{\begin{bmatrix} \sigma'(\mathbf{W}_1\mathbf{x}+b_1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma'(\mathbf{W}_n\mathbf{x}+b_1) \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{nn} \end{bmatrix}}_{\mathbf{W}}.$$

Para analizar la parte real de los valores propios del Jacobiano se propone considerar el máximo valor de las partes reales, si este valor es menor que 0 entonces el sistema será estable en una vecindad de los puntos de equilibrio. Observe que el valor de los estados $\mathbf{x}$ se encuentra únicamente las componentes de la matriz diagonal Σ de la forma $\sigma'(\mathbf{W}_i\mathbf{x}+b_i)$, con $i = 1:n$, estos son también los valores propios de esta matriz. Asumiendo que σ es una función de activación convencional que satisface la propiedad de ser monótonamente creciente entonces, $\sigma'(\cdot) > 0$. Denotando λ_σ como el máximo de los valores propios de la matriz Σ entonces, $Re(\lambda_\sigma) > 0$.

Ahora, asumiendo que $\mathbf{W}$ es simétrica y definiendo la matriz $J_c = \Sigma^{\frac{1}{2}}\mathbf{W}\Sigma^{\frac{1}{2}}$, se tiene que $\mathbf{W}$ es congruente con J_c. Consecuencia de la Ley de Inercia de Sylvester las matrices J_c y $\mathbf{W}$ al ser simétricas y congruentes tienen la misma signatura, es decir poseen el mismo número de valores propios con parte real negativa, positiva y cero. Por otra parte, se puede verificar que $\Sigma^{\frac{1}{2}}J_c\Sigma^{-\frac{1}{2}} = \Sigma\mathbf{W}$. De lo anterior se establece que $\mathbf{J_c}$ es similar a $\Sigma\mathbf{W}$, de modo que tiene idénticos valores propios. Entonces $\mathbf{W}$ tiene la misma signatura que $\Sigma\mathbf{W}$.

De acuerdo con este análisis el número de valores propios con parte real negativa, positiva y cero del Jacobiano J depende de los valores propios de $\mathbf{W}$. Si la matriz $\mathbf{W}$ posee todos sus autovalores con parte real negativa, el Jacobiano también y el sistema de la Ecuación 4-6 será estable. Para este análisis se consideró que la restricción de simetría sobre $\mathbf{W}$, condición que no es siempre alcanzable en las matrices de pesos de una estructura neuronal. Además, se ha eliminado la dependencia temporal para establecer una versión invariante en el tiempo del sistema diferencial, este supuesto también debe ser analizado en el efecto del modelado del la arquitectura neuronal residual.

4.3. Trabajo futuro

La investigación desarrollada permite considerar, desde el análisis matemático de la función que representa la red neuronal, el control de propiedades para generar otros métodos de entrenamiento que tengan como propósito reducir el sobreajuste o mejorar el desempeño de las redes neuronales. Continuando con el estudio de la continuidad de Lipschitz y la propuesta de reducir la cota, se pueden proponer métodos adaptativos en los cuales la complejidad del modelo cambie de acuerdo con el acotamiento de la cota de Lipschitz. También se pueden

establecer otras cotas e indagar en el uso de métricas distintas, por ejemplo se podría considerar la norma nuclear de las matrices de pesos o combinaciones entre varias normas. A nivel teórico también se considera relevante estudiar la Proposición 3.1.1, en la cual se plantea que la optimización iterativa, fraccionando restricciones, puede ser similar a un problema sujeto a múltiples restricciones.

El método de regularización LBA se implementó en experimentos de arquitecturas residuales convencionales, como trabajo futuro se considera escalar la implementación hacia arquitecturas más complejas. Por ejemplo, se pueden proponer estructuras neuronales con operaciones convoluciones, normalización y *pooling*. Para realizar estos estudios es necesario demostrar que a través de estas operaciones se preserva la propiedad de continuidad de Lipschitz y establecer las cotas de cada capa de pesos para ser empleadas como factor adaptativo de regularización. A futuro también se puede indagar en la replica del método de regularización LBA para tareas diferentes a clasificación, se podrían considerar regresión, segmentación de imágenes y aplicaciones en series temporales.

Por otra parte, en el Sección 4.2 se presentó una breve perspectiva del análisis de las redes neuronales residuales como sistemas continuos. Este enfoque sugiere una diversidad de estudios que se pueden llevar a cabo. Estableciendo las redes neuronales como sistemas dinámicos continuos el proceso de aprendizaje puede trasladarse a resolver un problema de control óptimo o control simultaneo. Investigaciones futuras pueden incluir el desarrollo de algoritmos de entrenamiento en tiempo continuo que posteriormente puedan ser discretizados e implementados para las redes neuronales. Se espera que los nuevos algoritmos mejoren algún elemento del proceso de aprendizaje convencional, por ejemplo, la precisión, la convergencia, la memoria, entre otros. Además, en esta perspectiva se pueden estudiar propiedades como la controlabilidad, observabilidad y estabilidad, estos análisis podrían relacionarse con la reducción del sobreajuste u otras problemáticas de las redes neuronales.

Los experimentos llevados a cabo en este trabajo se realizaron con conjuntos de datos estándares, ampliamente usados en la validación de algoritmos de clasificación. Como trabajo futuro se propone indagar en la implementación y validación de la técnica propuesta en una aplicación del contexto real. Actualmente diversas aplicaciones requieren el uso de algoritmos de clasificación, a futuro se pueden guiar investigaciones en las cuales se establezca para una necesidad particular de cualquier sector y se acondicione el método propuesto para evaluar su desempeño. Se estima que la regularización LBA propuesta en este trabajo alcanza un nivel de maduración tecnológica (*Technology Readiness Level*) TRL 4[1] [126], logrando resultados adecuados en un entorno de simulación. El método tiene potencial para ser usado como tecnología de nivel operativo en entornos de aplicaciones reales, pero para

[1]Niveles de maduración tecnológica establecidos por el Ministerio de ciencia, tecnología e innovación

esto es necesario desarrollar investigaciones que abarquen todas las etapas de la maduración tecnológica partiendo desde el punto en que se encuentra.

A. Anexo: Propiedades funciones de activación

A.1. Función logística

La función logística, en su forma univariada, tiene la forma funcional descrita por la Ecuación A-1. Esta función $\sigma : \mathbb{R} \to \mathbb{R}$, tiene como rango el intervalo $(0, 1)$ y como dominio $\mathbb{R}$.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{A-1}$$

La función logística tiene las siguientes propiedades:

- Es monótonamente creciente
- Es continua en todo su dominio
- Es derivable en todo su dominio y su derivada es $\sigma(x)(1 - \sigma(x))$
- Es Lipschtiz continua
- La derivada de la función está acotada

Las dos últimas propiedades enunciadas se demuestran a continuación. Primero se verifica que la derivada de la función σ está acotada.

$$\begin{aligned}
|\sigma'(x)| &= |\sigma(x)(1 - \sigma(x))| \\
&= |\sigma(x)||(1 - \sigma(x))| \\
&\leq |\sigma(x)||(1 - \sigma(x))| \\
&\leq \left| 1 - \frac{1}{1 + e^{-x}} \right| \\
&\leq \left| \frac{e^{-x}}{(1 + e^{-x})^2} \right| \\
&\leq \frac{1}{4}
\end{aligned}$$

□

A continuación, sean dos entradas arbitrarias x_a, x_b ambas $\in \mathbb{R}$, una tercera entrada x_c tal que $x_a < x_c < x_b$ y empleando el **Teorema del Valor Medio** se tiene

$$\begin{aligned}
\frac{\sigma(x_b) - \sigma(x_a)}{x_b - x_a} &= \sigma'(x_c) \\
\left|\frac{\sigma(x_b) - \sigma(x_a)}{x_b - x_a}\right| &= |\sigma'(x_c)| \\
\frac{|\sigma(x_b) - \sigma(x_a)|}{|x_b - x_a|} &\leq \frac{1}{4} \\
|\sigma(x_b) - \sigma(x_a)| &\leq \frac{1}{4}|x_b - x_a|,
\end{aligned}$$

en consecuencia la función σ es Lipschitz □. La constante de Lipschitz denotada como LB_σ se puede encontrar de la siguiente manera:

$$\begin{aligned}
LB_\sigma &= \sup\left|\frac{\sigma(x_b) - \sigma(x_a)}{x_b - x_a}\right| \\
&= \sup|\sigma'(x_c)| \\
&= \frac{1}{4}.
\end{aligned}$$

□

Dado que la función logística satisface $|\sigma(x_b) - \sigma(x_a)| \leq LB|x_b - x_a|$ para $0 \leq LB < 1$, se denomina contracción o función corta. En este tipo se funciones, definidas en un espacio métrico, se satisface el **Teorema del punto fijo de Banach**, el cual establece que para una contracción definida en algún espacio métrico completo tiene un único punto fijo. Para el caso particular univariado de la función logística y por su definición en el espacio $\mathbb{R}$ (métrico y completo), se tiene que para algún x^* $\sigma(x^*) = x^*$, además, la secuencia de composiciones del tipo

$$\begin{aligned}
\{x, \sigma(x),\ \sigma \circ \sigma(x), \cdots, \sigma \circ \cdots \sigma(x)\} \\
\{x, \sigma(x),\ \sigma(\sigma(x)), \cdots, \sigma(\cdots(\sigma(x)))\}
\end{aligned}$$

converge al punto fijo.

A.2. Función ReLU

La función Rectificador Lineal ReLU, en su forma univariada, tiene la forma funcional descrita por la Ecuación A-2, tambien es posible describirla por trozos como se presenta en la

Ecuación A-3. Esta función se define en $\text{ReLU} : \mathbb{R} \to \mathbb{R}$, tiene como rango el intervalo $[0, \infty)$ y como dominio $\mathbb{R}$.

$$\text{ReLU}(x) = \text{máx}\,(0, x) \tag{A-2}$$

$$\text{ReLU}(x) = \begin{cases} 0 & \text{si} \quad x \leq 0 \\ x & \text{si} \quad x > 0 \end{cases} \tag{A-3}$$

La función ReLU tiene las siguientes propiedades:

- Es monótonamente creciente
- Es continua en todo su dominio
- Es diferenciable en todo su dominio salvo en $x = 0$, para cualquier otro punto la derivada se define a trozos como

$$\text{ReLU}'(x) = \begin{cases} 0 & \text{si} \quad x < 0 \\ 1 & \text{si} \quad x > 0 \\ \text{indeterminado} & \text{si} \quad x = 0 \end{cases} \tag{A-4}$$

- Es convexa
- Es 1-Lipschtiz

Se demuestra el ultimo ítem, para esto considere $x_1, x_2 \in \mathbb{R} \geq 0$ entonces,

$$|f(x_1) - f(x_2)| = |x_1 - x_2|,$$

por lo tanto, la constante de Lipschitz denotada como LB_{ReLU} corresponde a 1.

Si se tiene que $x_1 \geq 0$ y $x_2 < 0$ entonces $x_2 \leq x_1$, por lo tanto

$$\begin{aligned} |f(x_1) - f(x_2)| &= |x_1 - 0| \\ &= |x_1|, \end{aligned}$$

luego, $x_1 - x_2 \geq 0$, por lo tanto, $|x_1| \leq |x_1 - x_2|$ y así $LB_{\text{ReLU}} = 1$.

Ahora, si se tiene $x_1 < 0$ y $x_2 \geq 0$ entonces,

$$\begin{aligned} |f(x_1) - f(x_2)| &= |0 - x_2| \\ &= |x_2|. \end{aligned}$$

Dado que $x_2 - x_1 \geq 0$ luego $|x_2 - x_1| \geq 0$, $|x_2| \leq |x_2 - x_1|$, lo que es equivalente a $|x_2| \leq |x_1 - x_2|$ con $LB_{\text{ReLU}} = 1$.

En el caso que $x_1 < 0$ y $x_2 \leq 0$, entonces $|f(x_1) - f(x_2)| = 0$, y como $|x_1 - x_2| > 0$, entonces $|f(x_1) - f(x_2)| \leq |x_1 - x_2|$ y $LB_{\text{ReLU}} = 1$.

A.3. Función tangente hiperbólica

La función tangente hiperbólica tiene la forma funcional descrita por la Ecuación A-5

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{A-5}$$

El dominio de esta función corresponde al conjunto de los números $\mathbb{R}$, y el rango es el intervalo $(-1, 1)$.

La siguientes son algunas propiedades de la función:

- Es monótonamente creciente
- Es continua en todo su dominio
- Es diferenciable en todo su dominio con derivada

$$\tanh'(x) = 1 - \tanh^2(x)$$

- Es convexa en el intervalo $(-\infty, 0)$ y cóncava en $(0, +\infty)$
- Es Lipschitz continua
- La derivada de la función está acotada

Las dos últimas propiedades enunciadas se demuestran a continuación. Primero se verifica que la derivada de la función tanh está acotada, esto se puede analizar fácilmente, ya que

$$\begin{aligned}\tanh'(x) &= 1 - \tanh^2(x) \\ &= \operatorname{sech}^2(x) \\ &\leq 1.\end{aligned}$$

Siguiendo el análisis desarrollado para la función sigmoide, entonces se puede concluir que la función tangente hiperbólica es también Lipschitz continua. La constante de Lipschitz denotada como LB_{th}, se puede encontrar de la siguiente manera:

$$\begin{aligned}LB_{th} &= \sup \left| \frac{\sigma(x_b) - \sigma(x_a)}{x_b - x_a} \right| \\ &= \sup |\sigma'(x_c) \\ &= 1.\end{aligned}$$

□

B. Anexo: Propiedad Lipschitz en algunas redes neuronales

B.1. Composición de funciones Lipschitz

De acuerdo a lo estudiado en el Capitulo 3, las redes neuronales pueden ser vistas como la composición, generalmente no lineal, de transformaciones afines, su formulación matemática se presenta en la Ecuación 2-3. Las funciones de activación σ, en la mayoria de arquitecturas, son Lipschitz Continuas, esto motiva cuestionarse si la composición de la Ecuación 2-3 preserva esta propiedad.

Una proposición a considerarse es la siguiente:

Proposición B.1.1

Sean $f_1 : \mathcal{X} \to \mathbb{R}^m$ con $\mathcal{X} \subseteq \mathbb{R}^n$ y $f_2 : \mathcal{Y} \to \mathbb{R}^p$ con $\mathcal{Y} \subseteq \mathbb{R}^m$, funciones Lipschitz continuas, con constantes M_1 y M_2, respectivamente. De tal manera que para $x_a, x_b \in \mathcal{X}$ se tiene que,

$$||f_1(x_a) - f_1(x_b)|| \leq M_1||x_a - x_b||$$

y para $y_a, y_b \in \mathcal{Y}$

$$||f_2(y_a) - f_2(y_b)|| \leq M_2||y_a - y_b||$$

Entonces la función

$$f_3(x) = f_1 \circ f_2$$

es también Lipschtiz continua, y su constante de Lipschitz es $M_1 M_2$.

Considérese que para y_a, y_b la función f_2 actúa de la siguiente manera $x_a = f_2(y_a)$ y $x_b = f_2(y_b)$. Entonces, se puede observar que:

$$\begin{aligned} ||f_1(f_2(y_a))) - f_1(f_2(y_b)))|| &= ||f_1(x_a) - f_1(x_b)|| \\ &\leq M_1||x_a - x_b|| \\ &= M_1||f_2(y_a) - f_2(y_b)|| \\ &\leq M_1 M_2||y_a - y_b|| \end{aligned}$$

Entonces se cumple que,

$$||f_3(y_a) - f_3(y_b)|| \leq M_1 M_2||y_a - y_b||$$

Por lo tanto, f_3, composición de las funciones f_1 y f_2, es Lipschitz Continua. □

Esta propiedad se puede generalizar para la composición de l funciones Lipschitz, típicamente este sería es el caso de las redes neuronales. En este escenario, una cota superior para una red neuronal con l capas completamente conectadas será el producto de la constantes de Lipschitz de cada una de las capas. Teniendo en cuenta que las funciones matemáticas que representan la transformación entre las capas garanticen la continuidad de Lipschitz. Entonces, si para cada capa j la constante de Lipschitz se denota como $L_i(\sigma^{(j)})$, la cota superior de toda la red denotada como LB será:

$$LB = \prod_{j=1}^{L} L_i(\sigma^{(j)})$$

B.2. Transformación afín

En este apartado se analiza si una transformación del tipo $T : \mathbf{Wx}+\mathbf{b}$ es una Lipschitz Continua. Dado que una red neuronal convencional es la composición de funciones de activación con transformación de este tipo y se estudió que la composición de funciones Lipschitz también lo es, entonces para estudiar el comportamiento general de la red es necesario analizar la propiedad en la transformación.

Proposición B.2.1

Sean $T : \mathcal{X} \to \mathbb{R}^m$ con $\mathcal{X} \subseteq \mathbb{R}^n$ una transformación afín definida como $T(\mathbf{x}) = \mathbf{Wx}+\mathbf{b}$, T satisface la propiedad de ser Lipschitz Continua. De tal manera que para $x_a, x_b \in \mathcal{X}$

se tiene que existe un $M_T \geq 0 \in \mathbb{R}$ tal que:

$$||T(x_a) - T(x_b)|| \leq M_T||x_a - x_b||$$

Veamos que:

$$\begin{aligned} ||T(x_a) - T(x_b)||^2 &= ||(\mathbf{W}x_a + b) - (\mathbf{W}x_b + b)||^2 \\ &= ||\mathbf{W}(x_a - x_b)||^2, \end{aligned}$$

denotando la matriz $\mathbf{W}$ de dimensión $m \times n$ por los vectores filas de manera que

$$\mathbf{W} = \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_m \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \cdots w_{1n} \\ w_{21} & w_{22} & \cdots w_{2n} \\ \vdots & & \\ w_{m1} & w_{m2} & \cdots w_{mn} \end{bmatrix},$$

y llamando $u = x_a - x_b$, entonces tenemos

$$\begin{aligned} ||T(x_a) - T(x_b)||^2 &= ||\mathbf{W}(x_a - x_b)||^2 \\ &= \left|\left| \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_m \end{bmatrix} u \right|\right|^2 \\ &= \left|\left| \begin{bmatrix} < W_1, u > \\ < W_2, u > \\ \vdots \\ < W_m, u > \end{bmatrix} \right|\right|^2. \end{aligned}$$

Ahora, tomando la norma euclidiana se tiene que:

$$\begin{aligned} ||T(x_a) - T(x_b)||^2 &= ||\mathbf{W}(x_a - x_b)||^2 \\ &= \left|\left| \begin{bmatrix} < W_1, u > \\ < W_2, u > \\ \vdots \\ < W_m, u > \end{bmatrix} \right|\right|^2 \\ &= < W_1, u >^2 + < W_2, u >^2 + \cdots + < W_m, u >^2 \\ &= | < W_1, u > |^2 + | < W_2, u > |^2 + \cdots + | < W_m, u > |^2, \end{aligned}$$

y aplicando la desigualdad de Cauchy-Schwarz entonces $| < W_i, u > |^2 \leq ||W_i||^2||u||^2$, $\forall i = 1 : m$, de modo que

$$\begin{aligned} ||T(x_a) - T(x_b)||^2 &= | < W_1, u > |^2 + | < W_2, u > |^2 + \cdots + | < W_m, u > |^2 \\ &\leq ||W_1||^2||u||^2 + ||W_2||^2||u||^2 + \cdots + ||W_m||^2||u||^2 \\ &\leq ||u||^2(||W_1|| + ||W_2|| + \cdots + ||W_m||^2). \end{aligned}$$

llamando $||W_1|| + ||W_2|| + \cdots + ||W_m||^2 = M_T^2$, claramente una constante positiva $M_T^2 \geq 0$, entonces tenemos que:

$$||T(x_a) - T(x_b)||^2 \leq ||u||^2(||W_1|| + ||W_2|| + \cdots + ||W_m||^2) \quad \leq M_T^2||u||^2 \leq M_T^2||x_a - x_b||^2,$$

finalmente, tomando la raíz positiva en ambos lados de la desigualdad se tiene

$$\begin{aligned} ||T(x_a) - T(x_b)|| &\leq |M_T| \ ||x_a - x_b|| \\ ||T(x_a) - T(x_b)|| &\leq M_T||x_a - x_b|| \end{aligned}$$

Por lo tanto, T, una transformación afín, es Lipschitz Continua. □

B.3. Constante de Lipschitz transformación afín

De acuerdo a la demostración A1 una cota superior para la constante de Lipschitz corresponde al producto de las constantes de las funciones involucradas en la composición. Para alguna capa j la salida $x^{(j+1)}$ y se asocia con una función f_c de tal manera que $x^{(j+1)} = f_c(x^{(j)}) = \sigma(\mathbf{W}x^{(j)} + \mathbf{b})$. La función f_c es la composición de una función σ y de una transformación afín $\mathbf{W}\mathbf{x} + b$.

A continuación se determina el valor de la constante de Lipschitz para la función afín $T : \mathbb{X} \to \mathbb{R}^m$, $\mathcal{X} \subseteq \mathbb{R}^n$, tal que $T(\mathbf{x}) = \mathbf{W}\mathbf{x} + b$. Dado que las funciones afines son Lipschitz continuas se satisface que para $x_a, x_b \in \mathcal{X}$:

$$\begin{aligned} ||T(x_a) - T(x_b)|| &\leq M_T||x_a - x_b|| \\ ||(\mathbf{W}x_a + b) - (\mathbf{W}x_b + b)|| &\leq M_T||x_a - x_b|| \\ ||\mathbf{W}x_a - \mathbf{W}x_b|| &\leq M_T||x_a - x_b|| \\ ||\mathbf{W}(x_a - x_b)|| &\leq M_T||x_a - x_b|| \end{aligned}$$

Se puede denotar $y = x_a - x_b$, dado que $x_a \neq x_b$ entonces $y \neq \mathbf{0}$, de manera que la expresión anterior puede verse como:

$$\begin{aligned} ||\mathbf{W}(x_a - x_b)|| &\leq M_T||x_a - x_b|| \\ ||\mathbf{W}y|| &\leq M_T||y|| \\ \frac{||\mathbf{W}y||}{||y||} &\leq M_T \end{aligned}$$

De modo que,

$$M_T = \sup\left\{\frac{||\mathbf{W}y||}{||y||}\right\}$$

De acuerdo con lo anterior se puede establecer que la constante de Lipschitz para una transformación afín es la norma inducida de la matriz $\mathbf{W}$. Tomando la norma euclidiana, la norma inducida corresponde a la norma espectral que se define como el mayor valor singular de $\mathbf{W}$, o a la raíz cuadrada del máximo valor propio de la matriz semidefinida positiva $\mathbf{W}^T\mathbf{W}$.

$$M_T = \sqrt{\rho(\mathbf{W}^T\mathbf{W})} = \sqrt{\max \lambda_i(\mathbf{W}^T\mathbf{W})}$$

B.4. Propiedad Lipschitz de un bloque residual

Considere una red neuronal compuesta por un bloques residuales de tal manera que la salida para alguna capa $j+1$ denotada como $x^{(j+1)}$ esta determinada por la función

$$x^{(j+1)} = f_s(x^{(j)}, \theta_s^{(j)}) + f(x^{(j)}, \theta^{(j)}) \tag{B-1}$$

Con f la función del bloque residual tal que $f : \mathcal{X} \to \mathbf{R}^n$ y f_s la función del salto, igual que f se tiene que preservar que $f_s : \mathcal{X} \to \mathbf{R}^n$ con $\mathcal{X} \subseteq \mathbf{R}^n$. Donde $\theta_s^{(j)}$ son los parámetros de la función de salto y $\theta^{(j)}$ parámetros de la función del bloque residual.

Asumiendo que la función del bloque residual f y de la conexión de salto f_s son Lipschitz Continuas, de tal forma que para dos entradas cualesquiera en la capa j x_a y x_b, se cumple que:

$$||f(x_a) - f(x_b)|| \leq M_f||x_a - x_b|| \tag{B-2}$$

$$||f_s(x_a) - f_s(x_b)|| \leq M_s||x_a - x_b|| \tag{B-3}$$

Entonces bajo estas condiciones, se plante la siguiente proposición:

Proposición B.4.1

Sean $f, f_s : \mathcal{X} \to \mathbb{R}$ con $\mathcal{X} \subseteq \mathbb{R}^n$ con parámetros $\theta^{(j)}$ y $\theta_s^{(j)}$, funciones Lipschitz continuas, con constantes M_f y M_s, Entonces la función

$$f_{BR}(x) = f(x, \theta^{(j)}) + f_s(x, \theta_s^{(j)})$$

es también Lipschtiz continua, y su constante de Lipschitz es $(M_f + M_s)$.

Entonces, se puede observar que para $x_a, x_b \in \mathcal{X}$:

$$\begin{aligned}
||f_{BR}(x_a) - f_{BR}(x_b)|| &= ||(f(x_a, \theta^{(j)}) + f_s(x_a, \theta_s^{(j)})) - (f(x_b, \theta^{(j)}) + f_s(x_b, \theta_s^{(j)}))|| \\
&= ||(f(x_a, \theta^{(j)}) - f(x_b, \theta^{(j)})) + (f_s(x_a, \theta_s^{(j)}) - f_s(x_b, \theta_s^{(j)})|| \\
&\leq ||f(x_a, \theta^{(j)}) - f(x_b, \theta^{(j)})|| + ||f_s(x_a, \theta_s^{(j)}) - f_s(x_b, \theta_s^{(j)})|| \\
&\leq M_f||x_a - x_b|| + M_s||x_a - x_b|| \\
&= (M_f + M_s)||x_a - x_b||
\end{aligned}$$

Entonces se cumple que,

$$||f_{BR}(x_a) - f_{BR}(x_b)|| \leq (M_f + M_s)||x_a - x_b||,$$

por lo tanto, f_{BR} es Lipschitz Continua. □

De acuerdo a lo anterior, en una red neuronal con bloque residuales la función del bloque f_{BR}, que transforma la entrada de la capa $x^{(j)}$ en $x^{(j+1)}$, preserva la continuidad de Lipschitz si las funciones del bloque y del salto satisfacen esta propiedad. En el caso que f_s sea la función identidad, la cual es Lipschitz continua con constante igual a la unidad, entonces la función del bloque residual f_{BR} es Lipschitz Continua con constante $1 + M_f$.

C. Anexo: Demostraciones y teoremas relevantes

C.1. Combinación lineal de funciones Lipschitz

Teorema C.1.1

Sean f_1, f_2 funciones Lipschitz y sean a y b constantes $\in \mathbb{R}$, entonces la función $h = af_1 + bf_2$, es Lipschitz, con $h : \mathcal{X} \to \mathcal{Y}$ y la definición de f_1 y f_2 coherente con los espacios de definición de h.

Sean x_1, $x_2 \in \mathcal{X}$, se debe mostrar que existe un $LB \in \mathbb{R} > 0$, tal que

$$||h(x_1) - h(x_2)|| \leq LB||x_1 - x_2||.$$

Dado que f_1 y f_2 son Lipschitz entonces satisfacen que

$$||f_1(x_1) - f_1(x_2)|| \leq LB_1||x_1 - x_2||$$

y

$$||f_2(x_1) - f_2(x_2)|| \leq LB_2||x_1 - x_2||,$$

respectivamente.

Veamos que

$$\begin{aligned}
||(af_1 + bf_2)(x_1) - (af_1 + bf_2)(x_2)|| &= ||(af_1(x_1) + bf_2(x_1) - (af_1(x_2) + bf_2(x_2)|| \\
&= ||a(f_1(x_1) - f_1(x_2)) + b(f_2(x_1) - f_2(x_2)|| \\
&\leq ||a(f_1(x_1) - f_1(x_2))|| + ||b(f_2(x_1) - f_2(x_2))|| \\
&\leq |a|||f_1(x_1) - f_1(x_2)|| + |b|||f_2(x_1) - f_2(x_2)|| \\
&\leq |a|LB_1|x_1 - x_2| + |b|LB_2|x_1 - x_2| \\
&\leq (|a|LB_1 + |b|LB_2)|x_1 - x_2|.
\end{aligned}$$

Se puede ver que $(|a|LB_1 + |b|LB_2)$ es una constante positiva, se denominara LB_h, entonces se tiene que

$$||(af_1 + bf_2)(x_1) - (af_1 + bf_2)(x_2)|| \leq (|a|LB_1 + |b|LB_2)|x_1 - x_2|,$$

lo que implica que la función h definida como una combinación lineal de dos funciones Lipschitz $h = af_1 + bf_2$, es también Lipschitz. □

C.2. Subespacio vectorial de funciones Lipschitz

Definición C.2.1

Sea $\mathcal{L} = \{f : \mathcal{X} \to \mathcal{Y} : f \text{ es Lipschitz}\}$ con $\mathcal{X}$ y $\mathcal{Y}$ espacios métricos se define $\mathcal{L}$ como el **Conjunto de Funciones Lipschitz**.

Proposición C.2.1

$\mathcal{L}$ es un subespacio vectorial del conjunto de funciones $f : \mathcal{X} \to \mathcal{Y}$.

Usando el Teorema demostrado en Sección C.1 se sabe que $\mathcal{L}$ es cerrado para la suma y el producto escalar.

C.3. Equivalencia de las normas en $\mathbb{R}^n$

Definición C.3.1

Sea V un espacio vectorial normado y $||\cdot||_1$ y $||\cdot||_2$ dos normas en V, las dos normas son equivalentes si y solo si existen $a, b > 0 \in \mathbb{R}$ tal que se satisface:

$$a||\cdot||_1 \leq ||\cdot||_2 \leq b||\cdot||_1.$$

Proposición C.3.1

En $\mathbb{R}^n$ como espacio vectorial dos normas cualesquiera son equivalentes.

Para demostrar esta proposición partimos de establecer que la relación de equivalencia cumple las propiedades de ser **reflexiva**, **simétrica** y **transitiva**. Esta última propiedad permite concluir que si una norma cualquiera $||\cdot||$ es equivalente a la norma $||\cdot||_1$, entonces cualesquiera dos normas serán equivalentes entre sí, entonces bastará encontrar que dada una

norma cualquiera $||\cdot||$ esta será equivalente a la norma $||\cdot||_1$.

En $\mathbb{R}^n$, un elemento cualquiera de la forma $\mathbf{x} = [x_1, x_2, ..., x_n]$ pueden representarse en relación a las bases canónicas $e_1 = [1, 0, ..., 0]$, $e_2 = [0, 1, 0, ..,0]$, ..., $e_n = [0, 0, ..., 1]$, así

$$\mathbf{x} = x_1e_1 + x_2e_2 + ... + x_ne_n,$$

con $x_i \in \mathbb{R}$, y los elementos x_ie_i vectores. Por lo tanto la $||\cdot||$, puede representarse

$$||\mathbf{x}|| = ||x_1e_1 + x_2e_2 + ... + x_ne_n||.$$

Dado que los x_i corresponden a números reales por propiedades se tiene que:

$$\begin{aligned} ||\mathbf{x}|| &= ||x_1e_1 + x_2e_2 + ... + x_ne_n|| \\ &\leq ||x_1e_1|| + ||x_2e_2|| + ...||x_ne_n|| \\ &= \sum_{i=0}^{n} ||x_ie_i|| \\ &= \sum_{i=0}^{n} |x_i|||e_i||. \end{aligned}$$

Definiendo $b = \text{máx}\,||e_i||, \forall i = 1 : n$, entonces,

$$\begin{aligned} ||\mathbf{x}|| &\leq \sum_{i=0}^{n} |x_i|||e_i|| \\ &\leq \sum_{i=0}^{n} |x_i|b \\ &= b||\mathbf{x}||_1. \end{aligned}$$

Este resultado implica que

$$||\mathbf{x}|| \leq b||\mathbf{x}||_1. \tag{C-1}$$

Ahora, considerando $S = \{\mathbf{x} : ||\mathbf{x}||_1 \leq 1\}$, este conjunto tendrá un mínimo $||m||_1$ para el cual se cumple que $||m||_1 < ||n||_1$, para todo $||n||_1 \in S$. ahora bien, no se considera al elemento $x = 0$, por lo tanto $||m|| > 0$. Los elementos de la forma $\mathbf{x}/||\mathbf{x}||_1$ están contenidos en el conjunto S, ya que $\left|\left|\frac{\mathbf{x}}{||\mathbf{x}||_1}\right|\right| = 1$. Entonces, $||m||_1$ también es menor para $\left|\left|\frac{\mathbf{x}}{||\mathbf{x}||_1}\right|\right|$. Denotando $a = ||m||$, entonces

$$a \leq \left|\left| \frac{\mathbf{x}}{||\mathbf{x}||_1} \right|\right|,$$

por lo tanto

$$a||\mathbf{x}||_1 \leq ||\mathbf{x}||. \tag{C-2}$$

De las ecuaciones C-1 y C-2, se tiene que para dos números reales positivos a, b

$$a||\mathbf{x}||_1 \leq ||\mathbf{x}|| \leq b||\mathbf{x}||_1.$$

demostrada esta equivalencia y por la propiedad de transitividad se demuestra que dos normas cualesquiera en $\mathbb{R}^n$ son equivalentes. □

C.4. Enunciado del Teorema de Hartman-Grobman

Definición C.4.1

Sea el sistema de n ecuaciones diferenciales $\dot{\mathbf{x}} = f(\mathbf{x})$ y $\mathbf{x}_0 \in \mathbb{R}^n$, se dice que $\mathbf{x}_0$ es un punto de equilibrio hiperbólico si ningún valor propio de la matriz $Df(\mathbf{x}_0)$ tiene parte real nula.

Teorema C.4.1

Sea $\mathbf{x}_0 \in \mathbb{R}^n$ un equilibrio hiperbólico del sistema de ecuaciones diferenciales $\dot{\mathbf{x}} = f(\mathbf{x})$, con $f : \mathbb{R}^n \to \mathbb{R}^n$, $f \in C^1$. Entonces existe un entorno U de $\mathbf{x}_0$, y V de $\mathbf{0}$, tales que los sistemas diferenciales

$$\dot{\mathbf{x}} = f(\mathbf{x})$$
$$\dot{\mathbf{x}} = Df(\mathbf{x}_0)\mathbf{x},$$

son topológicamente conjugados.

D. Implementación computacional

La implementación computacional se encuentra en el repositorio https://github.com/mavivi95/overfittingLipschitzBoundLPAR. A continuación se detalla por experimento su ubicación:

- Experimento A, Subsección 2.3.2: https://github.com/mavivi95/overfittingLipschitzBound/blob/main/overfittingExperimentoA.ipynb.
- Experimento B, Subsección 2.3.2: github.com/mavivi95/overfittingLipschitzBound/blob/main/overfittingExperimentoB.ipynb.
- Experimento A, Subsección 3.2.1: https://github.com/mavivi95/overfittingLipschitzBound/blob/main/validationMethodLBExperimentoA.ipynb.
- Experimento B, Subsección 3.2.1: https://github.com/mavivi95/overfittingLipschitzBound/blob/main/validationMethodLBExperimentoB.ipynb.
- Experimento C, conjunto de datos Iris, Subsección 3.2.1: https://github.com/mavivi95/overfittingLipschitzBound/blob/main/validationMethodLBExperimentoC1.ipynb
- Experimento C, conjunto de datos *Wine dataset*, Subsección 3.2.1: https://github.com/mavivi95/overfittingLipschitzBound/blob/main/validationMethodLBExperimentoC2.ipynb
- Experimento C, conjunto de datos *Cancer dataset*, Subsección 3.2.1: https://github.com/mavivi95/overfittingLipschitzBound/blob/main/validationMethodLBExperimentoC3.ipynb
- Experimento D, círculos concéntricos, Subsección 3.2.1: https://github.com/mavivi95/overfittingLipschitzBound/blob/main/validationMethodLBExperimentoD1.ipynb.
- Experimento D, medias lunas, Subsección 3.2.1: https://github.com/mavivi95/overfittingLipschitzBound/blob/main/validationMethodLBExperimentoD2.ipynb

- Experimento D, espirales concéntricos, Subsección 3.2.1: https://github.com/mavivi95/overfittingLipschitzBound/blob/main/validationMethodLBExperimentoD3.ipynb
- Experimento A, Subsección 3.2.2: https://github.com/mavivi95/overfittingLipschitzBound/blob/main/validationMethodLBExperimentoAadv.ipynb.
- Experimento B, Subsección 3.2.2: https://github.com/mavivi95/overfittingLipschitzBound/blob/main/validationMethodLBExperimentoBadv.ipynb.

Bibliografía

[1] C. C. Aggarwal, *Neural Networks and Deep Learning*. Springer, 2018.

[2] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," 2018. [Online]. Available: https://arxiv.org/abs/1807.05511

[3] A. Kamilaris and F. X. Prenafeta-Boldú, "A review of the use of convolutional neural networks in agriculture," *The Journal of Agricultural Science*, vol. 156, no. 3, p. 312–322, 2018. [Online]. Available: https://doi.org/10.1017/S0021859618000436

[4] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, jan 2018. [Online]. Available: https://doi.org/10.1109%2Fmsp.2017.2765202

[5] A. Fadaeddini, M. Eshghi, and B. Majidi, "A deep residual neural network for low altitude remote sensing image classification," in *2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS)*, 2018, pp. 43–46. [Online]. Available: https://ieeexplore.ieee.org/document/8336623

[6] L. Massidda, M. Marrocu, and S. Manca, "Non-intrusive load disaggregation by convolutional neural network and multilabel classification," *Applied Sciences*, vol. 10, no. 4, 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/4/1454

[7] K. Muralitharan, R. Sakthivel, and R. Vishnuvarthan, "Neural network based optimization approach for energy demand prediction in smart grid," *Neurocomputing*, vol. 273, pp. 199–208, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231217313681

[8] O. Al-Salman, J. Mustafina, and G. Shahoodh, "A systematic review of artificial neural networks in medical science and applications," in *2020 13th International Conference on Developments in eSystems Engineering (DeSE)*, 2020, pp. 279–282. [Online]. Available: https://ieeexplore.ieee.org/document/9450245

[9] M. M. Bejani and M. Ghatee, "A systematic review on overfitting control in shallow and deep neural networks," *Artificial Intelligence Review*, vol. 54, no. 8, pp. 6391–6438, 2021. [Online]. Available: https://doi.org/10.1007/s10462-021-09975-1

[10] S. Salman and X. Liu, "Overfitting mechanism and avoidance in deep neural networks," *arXiv preprint arXiv:1901.06566*, 2019. [Online]. Available: https://arxiv.org/abs/1901.06566

[11] X. Ying, "An overview of overfitting and its solutions," in *Journal of physics: Conference series*, vol. 1168, no. 2. IOP Publishing, 2019, p. 022022. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/1168/2/022022

[12] I. Bilbao and J. Bilbao, "Overfitting problem and the over-training in the era of data: Particularly for artificial neural networks," in *2017 eighth international conference on intelligent computing and information systems (ICICIS)*. IEEE, 2017, pp. 173–177. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8260032

[13] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in neural information processing systems*, vol. 31, 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf

[14] D. Karlsson and O. Svanström, "Modelling dynamical systems using neural ordinary differential equations," Master's thesis, Chalmers University of Technology, 2019. [Online]. Available: https://odr.chalmers.se/server/api/core/bitstreams/83a4c17f-35e7-43ce-ac60-43a0b799f82f/content

[15] E. Weinan, "A proposal on machine learning via dynamical systems," *Communications in Mathematics and Statistics*, vol. 1, no. 5, pp. 1–11, 2017. [Online]. Available: https://link.springer.com/article/10.1007/s40304-017-0103-z

[16] M. Benning, E. Celledoni, M. J. Ehrhardt, B. Owren, and C.-B. Schönlieb, "Deep learning as optimal control problems: Models and numerical methods," *arXiv preprint arXiv:1904.05657*, 2019. [Online]. Available: https://arxiv.org/abs/1904.05657

[17] Q. Li, L. Chen, C. Tai *et al.*, "Maximum principle based algorithms for deep learning," *Journal of Machine Learning Research*, pp. 1–29, 2018. [Online]. Available: https://www.jmlr.org/papers/volume18/17-653/17-653.pdf

[18] Q. Li and S. Hao, "An optimal control approach to deep learning and applications to discrete-weight neural networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2985–2994. [Online]. Available: http://proceedings.mlr.press/v80/li18b/li18b.pdf

[19] J. Han, Q. Li *et al.*, "A mean-field optimal control formulation of deep learning," *Research in the Mathematical Sciences*, vol. 6, no. 1, pp. 1–41, 2019. [Online]. Available: https://link.springer.com/article/10.1007/s40687-018-0172-y

[20] E. Haber and L. Ruthotto, "Stable architectures for deep neural networks," *Inverse problems*, vol. 34, no. 1, p. 014004, 2017. [Online]. Available: https://iopscience.iop.org/article/10.1088/1361-6420/aa9a90/meta

[21] B. Chang, L. Meng, E. Haber, F. Tung, and D. Begert, "Multi-level residual networks from dynamical systems view," *arXiv preprint arXiv:1710.10348*, 2017. [Online]. Available: https://arxiv.org/abs/1710.10348

[22] M. Ciccone, M. Gallieri, J. Masci, C. Osendorfer, and F. Gomez, "Nais-net: Stable deep networks from non-autonomous differential equations," *Advances in Neural Information Processing Systems*, vol. 31, 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/7bd28f15a49d5e5848d6ec70e584e625-Paper.pdf

[23] C. Finlay, J. Calder, B. Abbasi, and A. Oberman, "Lipschitz regularized deep neural networks generalize and are adversarially robust," *arXiv preprint arXiv:1808.09540*, 2018. [Online]. Available: https://arxiv.org/abs/1808.09540

[24] P. Pauli, A. Koch, J. Berberich, P. Kohler, and F. Allgöwer, "Training robust neural networks using lipschitz bounds," *IEEE Control Systems Letters*, vol. 6, pp. 121–126, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9319198

[25] H. Gouk, E. Frank, B. Pfahringer, and M. J. Cree, "Regularisation of neural networks by enforcing lipschitz continuity," *Machine Learning*, vol. 110, no. 2, pp. 393–416, 2021. [Online]. Available: https://link.springer.com/article/10.1007/s10994-020-05929-w

[26] B. Dherin, M. Munn, M. Rosca, and D. G. Barrett, "Why neural networks find simple solutions: the many regularizers of geometric complexity," *arXiv preprint arXiv:2209.13083*, 2022. [Online]. Available: https://arxiv.org/abs/2209.13083

[27] T. Zhou, Q. Li, H. Lu, Q. Cheng, and X. Zhang, "Gan review: Models and medical image fusion applications," *Information Fusion*, vol. 91, pp. 134–148, 2023. [Online]. Available: https://doi.org/10.1016/j.inffus.2022.10.017

[28] C. A. Charu, *Neural networks and deep learning: a textbook*. Spinger, 2018.

[29] S. Theodoridis, "Neural networks and deep learning," *Machine Learning*, pp. 875–936, 2015.

[30] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958. [Online]. Available: https://psycnet.apa.org/record/1959-09865-001

[31] B. Pang, E. Nijkamp, and Y. N. Wu, "Deep learning with tensorflow: A review," *Journal of Educational and Behavioral Statistics*, vol. 45, no. 2, pp. 227–248, 2020. [Online]. Available: https://doi.org/10.3102/1076998619872761

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. [Online]. Available: https://ieeexplore.ieee.org/document/7780459

[33] D. Hunter, H. Yu, M. S. Pukish III, J. Kolbusz, and B. M. Wilamowski, "Selection of proper neural network sizes and architectures—a comparative study," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 2, pp. 228–240, 2012. [Online]. Available: https://ieeexplore.ieee.org/document/6152147

[34] MATLAB, "Statistics and machine learning toolbox." [Online]. Available: https://la.mathworks.com/products/statistics.html

[35] R, "neuralnet: Training of neural networks." [Online]. Available: https://www.rdocumentation.org/packages/neuralnet/versions/1.44.2/topics/neuralnet

[36] TensorFlow, "Tensorflow 2.10.0." [Online]. Available: https://www.tensorflow.org/

[37] J. Reunanen, "Overfitting in making comparisons between variable selection methods," *Journal of Machine Learning Research*, vol. 3, pp. 1371–1382, 2003. [Online]. Available: https://www.jmlr.org/papers/volume3/reunanen03a/reunanen03a.pdf

[38] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, pp. 1157–1182, 2003. [Online]. Available: https://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf

[39] F. Johnson, A. Valderrama, C. Valle, B. Crawford, R. Soto, and R. Ñanculef, "Automating configuration of convolutional neural network hyperparameters using genetic algorithm," *IEEE Access*, vol. 8, pp. 156 139–156 152, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9177040

[40] Y. Zhu, G. Li, R. Wang, S. Tang, H. Su, and K. Cao, "Intelligent fault diagnosis of hydraulic piston pump combining improved lenet-5 and pso hyperparameter optimization," *Applied Acoustics*, vol. 183, p. 108336, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0003682X21004308

[41] A. Gaspar, D. Oliva, E. Cuevas, D. Zaldívar, M. Pérez, and G. Pajares, "Hyperparameter optimization in a convolutional neural network using metaheuristic algorithms," in *Metaheuristics in Machine Learning: Theory and Applications*. Springer, 2021, pp. 37–59. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-70542-8_2

[42] O. S. Steinholtz, "A comparative study of black-box optimization algorithms for tuning of hyper-parameters in deep neural networks," *Luleå University of Technology*, 2018. [Online]. Available: https://ltu.diva-portal.org/smash/get/diva2:1223709/FULLTEXT01.pdf

[43] L. Lugo, "A recurrent neural network approach for whole genome bacteria classification," Master's thesis, Universidad Nacional de Colombia, Bogota, Colombia, 2018.

[44] A. T. Sarmiento and O. Soto, "New product forecasting demand by using neural networks and similar product analysis." Master's thesis, Universidad Nacional de Colombia, Medellin, Colombia, 2014.

[45] A. E. Casas Fajardo, "Propuesta metodológica para calcular el avalúo de un predio empleando redes neuronales artificiales," Master's thesis, Universidad Nacional de Colombia, Bogotá, Colombia, 2014.

[46] S. Ortega Alzate, "Exploración de las redes neuronales para la proyección de la máxima pérdida esperada de una póliza de seguros: aplicación para un seguro previsionales," Master's thesis, Universidad Nacional de Colombia, Medellin, Colombia, 2021.

[47] D. Collazos, "Kernel-based enhancement of general stochastic network for supervised learning," Master's thesis, Universidad Nacional de Colombia, Manizales, Colombia, 2016.

[48] Y. Lu, A. Zhong, Q. Li, and B. Dong, "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3276–3285. [Online]. Available: http://proceedings.mlr.press/v80/lu18d/lu18d.pdf

[49] B. Geshkovski and E. Zuazua, "Turnpike in optimal control of pdes, resnets, and beyond," *Acta Numerica*, vol. 31, pp. 135–263, 2022. [Online]. Available: https://doi.org/10.1017/S0962492922000046

[50] D. Ruiz-Balet and E. Zuazua, "Neural ode control for classification, approximation and transport," *arXiv preprint arXiv:2104.05278*, 2021. [Online]. Available: https://arxiv.org/abs/2104.05278

[51] D. Ruiz-Balet, E. Affili, and E. Zuazua, "Interpolation and approximation via momentum resnets and neural odes," *Systems & Control Letters*, vol. 162, p. 105182, 2022. [Online]. Available: https://doi.org/10.1016/j.sysconle.2022.105182

[52] M. Fazlyab, A. Robey, H. Hassani, M. Morari, and G. Pappas, "Efficient and accurate estimation of lipschitz constants for deep neural networks," *Advances in Neural*

Information Processing Systems, vol. 32, 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/95e1533eb1b20a97777749fb94fdb944-Paper.pdf

[53] A. Xue, L. Lindemann, A. Robey, H. Hassani, G. J. Pappas, and R. Alur, "Chordal sparsity for lipschitz constant estimation of deep neural networks," *arXiv preprint arXiv:2204.00846*, 2022. [Online]. Available: https://arxiv.org/abs/2204.00846

[54] L. Massidda, M. Marrocu, and S. Manca, "Non-intrusive load disaggregation by convolutional neural network and multilabel classification," *Applied Sciences*, vol. 10, no. 4, p. 1454, 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/4/1454

[55] K. Muralitharan, R. Sakthivel, and R. Vishnuvarthan, "Neural network based optimization approach for energy demand prediction in smart grid," *Neurocomputing*, vol. 273, pp. 199–208, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0925231217313681

[56] R. Lu and S. H. Hong, "Incentive-based demand response for smart grid with reinforcement learning and deep neural network," *Applied energy*, vol. 236, pp. 937–949, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0306261918318798

[57] A. P. Marugán, F. P. G. Márquez, J. M. P. Perez, and D. Ruiz-Hernández, "A survey of artificial neural network in wind energy systems," *Applied energy*, vol. 228, pp. 1822–1836, 2018. [Online]. Available: https://doi.org/10.1016/j.apenergy.2018.07.084

[58] F. Saeed, M. A. Khan, M. Sharif, M. Mittal, L. M. Goyal, and S. Roy, "Deep neural network features fusion and selection based on pls regression with an application for crops diseases classification," *Applied Soft Computing*, vol. 103, p. 107164, 2021. [Online]. Available: https://doi.org/10.1016/j.asoc.2021.107164

[59] M. Loey, A. ElSawy, and M. Afify, "Deep learning in plant diseases detection for agricultural crops: A survey," *International Journal of Service Science, Management, Engineering, and Technology (IJSSMET)*, vol. 11, no. 2, pp. 41–58, 2020. [Online]. Available: https://www.igi-global.com/article/deep-learning-in-plant-diseases-detection-for-agricultural-crops/248499

[60] B. Pandey, D. K. Pandey, B. P. Mishra, and W. Rhmann, "A comprehensive survey of deep learning in the field of medical imaging and medical natural language processing: Challenges and research directions," *Journal of King Saud University-Computer and Information Sciences*, 2021.

[61] A. Nogales, A. J. Garcia-Tejedor, D. Monge, J. S. Vara, and C. Antón, "A survey of deep learning models in medical therapeutic areas," *Artificial*

Intelligence in Medicine, vol. 112, p. 102020, 2021. [Online]. Available: https://doi.org/10.1016/j.artmed.2021.102020

[62] Colciencias, "Plan Nacional de CTeI para el desarrollo del sector Tecnologías de la Información TIC 2017 - 2022,," Bogotá, Colombia, 2017.

[63] República de Colombia, "Plan de Desarrollo Nacional 2018-2022 "Pacto por Colombia, pacto por la equidad"," Bogotá, Colombia, 2018.

[64] Colciencias, "Política Nacional de Ciencia e Innovación para el Desarrollo Sostenible Libro Verde 2030," Bogotá, Colombia, 2018.

[65] J. C. Riaño Rojas, "Desarrollo de una metodología como soporte para la detección de enfermedades vasculares del tejido conectivo a través de imágenes capilaroscópicas," Ph.D. dissertation, Universidad Nacional de Colombia, Bogotá, Colombia, 2010.

[66] T. T. Tang, J. A. Zawaski, K. N. Francis, A. A. Qutub, and M. W. Gaber, "Image-based classification of tumor type and growth rate using machine learning: a preclinical study," *Scientific reports*, vol. 9, no. 1, pp. 1–10, 2019. [Online]. Available: https://www.nature.com/articles/s41598-019-48738-5

[67] C. A. Pedraza Bonilla and L. Rodríguez Mújica, "Método para la estimación de maleza en cultivos de lechuga utilizando aprendizaje profundo e imágenes multiespectrales," Master's thesis, Universidad Nacional de Colombia., Bogotá, Colombia, 2016.

[68] C. Barrios Perez, "Zonificación agroecológica para el cultivo de arroz de riego (*Oryza Sativa L.*) en colombia," Master's thesis, Universidad Católica de Colombia., Palmira, Colombia, 2016.

[69] A. F. Montenegro and C. D. Parada, "Diseño e implementación de un sistema de detección de malezas en cultivos cundiboyacenses," Master's thesis, Universidad Católica de Colombia., Bogotá, Colombia, 2015.

[70] M. Minsky and S. Papert, "An introduction to computational geometry," *Cambridge tiass., HIT*, vol. 479, p. 480, 1969.

[71] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989. [Online]. Available: https://link.springer.com/article/10.1007/BF02551274

[72] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986. [Online]. Available: https://www.nature.com/articles/323533a0

[73] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: LSTM cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8737887

[74] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, "Neural speech synthesis with transformer network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 6706–6713.

[75] G. Parascandolo, H. Huttunen, and T. Virtanen, "Taming the waves: sine as activation function in deep neural networks," 2017. [Online]. Available: https://openreview.net/forum?id=Sks3zF9eg

[76] J. Heredia-Juesas and J. Á. Martínez-Lorenzo, "Consensus function from an $\mathcal{L}_p^q-$ norm regularization term for its use as adaptive activation functions in neural networks," *arXiv e-prints*, pp. arXiv–2206, 2022. [Online]. Available: https://arxiv.org/abs/2206.15017

[77] A. D. Jagtap, Y. Shin, K. Kawaguchi, and G. E. Karniadakis, "Deep kronecker neural networks: A general framework for neural networks with adaptive activation functions," *Neurocomputing*, vol. 468, pp. 165–180, 2022. [Online]. Available: https://doi.org/10.1016/j.neucom.2021.10.036

[78] D. Devikanniga, K. Vetrivel, and N. Badrinath, "Review of meta-heuristic optimization based artificial neural networks and its applications," in *Journal of Physics: Conference Series*, vol. 1362, no. 1. IOP Publishing, 2019, p. 012074. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/1362/1/012074/meta

[79] N. Gupta, M. Khosravy, N. Patel, S. Gupta, and G. Varshney, "Evolutionary artificial neural networks: comparative study on state-of-the-art optimizers," in *Frontier applications of nature inspired computation*. Springer, 2020, pp. 302–318. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-15-2133-1_14

[80] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization." *Journal of machine learning research*, vol. 12, no. 7, 2011. [Online]. Available: https://jmlr.org/papers/volume12/duchi11a/duchi11a.pdf

[81] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[82] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012. [Online]. Available: https://arxiv.org/abs/1212.5701

[83] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016. [Online]. Available: https://arxiv.org/abs/1609.04747

[84] P. Netrapalli, "Stochastic gradient descent and its variants in machine learning," *Journal of the Indian Institute of Science*, vol. 99, no. 2, pp. 201–213, 2019. [Online]. Available: https://link.springer.com/article/10.1007/s41745-019-0098-4

[85] S. Lawrence, C. L. Giles, and A. C. Tsoi, "Lessons in neural network training: Overfitting may be harder than expected," in *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 1997, pp. 540–545. [Online]. Available: https://clgiles.ist.psu.edu/papers/AAAI-97.overfitting.hard_to_do.pdf

[86] X.-x. Wu and J.-g. Liu, "A new early stopping algorithm for improving neural network generalization," in *2009 Second International Conference on Intelligent Computation Technology and Automation*, vol. 1. IEEE, 2009, pp. 15–18. [Online]. Available: https://ieeexplore.ieee.org/document/5287721

[87] H. Liang, S. Zhang, J. Sun, X. He, W. Huang, K. Zhuang, and Z. Li, "Darts+: Improved differentiable architecture search with early stopping," *arXiv preprint arXiv:1909.06035*, 2019. [Online]. Available: https://arxiv.org/abs/1909.06035

[88] L. Prechelt, "Early stopping-but when?" in *Neural Networks: Tricks of the trade*. Springer, 1998, pp. 55–69. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-35289-8_5

[89] M. Mahsereci, L. Balles, C. Lassner, and P. Hennig, "Early stopping without a validation set," *arXiv preprint arXiv:1703.09580*, 2017. [Online]. Available: https://arxiv.org/abs/1703.09580

[90] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019. [Online]. Available: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0

[91] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *2018 International Interdisciplinary PhD Workshop (IIPhDW)*. IEEE, 2018, pp. 117–122. [Online]. Available: https://ieeexplore.ieee.org/document/8388338

[92] K. el Hindi and A.-A. Mousa, "Smoothing decision boundaries to avoid overfitting in neural network training," *Neural Network World*, vol. 21, no. 4, p. 311, 2011. [Online]. Available: https://www.researchgate.net/publication/272237391_Smoothing_decision_boundaries_to_avoid_overfitting_in_neural_network_training

[93] H. Jabbar and R. Z. Khan, "Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study)," *Computer Science, Communication and Instrumentation Devices*, vol. 70, 2015.

[94] K.-j. Kim, "Artificial neural networks with evolutionary instance selection for financial forecasting," *Expert Systems with Applications*, vol. 30, no. 3, pp. 519–526, 2006. [Online]. Available: https://doi.org/10.1016/j.eswa.2005.10.007

[95] N. Srivastava, "Improving neural networks with dropout," Master's thesis, University of Toronto, 2013. [Online]. Available: http://www.cs.toronto.edu/~nitish/msc_thesis.pdf

[96] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014. [Online]. Available: https://jmlr.org/papers/v15/srivastava14a.html

[97] J. Ba and B. Frey, "Adaptive dropout for training deep neural networks," *Advances in neural information processing systems*, vol. 26, 2013. [Online]. Available: https://proceedings.neurips.cc/paper/2013/file/7b5b23f4aadf9513306bcd59afb6e4c9-Paper.pdf

[98] B. Ko, H.-G. Kim, K.-J. Oh, and H.-J. Choi, "Controlled dropout: A different approach to using dropout on deep neural network," in *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*. IEEE, 2017, pp. 358–362. [Online]. Available: https://ieeexplore.ieee.org/document/7881693

[99] D. Molchanov, A. Ashukha, and D. Vetrov, "Variational dropout sparsifies deep neural networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2498–2507. [Online]. Available: https://arxiv.org/abs/1701.05369

[100] G. Zhang, C. Wang, B. Xu, and R. Grosse, "Three mechanisms of weight decay regularization," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://www.researchgate.net/publication/328598833_Three_Mechanisms_of_Weight_Decay_Regularization

[101] S. J. Nowlan and G. E. Hinton, "Simplifying neural networks by soft weight sharing," in *The Mathematics of Generalization*. CRC Press, 2018, pp. 373–394. [Online]. Available: https://ieeexplore.ieee.org/document/6796174

[102] R. Ghosh and M. Motani, "Network-to-network regularization: Enforcing occam's razor to improve generalization," *Advances in Neural Information Processing Systems*, vol. 34, pp. 6341–6352, 2021. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/321cf86b4c9f5ddd04881a44067c2a5a-Paper.pdf

[103] B. Neal, S. Mittal, A. Baratin, V. Tantia, M. Scicluna, S. Lacoste-Julien, and I. Mitliagkas, "A modern take on the bias-variance tradeoff in neural

networks," *arXiv preprint arXiv:1810.08591*, 2018. [Online]. Available: https://arxiv.org/abs/1810.08591

[104] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever, "Deep double descent: Where bigger models and more data hurt," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2021, no. 12, p. 124003, 2021. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-5468/ac3a74/meta

[105] Z. Yang, Y. Yu, C. You, J. Steinhardt, and Y. Ma, "Rethinking bias-variance trade-off for generalization of neural networks," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 767–10 777. [Online]. Available: http://proceedings.mlr.press/v119/yang20j/yang20j.pdf

[106] Y. Dar, V. Muthukumar, and R. G. Baraniuk, "A farewell to the bias-variance tradeoff? an overview of the theory of overparameterized machine learning," *arXiv preprint arXiv:2109.02355*, 2021. [Online]. Available: https://arxiv.org/abs/2109.02355

[107] B. Ghojogh and M. Crowley, "The theory behind overfitting, cross validation, regularization, bagging, and boosting: tutorial," *arXiv preprint arXiv:1905.12787*, 2019. [Online]. Available: https://arxiv.org/abs/1905.12787

[108] Y. Yoshida and T. Miyato, "Spectral norm regularization for improving the generalizability of deep learning," *arXiv preprint arXiv:1705.10941*, 2017. [Online]. Available: https://arxiv.org/abs/1705.10941

[109] Y. Tsuzuku, I. Sato, and M. Sugiyama, "Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks," *Advances in neural information processing systems*, vol. 31, 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/485843481a7edacbfce101ecb1e4d2a8-Paper.pdf

[110] H. Li, J. Li, X. Guan, B. Liang, Y. Lai, and X. Luo, "Research on overfitting of deep learning," in *2019 15th International Conference on Computational Intelligence and Security (CIS)*. IEEE, 2019, pp. 78–81. [Online]. Available: https://ieeexplore.ieee.org/document/9023664

[111] A. Gavrilov, A. Jordache, M. Vasdani, and J. Deng, "Convolutional neural networks: Estimating relations in the ising model on overfitting," in *2018 IEEE 17th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*. IEEE, 2018, pp. 154–158. [Online]. Available: https://ieeexplore.ieee.org/document/8482067

[112] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012. [Online]. Available: https://ieeexplore.ieee.org/document/6296535

[113] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017. [Online]. Available: https://arxiv.org/abs/1708.07747

[114] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *International conference on learning and intelligent optimization*. Springer, 2011, pp. 507–523. [Online]. Available: https://ml.informatik.uni-freiburg.de/wp-content/uploads/papers/11-LION5-SMAC.pdf

[115] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, M. Birattari, and T. Stützle, "The irace package: Iterated racing for automatic algorithm configuration," *Operations Research Perspectives*, vol. 3, pp. 43–58, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214716015300270

[116] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[117] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[118] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015. [Online]. Available: https://arxiv.org/abs/1412.6572

[119] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Artificial intelligence safety and security*. Chapman and Hall/CRC, 2018, pp. 99–112. [Online]. Available: https://openreview.net/pdf?id=S1OufnIlx

[120] Z. Chen, Q. Li, and Z. Zhang, "Towards robust neural networks via close-loop control," *arXiv preprint arXiv:2102.01862*, 2021. [Online]. Available: https://arxiv.org/abs/2102.01862

[121] L. Böttcher, N. Antulov-Fantulin, and T. Asikis, "AI Pontryagin or how artificial neural networks learn to control dynamical systems," *Nature Communications*, vol. 13, no. 1, pp. 1–9, 2022. [Online]. Available: https://www.nature.com/articles/s41467-021-27590-0

[122] J. Zhuang, N. C. Dvornek, S. Tatikonda, and J. S. Duncan, "MALI: A memory efficient and reverse accurate integrator for neural ODEs," *arXiv preprint arXiv:2102.04668*, 2021. [Online]. Available: https://arxiv.org/abs/2102.04668

[123] C. Rackauckas, M. Innes, Y. Ma, J. Bettencourt, L. White, and V. Dixit, "Diffeqflux. jl-a julia library for neural differential equations," *arXiv preprint arXiv:1902.02376*, 2019. [Online]. Available: https://arxiv.org/abs/1902.02376

[124] D. M. Grobman, "Homeomorphism of systems of differential equations," *Doklady Akademii Nauk SSSR*, vol. 128, no. 5, pp. 880–881, 1959.

[125] P. Hartman, "A lemma in the theory of structural stability of differential equations," *Proceedings of the American Mathematical Society*, vol. 11, no. 4, pp. 610–620, 1960. [Online]. Available: https://www.ams.org/journals/proc/1960-011-04/S0002-9939-1960-0121542-7/S0002-9939-1960-0121542-7.pdf

[126] Ministerio de ciencia, tecnología e innovación. Minciencias, "Guía técnica para el reconocimiento de actores del SNCTeI," 2021. [Online]. Available: https://minciencias.gov.co/sites/default/files/upload/reconocimiento/m601pr05g07_guia_tecnica_para_el_reconocimiento_del_centro_de_desarrollo_tecnologico_-_cdt_v00_0.pdf