UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO - CHILE

“ESTRATEGIAS DIVIDE Y VENCERÁS PARA ENTRENAMIENTO DE REDES NEURONALES EN PRESENCIA DE MÚLTIPLES CLASES”

GABRIEL ANDRES MOLINA BARRA

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Ricardo Ñanculef

Octubre - 2019

DEDICATORIA

A mi Abuelita Meme,
quien es mi apoyo mas grande y la amo mucho.

AGRADECIMIENTOS

- A mis padres, por todo el apoyo y las buenas energías que me dieron durante tantos años, por lo cual este trabajo es tanto suyo como mio.

- A mi hermana, por ser siempre la voz de la razón y la lógica en mi hogar.

- A mi abuela, por apoyarme incondicionalmente en todo y desearme lo mejor. v

- A mi tía, por ser tan amable y apoyarme sin importar los problemas.

- A todos mis amigos y compañeros que conocí en la Universidad. Roberto, Koni, Fabián, Carlos, Rafael, Miguel, Javier, Jorje, Ignacio, Nano, Camilo y a todos los que faltan por temas de espacio, ustedes hicieron estos casi 7 años los mejores, en especial Cristóbal un gran amigo y me aguanto como compañero de trabajos en muchos ramos, sin ti no creo haber podido terminar la carrera.

- A Francisco y Margarita, grandes amigos y los que me motivaron a seguir aprendiendo más y seguir perfeccionándome.

- A Andres y a Fabián, que son grandes amigos en la universidad y fuera de ella.

- Al profesor José Luis, quien considero un excelente profesor y una gran persona, siempre dispuesto a ayudar a quien lo necesite.

- Por ultimo, quiero agradecer a mi profesor guía Ricardo Ñanculef, quien me acompaño durante el desarrollo de todo este trabajo y me mostró lo bello que es la investigación y la informática.

A todos ustedes, ¡Muchas Gracias!

RESUMEN

Actualmente las redes neuronales artificiales se han convertido en una de las metodologías más utilizadas para abordar problemas de aprendizaje en inteligencia artificial, lo que impulsa a crear modelos y arquitecturas cada vez más complejas para resolver distintos problemas. En esta memoria proponemos un nuevo método para el entrenamiento de redes neuronales en problemas de clasificación con múltiples categorías. En estos problemas, una red implementa una función de decisión que se utiliza para determinar la clase a la que pertenece un determinado dato de entrada (imagen, texto, etc). Cuando se tienen muchas categorías posibles, la metodología convencional consiste en aprender una distribución de probabilidad sobre todas las clases simultáneamente, usando una función objetivo que penaliza errores en todas las clases. Pero la efectividad de dichas redes no es simplemente observar el promedio del accuracy o la función de pérdida, si no que más bien se tiene que observar los resultados obtenidos por cada clase, ya que no se puede tener un accuracy alto en un par de clases y uno muy bajo en el resto de ellas. La idea que exploraremos en esta memoria consiste en descomponer este problema de aprendizaje en varias partes cuyas soluciones, combinadas apropiadamente, den una mejor solución a algunos problemas donde se ha superado incluso la habilidad humana para reconocer imágenes, como por ejemplo la clasificación de imágenes en el dataset CIFAR10 donde se ha alcanzado un accuracy del 90.82 %.

Palabras Claves— Accuracy, Descomposición, Redes Binarias, GAD, Arquitecturas

ABSTRACT

Currently artificial neural networks have become one of the most used methodologies to address learning problems in artificial intelligence, which drives to create increasingly complex models and architectures to solve different problems. In this report we propose a new method for the training of neural networks in classification problems with multiple categories. In these problems, a network implements a decision function that is used to determine the class to which a certain input data belongs (image, text, etc). When there are many possible categories, the conventional methodology consists of learning a probability distribution over all classes simultaneously, using an objective function that penalizes errors in all classes. But the result of the effectiveness of such networks is not simply to observe the average accuracy or loss function, but rather to observe the results obtained by each

class, since you can not have a high accuracy in a pair of classes and one very low in the rest of them. The idea that we will explore in this memory is to break down this learning problem into several parts whose solutions, properly combined, give a better solution to some problems where even the human ability to recognize images has been overcome, such as the classification of images in the CIFAR10 dataset where an accuracy of 90.82 % has been reached.

Keywords— Accuracy, Decomposition, Binary Neural Networks, GAD, Architectures

GLOSARIO

- **Accuracy**: Métrica para evaluar los modelos de clasificación. Informalmente, la precisión es la fracción de predicciones que nuestro modelo acertó.
- **Benchmark**: Estándar, o un conjunto de estándares, utilizado como punto de referencia para evaluar el rendimiento o el nivel de calidad.
- **Cross Validation**: En el área del aprendizaje automático, La validación cruzada o cross-validation es una técnica utilizada para evaluar el desempeño de la red y garantizar que es independientes de la partición entre datos de entrenamiento y prueba usados.
- **Data Augmentation**: El aumento de datos o data Augmentation es una forma de crear nuevos "datos" con diferentes rotaciones, ruido o cualquier cambio prudente que se pueda aplicar. Los beneficios de esto son dos: la primera es la capacidad de generar "más datos.ª partir de datos limitados y, en segundo lugar, evita el overfitting.
- **Deep Learning**: Deep Learning es uno de los métodos de aprendizaje de la inteligencia artificial, y a día de hoy pertenece a un subcampo de las maquinas de aprendizaje automatico. Este se define como un algoritmo automático estructurado o jerárquico que emula el aprendizaje humano con el fin de obtener ciertos conocimientos
- **Global Contrast Normalization (GCN)**: Técnica de pre-procesamiento en la que, opcionalmente, se resta el promedio por columnas de un conjunto de datos. Luego, este es dividido por la desviación estándar por filas del mismo. En el caso de imágenes multicanal, se emplea el procedimiento anterior en cada canal por separado.
- **Local Response Normalization (LRN)**:Procedimiento que normaliza la salida de una capa convolucional en caso de que la función de activación sea ReLU.
- **Overfitting**: Overfitting o Sobreajuste en español, es el efecto de sobreentrenar un algoritmo de aprendizaje con unos ciertos datos para los que se conoce el resultado deseado. El algoritmo de aprendizaje debe alcanzar un estado en el que será capaz de predecir el resultado en otros casos a partir de lo aprendido con los datos de entrenamiento, generalizando para poder resolver situaciones distintas a las acaecidas durante el entrenamiento.
- **One Against One (OAO)**: Método de descomposición en clasificación. Se crear $\frac{K(K-1)}{2}$ clasificadores en los cuales cada uno de ellos evalúa todo el dataset, para luego realizar un ejercicio de mayoría de votos y seleccionar a que clase pertenece cada dato de entrada.
- **State of the Art (SOTA)**: En el ámbito de la investigación científica, el SOTA (por sus siglas en inglés) hace referencia al estado último de la materia en términos de I+D, refiriéndose incluso al límite de conocimiento humano público sobre la materia.

ÍNDICE DE CONTENIDOS

RESUMEN IV

ABSTRACT IV

GLOSARIO VI

ÍNDICE DE FIGURAS IX

ÍNDICE DE TABLAS XII

INTRODUCCIÓN 1

CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA 2
- 1.1 Problema a estudiar 2
- 1.2 Objetivos 3
  - 1.2.1 Objetivo General 3
  - 1.2.2 Metas Especificar a cumplir 3

CAPÍTULO 2: MARCO CONCEPTUAL 4
- 2.1 ¿Qué son las redes neuronales artificiales? 4
- 2.2 Componentes de una red neuronal artificial 5
  - 2.2.1 Neurona 5
  - 2.2.2 Función de activación σ 5
  - 2.2.3 Tipos de redes neuronales artificiales 6
  - 2.2.4 Redes Feedforward 7
  - 2.2.5 Entrenamiento 8
  - 2.2.6 Redes Convolucionales 9
- 2.3 Estado del Arte 11
  - 2.3.1 Metodología DAG 11
  - 2.3.2 Estudio de metodologías actuales 12
  - 2.3.3 Datasets desbalanceados 20

CAPÍTULO 3: PROPUESTA DE SOLUCIÓN 28

CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN 29
- 4.1 Datasets 29
- 4.2 Arquitecturas y Benchmark 30
  - 4.2.1 MNIST 31
  - 4.2.2 Arquitectura Maestra 31
  - 4.2.3 Arquitectura Básica 35
  - 4.2.4 NORB 39
  - 4.2.5 Arquitectura Maestra 40

4.2.6 Arquitectura Básica 44
4.2.7 CIFAR-10 48
4.2.8 Arquitectura Maestra 48
4.2.9 Arquitectura Básica 52
4.3 Unbalanced CIFAR-10 57
4.3.1 Benchmark Arq. Maestra 59
4.3.2 Benchmark Arq. Básica 61
4.4 BAAT 64
4.5 Análisis Experimental de la Propuesta 69
4.5.1 MNIST 70
4.5.2 NORB 73
4.5.3 CIFAR-10 76
4.5.4 Unbalanced CIFAR-10 79
4.5.5 BAAT 82

CAPÍTULO 5: CONCLUSIONES 85
5.1 MNIST 85
5.2 NORB 86
5.3 CIFAR-10 86
5.4 Unbalanced CIFAR-10 87
5.5 BAAT 88
5.6 Conclusiones Generales 88
5.7 Trabajos Futuros 90

REFERENCIAS BIBLIOGRÁFICAS 91

ÍNDICE DE FIGURAS

1 Modelo básico de una red neuronal artificial. 4

2 Modelo de neurona con función de activación φ. 6

3 Modelo de Red Feedforward. 7

4 Modelo básico del backpropagation. 9

5 Esquema de como actúa una capa de pooling sobre una imagen. 10

6 Esquema completo de una red convolucional. 11

7 Visualización de la metodología original DAG donde las clases están representadas como 1,2,3,4. 12

8 Frontera de clasificación dibujadas por 6 ANN binarias con metodología OAA. . 14

9 Ejemplo de sistema clasificador para K clases usando la metodología OAHO. . . 18

10 Frontera de clasificación ideales generadas por una sola red clasificadora para K clases. Siendo esta la clasificación OAO de una red con múltiples neuronas de salida. 20

11 Matriz de confusión de las clases de MNIST para la arquitectura maestra sin descomposición. 32

12 Matriz de confusión de las clases de MNIST para la arquitectura maestra con descomposición en cadena. 33

13 Matriz de confusión de las clases de MNIST para la arquitectura maestra con descomposición OAO con mayoría de votos. 35

14 Matriz de confusión de las clases de MNIST para la arquitectura básica sin descomposición. 37

15 Matriz de confusión de las clases de MNIST para la arquitectura básica con descomposición en cadena. 38

16 Matriz de confusión de las clases de MNIST para la arquitectura básica con descomposición OAO con mayoría de votos. 39

17 Matriz de confusión de las clases de NORB para la arquitectura maestra sin descomposición. 41

18 Matriz de confusión de las clases de NORB para la arquitectura maestra con descomposición en cadena. 42

19 Matriz de confusión de las clases de NORB para la arquitectura maestra con descomposición OAO con mayoría de votos. 44

20 Matriz de confusión de las clases de NORB para la arquitectura básica sin descomposición. 46

21 Matriz de confusión de las clases de NORB para la arquitectura básica con descomposición en cadena. 47

22 Matriz de confusión de las clases de NORB para la arquitectura básica con descomposición OAO con mayoría de votos. 48

23 Matriz de confusión de las clases de CIFAR10 para la arquitectura maestra sin descomposición. 50

24 Matriz de confusión de las clases de CIFAR-10 para la arquitectura maestra con descomposición en cadena. 51

25 Matriz de confusión de las clases de CIFAR-10 para la arquitectura maestra con des-composición OAO con mayoría de votos. 52

26 Matriz de confusión de las clases de CIFAR10 para la arquitectura básica sin descomposición. 54

27 Matriz de confusión de las clases de CIFAR-10 para la arquitectura básica con descomposición en cadena. 55

28 Matriz de confusión de las clases de CIFAR-10 para la arquitectura básica con descomposición OAO con mayoría de votos. 57

29 Matriz de confusión de las clases de Unbalanced CIFAR-10 para la arquitectura maestra sin descomposición. 59

30 Matriz de confusión de las clases de Unbalanced CIFAR-10 para la arquitectura maestra con descomposición en cadena. 60

31 Matriz de confusión de las clases de Unbalanced CIFAR-10 para la arquitectura maestra con des-composición OAO con mayoría de votos. 61

32 Matriz de confusión de las clases de Unbalanced CIFAR-10 para la arquitectura básica sin descomposición. 62

33 Matriz de confusión de las clases de Unbalanced CIFAR-10 para la arquitectura básica con descomposición en cadena. 63

34 Matriz de confusión de las clases de Unbalanced CIFAR-10 para la arquitectura básica con des-composición OAO con mayoría de votos. 64

35 Matriz de confusión de las clases de BAAT para su arquitectura base sin descomposición. 67

36 Matriz de confusión de las clases de BAAT para su arquitectura base con descomposición en cadena. 68

37 Matriz de confusión de las clases de BAAT para la arquitectura base con des-composición OAO con mayoría de votos. 69

38 Matriz de confusión de los resultados del experimento 1 sobre el conjunto de prueba de MNIST. 71

39 Matriz de confusión de los resultados del experimento 2 sobre el conjunto de prueba de MNIST. 72

40 Matriz de confusión de los resultados del experimento 3 sobre el conjunto de prueba de MNIST. 73

41 Matriz de confusión de los resultados del experimento 1 sobre el conjunto de prueba de NORB. 74

42 Matriz de confusión de los resultados del experimento 2 sobre el conjunto de prueba de NORB. 75

43 Matriz de confusión de los resultados del experimento 3 sobre el conjunto de prueba de NORB. 76

44 Matriz de confusión de los resultados del experimento 1 sobre el conjunto de prueba de CIFAR-10. 77

45 Matriz de confusión de los resultados del experimento 2 sobre el conjunto de prueba de CIFAR-10. 78

46 Matriz de confusión de los resultados del experimento 3 sobre el conjunto de prueba de CIFAR-10. 79

47 Matriz de confusión de los resultados del experimento 2 sobre el conjunto de prueba de Unbalanced CIFAR-10. 81

48 Matriz de confusión de los resultados del experimento 3 sobre el conjunto de prueba de Unbalanced CIFAR-10. 82

49 Matriz de confusión de las clases de BAAT para su arquitectura base con descomposición GAD entrenada con solo Fine Tunning. 83

50 Matriz de confusión de las clases de BAAT para su arquitectura base con descomposición GAD entrenada mediante Fine Tunning con capas congeladas. 84

ÍNDICE DE TABLAS

1 Distribución de datos de entrenamiento en el dataset “Glass”. 22

2 Resumen de precisión en “Glass”. 22

3 Distribución de datos de entrenamiento y prueba en el dataset “Shuttle”. . . . 23

4 Resumen de precisión en “Shuttle”. 23

5 Rendimiento del dataset MNIST en distintos sistemas de redes neuronales. . . 24

6 Distribución total de datos por clase del dataset Letter. 25

7 Rendimiento el dataset “Letter”. en distintos sistemas de redes neuronales. . . 26

8 Precisión de clasificación de cuatro redes neuronales sobre los pequeños datasets “Iris”, “Wine” y “Vehicle”. 26

9 Accuracy de clasificación de cuatro redes neuronales sobre el gran dataset “PDB”. 27

10 Distribución de data de BAAT. 30

11 Especificaciones importantes de dataset a utilizar para la validación de la memoria. 30

12 MNIST Arquitectura maestra, red DNN-W20. 31

13 Accuracy por clase para el dataset MNIST usando la arquitectura maestra sin descomposición. 32

14 Benchmark MNIST arquitectura maestra con descomposición en cadena. . . . 33

15 Benchmark MNIST para la arquitectura maestra con descomposición OAO con mayoría de votos. 34

16 MNIST arquitectura basica, red DNN-W20. 36

17 Accuracy por clase para el dataset MNIST usando la arquitectura básica sin descomposición. 36

18 Benchmark MNIST para la arquitectura básica con descomposición en cadena. . 37

19 Benchmark MNIST para la arquitectura básica con descomposición OAO con mayoría de votos. 38

20 NORB Arquitectura maestra, AP-Relu. 40

21 Accuracy por clase para el dataset NORB usando la arquitectura maestra sin descomposición. 41

22 Benchmark NORB para la arquitectura maestra con descomposición en cadena. 42

23 Benchmark NORB para la arquitectura maestra con descomposición OAO con mayoría de votos. 43

24 NORB Arquitectura básica, MP-Relu. 45

25 Accuracy por clase para el dataset NORB usando la arquitectura básica sin descomposición. 45

26 Benchmark NORB para la arquitectura básica con descomposición en cadena. . 46

27 Benchmark NORB para la arquitectura básica con descomposición OAO con mayoría de votos. 47

28 CIFAR-10 arquitectura maestra, red All-CNN-C. 49

29 Accuracy por clase para el dataset CIFAR10 usando la arquitectura maestra sin descomposición. 50

30 Benchmark CIFAR-10 para la arquitectura maestra con descomposición en cadena. 51

31 Benchmark CIFAR-10 para la arquitectura maestra con descomposición OAO con mayoría de votos. 52

32 CIFAR-10 arquitectura basica, red CNN+Relu. 53

33 Accuracy por clase para el dataset CIFAR10 usando la arquitectura básica sin descomposición. 54

34 Benchmark CIFAR-10 para la arquitectura básica con descomposición en cadena. 55

35 Benchmark CIFAR-10 para la arquitectura básica con descomposición OAO con mayoría de votos. 56

36 Accuracy por clase de la arquitectura maestra y básica de CIFAR-10 sin descomposición. 58

37 Datos de entrenamiento y Prueba por clase para el dataset CIFAR10 desbalanceado. 58

38 Accuracy por clase para el dataset Unbalanced CIFAR-10 usando la arquitectura maestra sin descomposición. 59

39 Accuracy por clase para el dataset Unbalanced CIFAR-10 usando la arquitectura maestra con descomposición en cadena. 60

40 Benchmark Unbalanced CIFAR-10 para la arquitectura maestra con descomposición OAO con mayoría de votos. 61

41 Accuracy por clase para el dataset Unbalanced CIFAR-10 usando la arquitectura básica sin descomposición. 62

42 Accuracy por clase para el dataset Unbalanced CIFAR-10 usando la arquitectura básica con descomposición en cadena. 63

43 Benchmark Unbalanced CIFAR-10 para la arquitectura básica con descomposición OAO con mayoría de votos. 64

44 Arquitectura Convolutional-MaxPooling para el dataset BAAT 65

45 Distribución de datos de entrenamiento y prueba después de la división de datos en BAAT. 66

46 Accuracy por clase para el dataset BAAT sin descomposición. 66

47 Accuracy por clase para el dataset BAAT con descomposición en cadena. 67

48 Benchmark BAAT para la arquitectura base con descomposición OAO con mayoría de votos. 68

49 Resultados experimento 1 sobre el dataset MNIST. 70

50 Resultados experimento 2 sobre el dataset MNIST. 71

51 Resultados experimento 3 sobre el dataset MNIST. 72

52 Resultados experimento 1 sobre el dataset NORB. 74

53 Resultados experimento 2 sobre el dataset NORB. 75

54 Resultados experimento 3 sobre el dataset NORB. 76

55 Resultados experimento 1 sobre el dataset CIFAR-10. 77

56 Resultados experimento 2 sobre el dataset CIFAR-10. 78

57 Resultados experimento 3 sobre el dataset CIFAR-10. 79

58 Resultados experimento 2 sobre el dataset Unbalanced Cifar-10. 80

59 Resultados experimento 3 sobre el dataset Unbalanced Cifar-10. 81

60 Resultados experimento 2 sobre el dataset BAAT. 83

61 Resultados experimento 3 sobre el dataset BAAT. 84

INTRODUCCIÓN

Debido al incremento exponencial de la cantidad de datos que producimos y a las mejoras en capacidad computacional de los últimos años, el área del aprendizaje automático, especialmente la subárea de las redes neuronales (o ANN por sus siglas en inglés) ha llamado la atención por su capacidad de solucionar problemas con precisión cada vez mayor, superando en algunos casos la de un ser humano, y volviéndose una herramienta cada vez más utilizada para solucionar problemas como la detección de voz [1], clasificación de imágenes [2] e incluso diagnósticos automáticos en el área de la salud [3].

Las ANN son exitosas y bien usadas en la actualidad gracias a su capacidad de aprender automáticamente una tarea a partir de ejemplos.

Una de las tareas mas típicas de las ANN es la clasificación, donde la red se entrena para discernir la clase o categoría a la que pertenece un determinado objeto de entrada. En el mundo de las imágenes las redes que mejor desarrollan esta actividad son las redes neuronales convolucionales (o CNN por sus siglas en inglés).

Una CNN puede clasificar una cantidad de clases dependiendo de como es construida ya que la red aprende una distribución de probabilidad sobre todas las clases simultáneamente, usando una función objetivo que penaliza errores. Esto genera que la red identifique ciertos patrones que permite generar la clasificación correcta.

Normalmente, para identificar la calidad de la red construida, uno utiliza ciertas métricas, por ejemplo, podemos utilizar accuracy promedio por clase o el promedio de la función de pérdida de la red. El problema con estas métricas es que se puede tener un promedio alto pero si analizamos individualmente cada clase, la red tiende a ignorar algunas y enfocarse en otras, generando un problema de confianza, ya que las métricas no nos demuestran explícitamente si la red logro aprender a clasificar todas las clases por igual o solamente se enfoco en algunas.

La idea que exploraremos en esta memoria consiste en descomponer un problema de clasificación múltiple en varias partes cuyas soluciones, combinadas apropiadamente, den una mejor solución al problema.

CAPÍTULO 1
DEFINICIÓN DEL PROBLEMA

1.1. Problema a estudiar

Dado un conjunto X de objetos de interés (imágenes, textos, etc), y un conjunto de clases o categorías $Y = \{c_1, c_2, \ldots, c_K\}$, un problema de clasificación consiste en construir una función $f : X \to Y$ que permita asignar a un objeto $x \in X$, una clase o categoría "correcta" $y \in Y$. Este problema se denomina de múltiples clases cuando $K > 2$. Por ejemplo, en reconocimiento de dígitos manuscritos, X es el conjunto de todas las posibles imágenes de una cierta resolución correspondientes a números escritos a mano por una persona y el conjunto de clases es $Y = \{0, 1, 2, \ldots, 9\}$.

Una red neuronal artificial es un sistema capaz de aprender automáticamente esta función, a partir de ejemplos. Típicamente, estos ejemplos consisten en un un conjunto de datos etiquetados de la forma $\{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$, donde $x^{(i)} \in X$ representa un posible objeto de interés e $y^{(i)} \in Y$ la categoría que se desea asociar a $x^{(i)}$.

Al pensar en solucionar un problema de clasificación con múltiples clases usando redes neuronales se piensa inmediatamente en una arquitectura con un número de neuronas de salida igual a la cantidad de clases. Esto se ha vuelto una práctica común, pero no se ha investigado en profundidad si descomponer un problema de clasificación con múltiples categorías en subproblemas binarios, mejora el comportamiento de una red neuronal profunda, estrategia que en cambio se usa con frecuencia con muchos clasificadores de machine learning clásicos (no profundos), mejorando sus resultados.

Entrenar una sola red profunda que clasifique todas las clases al mismo tiempo acarrea ciertos problemas cuando se tiene una gran cantidad de clases, ya que la red tiende a ignorar ciertas clases y a enfocarse más en otras. Este problema se intensifica cuando se tiene una desproporción en la cantidad de datos de entrenamiento por clase, por lo cual nace la necesidad de investigar otras estructuras de redes neuronales para clasificar múltiples clases.

Con esta idea en mente, se adaptará el método propuesto en [4] para el entrenamiento de SVMs (máquinas de vectores de soporte) multi-categoría al entrenamiento de redes neuronales para subdividir el problema de clasificación. Se compararán los resultados obtenidos con tres benchmark. Los dos primeros consisten en los métodos de división en cadena y OAO presentados en [5]. El tercero consistirá en usar una red neuronal ordinaria con arquitectura extraída de los estados del arte de cada dataset [6]. Finalmente se estudiará como se comporta la propuesta sobre un dataset desbalanceado artificialmente y otro dataset desbalanceado de manera natural.

1.2. Objetivos

1.2.1. Objetivo General

Determinar si descomponer un problema de clasificación con múltiples categorías en subproblemas binarios mejora el entrenamiento de una red neuronal profunda.

1.2.2. Metas Especificar a cumplir

- Implementar en lenguaje Python dos métodos de descomposición propuestos en la literatura para resolver problemas de clasificación multi-categorías con redes neuronales y que representen el estado del arte en el tema.
- Proponer un método de descomposición todavía no utilizado para resolver problemas de clasificación multi-categorías con redes neuronales.
- Comparar los métodos de descomposición en al menos tres problemas de clasificación multi-categorías.
- Evaluar los métodos de descomposición en problemas de clasificación desbalanceados.
- Documentar los resultados obtenidos para facilitar investigación futura en el tema.

CAPÍTULO 2
MARCO CONCEPTUAL

2.1. ¿Qué son las redes neuronales artificiales?

Las redes neuronales artificiales (o ANN por sus siglas en inglés) son un modelo computacional inspirado en el comportamiento básico de las neuronas biológicas del cerebro. Las ANN están compuestas por capas que contienen un conjunto de nodos (los cuales representan las neuronas) y a su vez, cada capa de la ANN está conectada a una capa siguiente a través de la conexión de las neuronas de una capa a otra. Esta conexión permite simular el proceso básico de transmisión de información en el cerebro, permitiendo a las ANN aprender y especializarse automáticamente a una tarea determinada.

Según [7], una ANN se puede descomponer en 3 partes importantes. Primero se tiene una capa de entrada (input layer) la cual transforma la información que entra a la red. Esta información se puede interpretar como un impulso que se irá transmitiendo a las distintas capas. La segunda parte consiste en las capas escondidas (hidden layers), las cuales se pueden describir como funciones que transforman la representación inicial de los datos de entrada en una representación propia de la red. La tercera y última parte es la capa de salida (output layer). En esta capa se recoge la representación interna de la red y se transforma en una respuesta legible para un humano.

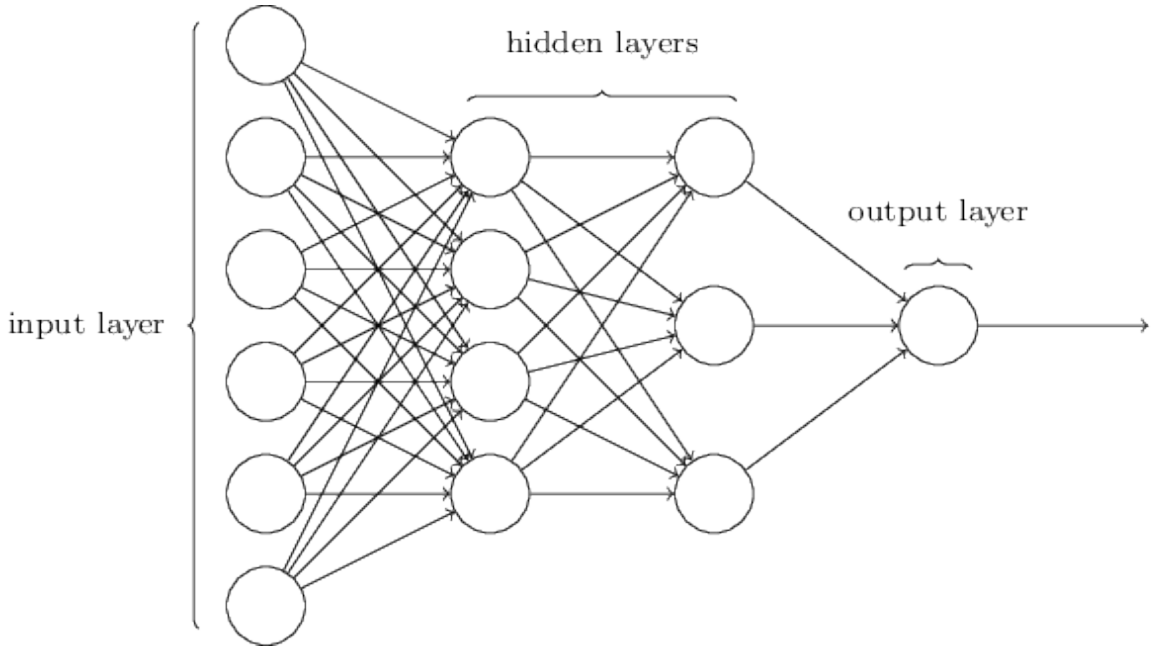


Figura 1: Modelo básico de una red neuronal artificial.

2.2. Componentes de una red neuronal artificial

2.2.1. Neurona

El componente básico de las redes neuronales es la neurona artificial, la cual simula el comportamiento de enviar o no un "impulso" a otras unidades de la red. El primer modelo de neurona artificial fue definido por [8] mediante la siguiente función:

$$N = \sigma\left(\sum_{i=1}^{d} W_i\, X_i - b\right).$$

Los elementos de la función son:

- σ = Función de activación.
- X = Vector de datos de entrada, representado de forma que la red neuronal pueda aceptar el input.
- W = Vector de pesos. El peso W_i se asocia a una conexión entre dos neuronas, y regula la importancia de la transmisión de información entre las capas de la red.
- b = Valor de sesgo, que permite mover la función de activación hacia la izquierda o hacia la derecha.

2.2.2. Función de activación σ

La función de activación en redes neuronales es muy importante, ya que introduce propiedades no lineales en el modelo, permitiendo a la red aproximar relaciones complejas entre los inputs y los outputs que se le entregan. Su objetivo principal es convertir una señal de entrada de una neurona en una señal de salida que se usa como una señal de entrada en la siguiente capa de la red. Según [9], la función de activación tiene que ser no-constante y diferenciable de modo que la red obtenga la propiedad denominada "de aproximación universal", es decir, que sea capaz de aproximar cualquier función entre inputs y outputs. Alguna de las funciones de activación más comunes y que utilizaremos en esta memoria son:

- **Relu:** Usada ampliamente en las CNN y en Deep Learning. Esta función asigna a 0 todos los valores de entrada menores a 0 y es lineal cuando los valores de entrada son mayores a 0, es decir,
$$\sigma(x) = max(x, 0).$$

- **Sigmoid:** Una función ampliamente usada en los inicios del Deep Learning, tiene forma de S y se caracteriza por mantener un valor 0 hasta que se supera cierto umbral de activación desde el cual crece rápidamente hasta su valor máximo. Matemáticamente,

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

- **Tangente hiperbólica:** También conocida como la "mejora" de la función sigmoid, simula de mejor manera el comportamiento de una neurona real. Su rango de normalización es de $[-1, 1]$. La ventaja sobre la función sigmoid es que las entradas negativas se mapearán a valores fuertemente negativos y las entradas cero se mapearán cerca de cero. Matemáticamente,

$$\sigma(x) = tanh(x) = \frac{2}{1 + e^{-2\,x}} - 1 = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

- **Softmax:** Función de activación logística más generalizada para la clasificación multiclase. Esta función es necesaria para que las probabilidades condicionales obtenidas por las neuronas en las capas de salida sumen 1. Matemáticamente,

$$\sigma(x)_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}.$$

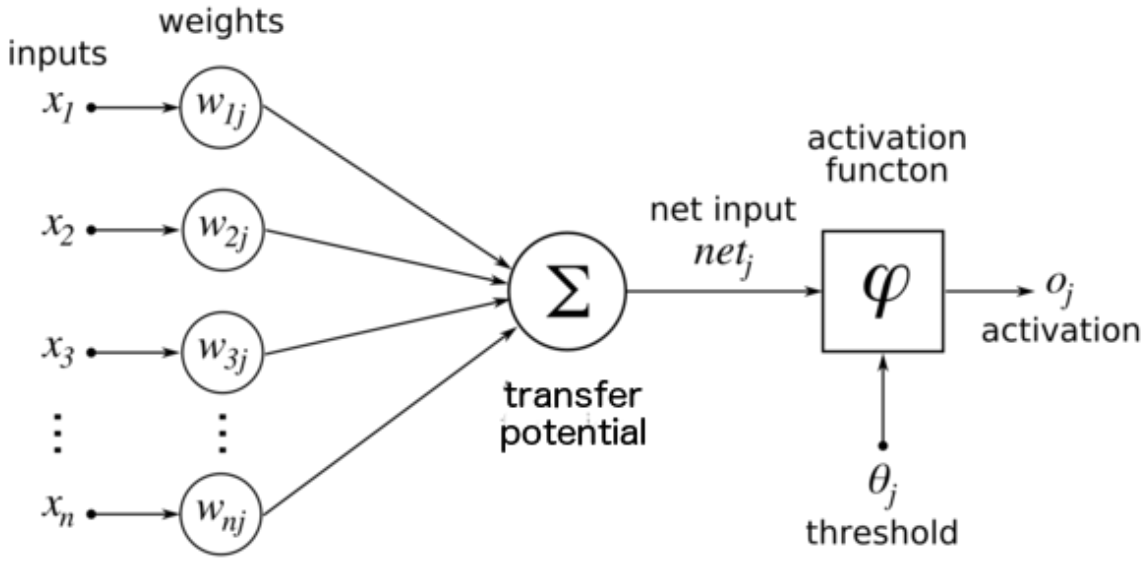


Figura 2: Modelo de neurona con función de activación φ.

2.2.3. Tipos de redes neuronales artificiales

Para esta memoria, trabajaremos con dos tipos de redes: redes feedforward y redes convolucionales.

2.2.4. Redes Feedforward

Las redes feedforward son las ANN más comunes. La idea detrás de este tipo de red es que el conocimiento o información que entra a la red tiene que ser pasado ordenadamente de capa en capa hacia la salida, por lo cual las neuronas no tienen ciclos ni loops entre sí y solo ocurre conexión entre capas vecinas.

Las redes feedforward fueron unas de las primeras redes en construirse por lo cual su idea no cambia mucho el concepto básico de red neuronal. Por lo tanto, este tipo de red se define especificando:

- Una capa de entrada donde se representan los datos de entrada como un vector de información.
- Una cantidad N de capas ocultas donde se procesa internamente la información.
- Una capa de salida donde se le da "sentido" al resultado obtenido por la red para que este resultado sea entendido por los humanos.

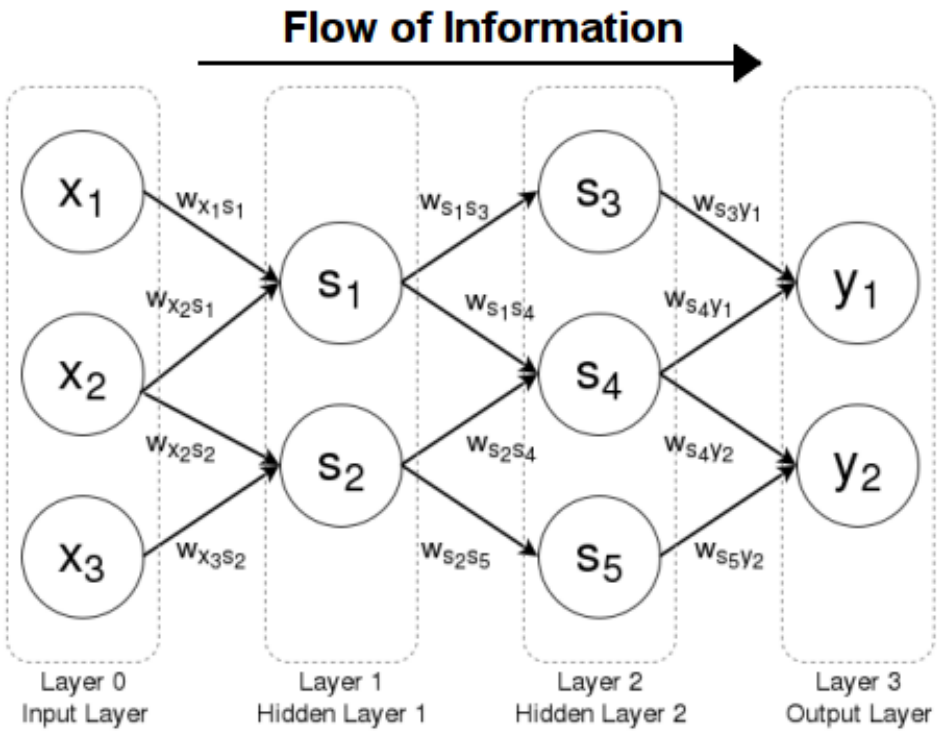


Figura 3: Modelo de Red Feedforward.

2.2.5. Entrenamiento

El entrenamiento de una red neuronal feedforward es importante, ya que en este proceso se ajustan los pesos de las neuronas y dependiendo de este ajuste la red neuronal logrará resolver el problema para cual fue construida o no. El método más usado actualmente para realizar este ajuste de pesos se conoce como **Backpropagation** y su expresión matemática es:

$$\Delta w_{ij} = \eta \, \phi_j \, y'_i.$$

- w_{ij}^k: Peso w de la neurona j a la neurona i.
- η Factor de aprendizaje, con valores]0,1[.
- ϕ Error de la neurona j.
- y'_i Output de la neurona i o estimador de i.

Podemos descomponer la variable ϕ_j en 2 tipos, una para neuronas en la última capa o capa de salida:

$$\phi_j = (y - y') \, f'_j(total_j).$$

y una para las neuronas en una capa escondida:

$$\phi_j = f'_j(total_j) \sum_{l \in L} (\phi_l w_{jl}).$$

- y output esperado.
- f'_j derivada de la función de activación.
- L Conjunto de neuronas de la siguiente capa.
- $total_j$ Ponderación de los y' de la capa anterior.

Una red neuronal realiza varias iteraciones de la regla antes mencionada para terminar su entrenamiento, con el objetivo llegar a unos pesos que le permitan realizar la tarea que se quiere aprender.

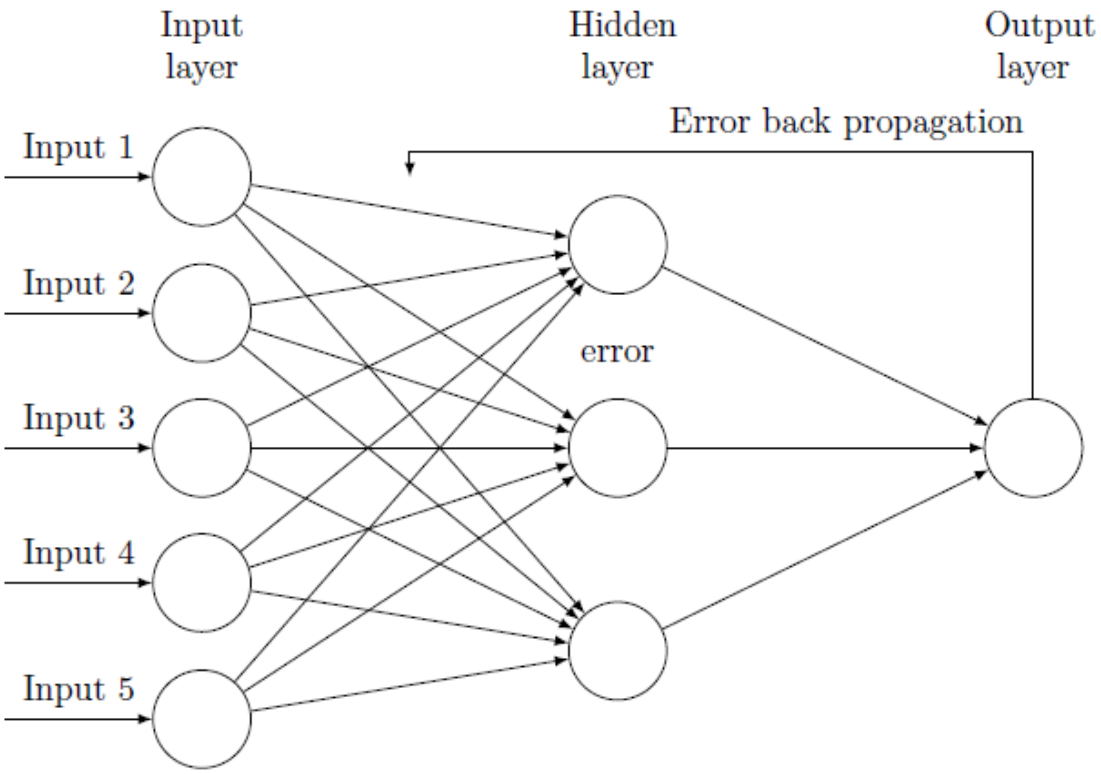


Figura 4: Modelo básico del backpropagation.

2.2.6. Redes Convolucionales

Las redes neuronales convolucionales (o CNN por sus siglas en inglés) son redes especializadas en problemas relacionados con la computación visual, por lo cual tienen un gran desempeño en clasificación de imágenes y reconocimiento de patrones en vídeos. Además, no se quedan atrás en otras áreas, como por ejemplo sistemas recomendadores [10] y sistemas generadores de imágenes [10].

Las CNN comparten muchas características de las redes feedforward, incluyendo la existencia de neuronas, capas y pesos. La gran diferencia es que los datos de entrada en una CNN pueden ser multidimensionales. En el caso de una imagen, esta tendrá 3 dimensiones, las cuales son el alto, el ancho y los canales de colores [1]. La gran diferencia que tienen las CNN es la implementación de una capa de convolución. Esta capa toma una cierta cantidad fija de píxeles de la imagen, predeterminados por una ventana de un cierto tamaño, y realiza una convolución con una cierta matriz de pesos, para determinar la activación de una neurona de la capa. Este proceso de repite hasta que la ventana recorre toda la imagen, en todos

[1]Nos referimos a los distintos colores de los píxeles de una imagen, normalmente son rojo, verde y azul, pero una imagen puede tener otros canales

sus canales, generando un conjunto de activaciones conocido como mapa de características. Este mapa se transforma en la entrada de la próxima capa de convolución. Además de las capas de convolución, una CNN típica también incluye una nueva capa denominada capa de pooling, la cual se usa para reducir las dimensiones espaciales del patrón de entrada, pero no la profundidad [2]. Esta capa tiene como finalidad:

- Reducir un poco las dimensiones de la imagen y mejorar el procesamiento o reducir el estrés computacional.
- Al tener menos dimensiones se tiende a reducir el overfitting de la red.

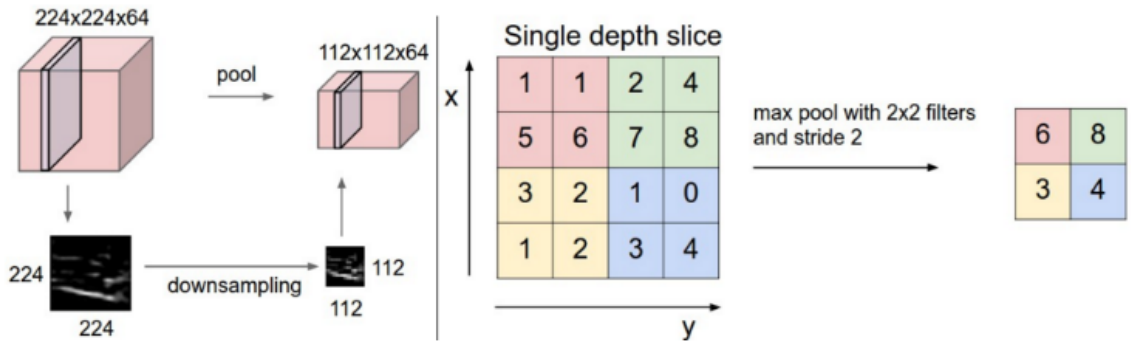


Figura 5: Esquema de como actúa una capa de pooling sobre una imagen.

Finalmente, lo que se hace normalmente es “aplanar” la salida de las capas convolucionales con una capa denominada Flatten para transformar dichas salida a un vector, para posteriormente entregar este vector a una red Feedforward para que finalmente, la red pueda realizar su tarea establecida. Resumiendo lo anterior, podemos decir que una red convolucional se compone de dos secciones: una sección de convolución donde se mapea la información de la imagen o video a una nueva representación y otra sección Feedforward donde se utilizan esos datos para generar clasificación, detección de patrones, etc.

[2]Esto se refiere a que no se afecta los canales de la imagen.

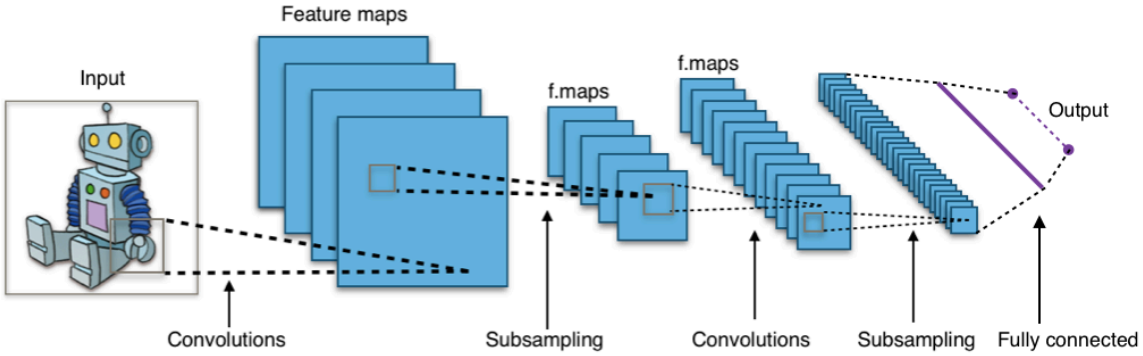


Figura 6: Esquema completo de una red convolucional.

2.3. Estado del Arte

2.3.1. Metodología DAG

En el año 2000, [4] propuso una nueva arquitectura de aprendizaje, el gráfico acíclico dirigido (o DAG por sus siglas en inglés). DAG es un grafo que solamente tiene una orientación y no posee ciclos, tiene un nodo raíz y nodos hojas, por lo cual su estructura es similar a un árbol binario.

La idea fundamental de esta arquitectura es aplicar la metodología de dividir y vencerás a un problema de clasificación de múltiples clases, generando K(K-1)/2 nodos, siendo K la cantidad de clases. Cada nodo de nuestro grafo DAG será un clasificador. Parte de la información de salida de un nodo serán los datos de entrada de otros nodos. Al terminar este proceso, se genera la clasificación final en los nodos hojas.

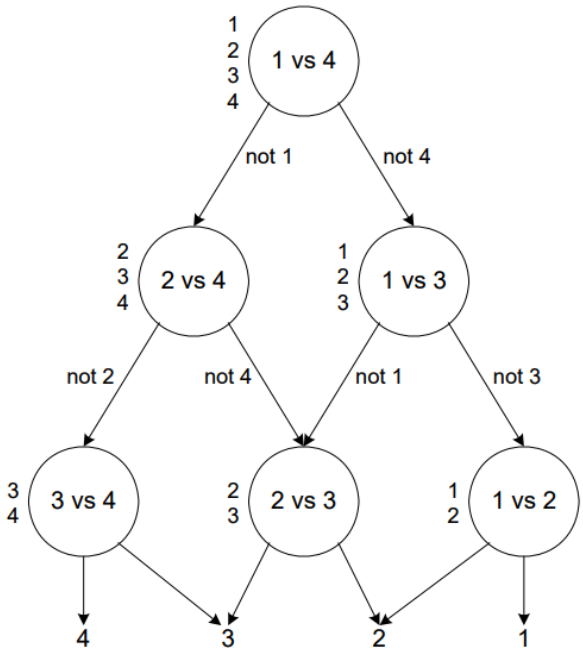


Figura 7: Visualización de la metodología original DAG donde las clases están representadas como 1,2,3,4.

La figura 7, se observar un grafo DAG generado para evaluar 4 clases. La lista de clases a evaluar que parte en el nodo raíz es [1,2,3,4]. Esta lista fue elegida ordenadamente para facilitar la explicación, pero el orden puede ser cualquiera. Se utilizará la misma forma de seleccionar los pares de clases a comparar que en [4]. Cada nodo compara la primera y última clase de su lista de clases. Por ejemplo, el nodo raíz compara las clases 1 y 4 de su lista. Luego de identificar la clases a comparar, el nodo clasifica los datos de entrada según las dos posibles opciones y luego distribuye los datos de entrada según la negación de estas dos opciones, enviando los datos que no son de la primera clase de la lista hacia la izquierda y los que no son de la última clase de la lista hacia la derecha. En el caso del nodo raíz, el clasificador toma los datos de entrada y los clasifica como 1 ó 4, para luego enviar los datos que no son 1 hacia la izquierda y los que no son 4 a la derecha. Este proceso se repite hasta completar la clasificación.

2.3.2. Estudio de metodologías actuales

El 10 de octubre del 2005, [5], se publicaron los resultados de un estudio exhaustivo donde se analiza como se comporta la clasificación frente a distintos tipos de arquitecturas. Las arquitecturas en este estudio son múltiples ANN binarias para cada clase o una sola ANN para todas las clases.

Clasificación de K-clases usando una arquitectura de múltiples redes neuronales

Estas arquitecturas se basan en que existen $M \geqslant K$ redes neuronales binarias (donde K es el número de clases y M la cantidad de redes). Además, se hace una subdivisión dependiendo de qué metodología se usa para entregarle al sistema de redes binarias los datos de entrenamiento. Las distintas metodologías son One Against All (OAA), One Against One (OAO) y P Against Q (PAQ). Los resultados de esta arquitectura y sus distintas metodologías de entrenamiento son:

- **One Against All (OAA)**

  La metodología OAA utiliza un sistema de M = K redes binarias. Cada red binaria tiene una neurona de salida la cual entrega un valor igual a 1 cuando un dato de entrada pertenece a la clase i o 0 cuando no pertenece a dicha clase. Cada una de las ANN es entrenada con el mismo conjunto de datos pero en cada una de las redes se cambia la etiqueta de las clases, transformando la etiqueta de la clase de interés a un 1 y el resto de las clases se etiquetan con un 0. Existen tres posibles opciones de salida para las K redes binarias $f_1 ... f_M$:

  - El caso ideal, donde $f_i = 1$ y $f_j = 0 \; \forall i, j \wedge i \neq j$. El sistema de decisión para este caso es $F(\bar{x}, f_1, f_2, ..., f_M) = argmax_{i=1,..,M} f_i$. donde $\bar{x}$ son los subconjuntos generador al remover una clase del conjunto x.
  - El segundo caso es que todas las redes $f_i = 0 \; \forall i \in 1, ..., M$. En este caso la respuesta del sistema tiene que ser "no lo se". Una segunda aproximación es hacer que la función de decisión mire la salida de la función de activación de cada red binaria para que obtenga la etiqueta de clase correspondiente a la red binaria con el mayor valor de salida en la función de activación. Matemáticamente esto es $F(\bar{x}, y_1, y_2, ..., y_M) = argmax_{i=1,..,M} = y_i$ donde y_i es la salida de la función de activación usada en la output layer de la i-ésima red binaria.
  - El último caso es cuando más de una red asigna un 1 a un mismo dato de entrada. Para este caso se tienen múltiples opciones. La más simple es asignar el resultado como "empate" y decidir si el empate es aceptable (es decir, se permite que un mismo dato tenga 2 o más etiquetas finales) o no. Si es aceptable, se puede escoger aleatoriamente la etiqueta final y si no es aceptable, se puede ocupar la fórmula en el punto anterior para seleccionar la etiqueta final.

  Ventajas

  - Como cada ANN esta entrenada individualmente, cada una posee su propio espacio de características, por lo cual se puede implementar un extractor de características haciendo flexible el entrenamiento, sin afectar al resto de las redes.
  - Cada red puede tener su propia arquitectura, es decir, su propia cantidad de capas escondidas, número de nodos, función de activación, etc.

- Al ser redes que se entrenan individualmente, esto se puede hacer simultáneamente en diferentes computadores, paralelizando el entrenamiento y haciéndolo más rápido.

Desventajas

- Si se posee data desbalanceada [3], se tiende a ignorar la clase con la cantidad menor de datos de entrenamiento. Esta metodología no corrige el problema de sensibilidad al desbalanceo inherente al método sin descomposición ya que normalmente empeorar el desbalance en problemas desbalanceados y crea desbalance en el caso de problemas balanceados. Este comportamiento se debe a que la red usa como clase negativa todos los ejemplos que no son la clase objetivo, lo cual significa que la red tiende a observar una gran cantidad de ejemplos negativos en comparación a los positivos.
- Como las redes no tienen una comunicación constante entre sí en su entrenamiento, la frontera de división puede sobreponerse, clasificando un mismo dato con dos etiquetas o puede que algunas secciones del espacio de búsqueda queden fuera de todas las frontera, imposibilitando su clasificación.

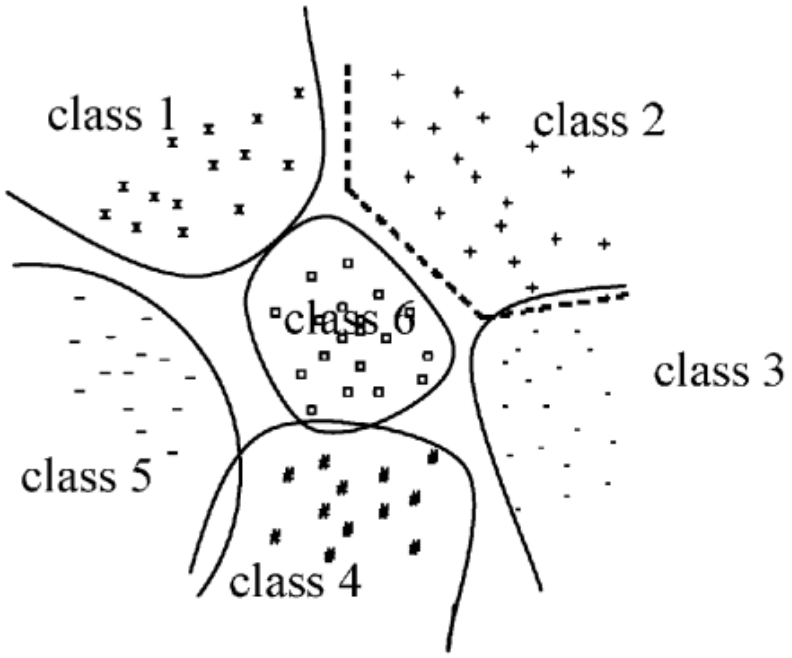


Figura 8: Frontera de clasificación dibujadas por 6 ANN binarias con metodología OAA.

- **One Against One (OAO)** OAO es una metodología compuesta por $K(K-1)/2$ [4] redes binarias. Cada red binaria es entrenada para discriminar entre una clase i y otra clase

[3] Conjunto de datos donde no se tiene el mismo número de entradas/imágenes/información para cada clase
[4] Recordemos que K es la cantidad de clases

j. Para evitar considerar un par de clases dos veces, se conviene que i sea siempre menor que j. Cada red es entrenada con datos de la clase i y j y su salida $f_m(i,j)$ es binaria, indicando si el dato de entrada $\bar{x}$ pertenece a la clase i o j. La salida final de este sistema es la colección total de los votos de las $K(K-1)/2$ neuronas. Ya que esta metodología genera una mayor cantidad de redes que clases, la función de decisión genera un gran impacto en el rendimiento del modelo. Existen dos funciones de decisión, las cuales son:

- La función de decisión más simple es por mayoría de voto, tomando el resultado de las $K(K-1)/2$ redes. La clase con la mayor cantidad de votos es la seleccionada como etiqueta por el sistema. Los empates se pueden resolver mediante selección aleatoria.
- Una extensión de la mayoría de votos es considerar la confianza $y_{i,j}$ con la cual una clase es asignada a un dato de entrada. Matemáticamente, se tiene que la función de decisión toma la forma:

$$F(\bar{x}) = argmax_p\Big(\sum_{j=p+1}^{K} y_{p,j} + \sum_{j=1}^{p-1}(1-y_{j,p})|p=1,...,K\Big),$$

  donde $y_{i,j}$ corresponde a la confianza que el dato $\bar{x}$ tiene de ser i y $(1-y_{i,j})$ es la confianza del dato $\bar{x}$ de ser j para todo $i=1,...,K-1$ y $j=i+1,..,K$.

Ventajas

- Como se entrenan K(K-1)/2 redes binarias, el sistema promueve la redundancia [5] por lo cual aumenta la generalización [6] del sistema.
- Posee independencia de espacio de características al igual que OAA.
- Tiene la capacidad de entrenamiento simultáneo al igual que OAA.
- No sufre tanto de data desbalanceada ya que cada red es entrenada únicamente clasificando dos clases. Esta ventaja se invalida si ambas clases están desbalanceadas.
- Gracias a la redundancia, se tiende a tener menos áreas del espacio de búsqueda sin cubrir por las fronteras de clasificación.
- Es escalable, ya que si se requiere incorporar otra clase al sistema, denominada clase $K+1$, solamente tenemos que entrenar K nuevas redes neuronales, sin afectar las pre-existentes $K(K-1)/2$ redes.

Desventajas

[5] La redundancia en ANN tiene efectivos positivos ya que si una red entrega una mala clasificación las otras redes pueden detectar correctamente ésta y suplir el error de una red

[6] La generalización refiere a que tan bien la red o el sistema de redes funciona con nuevos datos de entrada

- El excesivo incremento del sistema OAO. El sistema crece a una tasa de K^2 por lo cual, cuando K es muy grande, el sistema con metodología OAO puede tener mucho más redes que un sistema con OAA o PAQ. Estudios concluyen que con K >20, el tiempo de entrenamiento con metodología OAO es mayor que OAA y PAQ.

- **P Against Q (PAQ)**

  La metodología PAQ resuelve el problema de clasificación de K-clases implementando un sistema de M redes neuronales. Cada red tiene una salida binaria entrenada para P clases contra Q clases, donde $P \geq 1$ y $Q \geq 1$. Un modelo PAQ puede ser descrito con una tabla de verdad de K codewords de largo M, una para cada clase. El contenido de los codewords pueden ser 0, 1 o "no se necesita". Cada bit en los codewords es la salida de una red binaria de dos clases entrenada en la combinación de las palabras claves de ese bit. Sea $f_0, f_1, ..., f_{M-1}$ la salida de M redes binarias de dos clases, y $cw_1, cw_2, ..., cw_K$ los codewords para las K clases. Para entrenar la j-ésima red, la función de salida $f_j(x)$ es aprendida mediante un re-etiquetado de los datos de entrenamiento como $(x_1, f_j(x_1)), (x_2, f_j(x_2)), ..., (x_n, f_j(x_n))$, donde:

$$f_j(x_i) = \begin{cases} 1 \text{ si } x_i \text{es de la clase p y el j-ésimo bit de } cw_p \text{ es } 1, \\ 0 \text{ si } x_i \text{es de la clase p y el j-ésimo bit de } cw_p \text{ es } 0. \end{cases}$$

  Si el j-ésimo bit de un codeword de una clase es "no se necesita", entonces los datos que pertenecen a esa clase son removidos de los datos de entrenamiento. Como resultado, se obtienen M redes con una función de salida $\hat{f}_j(\bar{x}) \forall j \in \{0, 1, ..., M-1\}$. La función de decisión final depende de la codificación de las clases. Una codificación simple es codificar K clases en $M = \log_2(K)$ bits. Para un problema de 8 clases se tienen $M = 3$ bits.

  Ventajas

  - Posee independencia del espacio de características al igual que OAA y OAO.
  - Cada red posee su propia arquitectura, es decir, número de capas ocultas, número de neuronas por capa, función de activación, etc.
  - Cada red se puede entrenar independientemente de las otras redes.

  Desventajas

  - El problema principal de PAQ es que no genera redundancia cuando $M = K$, por lo cual si un clasificador falla en alguna clase, todo el sistema fallará al clasificar dicha clase. Esto es causado por que la distancia de Hamming entre dos codewords es 1, por lo tanto, solo basta con que una ANN falle para que se produzca un error en todo el sistema.

- **One Against higher orden modeling (OAHO)**

  El modelo OAHO entrena $(K-1)$ redes binarias, usando el siguiente algoritmo. Se tienen K clases en una lista de clases $\{C_1, C_2, ..., C_K\}$. La primera red neuronal NN$_1(C_1, C_2+)$ es entrenada con ejemplos de la clase C_1 marcadas como "1" y todas las otras clases como "0". La segunda red NN$_2(C_2, C_3+)$ se entrena con ejemplos de la clase C_2 marcadas como "1" y ejemplos de la clases con orden superior $C_3, C_4, ..., C_K$ marcadas como "0". En general, la red NN$_i(C_i, C_i+)$ es entrenada con clases C_i como clases "1" y $C_{i+1}, ...C_K$ como "0". La figura 9 ilustra como clasifica un sistema OAHO. Los datos de prueba $\bar{x}$ son enviados en un principio a los extractores de características asociados a cada red para ser transformados en un vector de características $\bar{x}_1, \bar{x}_2, ..., \bar{x}_{K-1}$, permitiendo a cada red definir su propio espacio de características. Primero se activa la primera red NN$_1(C_1, C_2+)$, si predice $\bar{x}_1$ como la clase C_1, el sistema entrega el resultado y el proceso se detiene. En caso contrario, NN$_2(C_2, C_3+)$ es activada, si esta red entrega un resultado positivo (1), el proceso se detiene, en caso contrario, se sigue repitiendo el proceso como ilustra la figura 9. Si el proceso avanza hasta la última red, la clasificación resultante del dato de entrada es C_{K-1} o C_K.

  Ventajas

  - El sistema OHAO puede ser muy eficiente si las restricciones usadas para ordenar la lista de clases están bien definidas para resolver un problema en específico. En general, las clases son ordenadas aleatoriamente, en base a la propiedad de los datos de entrenamiento o a la importancia de cada clase para el problema a resolver. Otra forma de ordenar esta lista es ordenar las K clases en base al tamaño de los datos de entrenamiento de dicha clase. Matemáticamente, se ordena la lista de clases como $\{C_1, C_2, ..., C_K\}$ si solo si $|\Omega^i_{tr}| \leq |\Omega^{i-1}_{tr}|$, $i = 1, 2, ..., K-1$, donde Ω^i_{tr} son los datos de entrenamiento de la clase C_i para $i = 1, 2, ..., K$. Usando esta aproximación, todas las clases con pocos datos de entrenamiento son usadas en conjunto como datos negativos en contra de los ejemplos de una única clase con muchos datos de entrenamiento en la red binaria con un alto nivel en la jerarquía. Este proceso reduce el impacto de dataset de entrenamiento desbalanceados.
  - Otro importante aspecto del modelo OAHO es su escalabilidad. Si queremos agregar una nueva clase, llamémosla C_0. El sistema de $K+1$ clasificadores se mantiene igual que la figura 9 con una nueva red binaria $NN(C_0, C_1+)$ agregada en el nivel más alto.

  Desventajas

  - El proceso de clasificación en un sistema OAHO es completamente jerárquico, esto implica que si cualquier nodo realiza un error en la clasificación, no se podrá corregir este error. Esto significa que el sistema es poco robusto a errores, por lo cual cada nodo del modelo OHAO tiene que ser entrenado de manera que sea lo suficientemente consistente para disminuir estos errores al máximo.

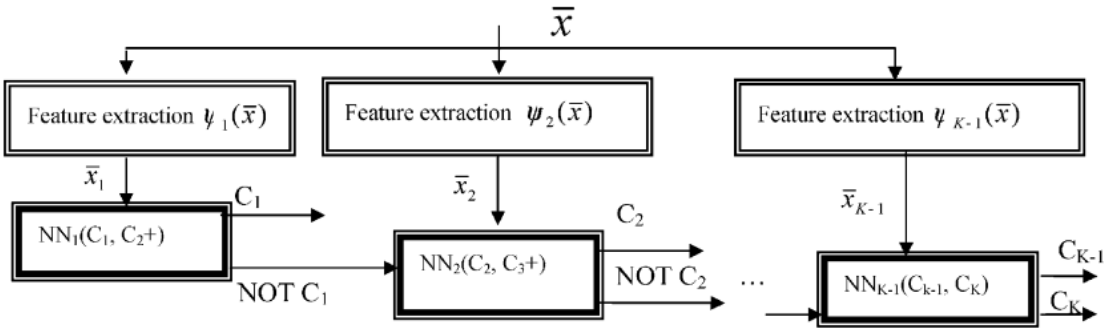


Figura 9: Ejemplo de sistema clasificador para K clases usando la metodología OAHO.

Clasificación de K-clases usando una única red neuronal

Esta arquitectura se basa en tener una única red neuronal que clasifique todas las clases. Esto se logra teniendo una M cantidad de neuronas de salida siendo M determinada por la cantidad de clases K, también se tiene que tener d neuronas de entrada, donde d es la dimensión de los atributos de los datos de entrada. Por la naturaleza de la red, hablar de la metodología OAO en un único clasificador para K es lo mismo que analizar los baseline con una sola red neuronal, por lo cual el estudio muestra las conclusiones con metodologías OAA y con PAQ.

- **One Against All (OAA)**

  Cuando modelamos una red con una estructura OAA para clasificar K clases, se tienen que tener M = K nodos de salida, cada uno denotado como $O_1, O_2, ..., O_k$. Cada clase está codificada usando un codeword O de K bits. Para la i-ésima clase, se tiene $O_1 = ...O_{i-1} = 0$, $O_i = 1$, $O_{i+1} = ..., O_k = 0$. Para los datos de entrenamiento $\bar{x}$, la salida esperada de la red en el nodo de salida f_i se establece en 1 si solo si la clase $\hat{x}$ pertenece a la clase i, de otra forma se establece como 0. Como se está usando una única red para clasificar las K clases, la estructura de la red tiende a ser más compleja. En la etapa final de clasificación, el dato de prueba $\bar{x}$ se le asigna una etiqueta de clase dependiendo del codeword binario con la menor distancia de Hamming a la salida de la red F($\bar{x}$). Esta arquitectura es la forma clásica de implementar una red que clasifica K clases mediante one hot vector.

  Ventajas

  - Al ser una única red, se puede entrenar la red de forma que las fronteras de clasificación sean más precisas, minimizando la existencia de doble clasificación o áreas del espacio de búsqueda sin cubrir.

- Al ser una sola red, se pueden realizar redes sumamente complejas y trabajar tanto el espacio de características como las capas escondidas para mejorar la precisión y el uso de la red.

Desventajas

- Al igual que su contratarte binaria, se tiende a ignorar las clase menos representativa o clase con menos datos de entrenamiento.
- Normalmente para estos clasificadores de K clases, el tiempo de entrenamiento es bastante alto cuando la cantidad de datos de entrenamiento es alta.
- Idear una arquitectura compleja puede ser bastante difícil, por lo cual puede tomar mucho más tiempo construir la red que su contra parte binaria.

- **P Against Q (PAQ)**

  La metodología PAQ también puede ser implementada en una sola red neuronal con $M > K$ [7] nodos de salida. La funciones de decisión en los nodos de salida $f_1, f_2, ..., f_M$, colectivamente representan un codeword de M bits asignado a cada clase. Por lo cual, la función de salida depende de la codificación realizada a cada clase, de cuantas clases se tienen y cuál será la distancia de Hamming aceptable, ya que como se puede tener más neuronas de salida que de clases, para generar code word con más bits, la codificación depende de quien implementa la arquitectura.

  Ventajas

  - Comparte todas las características positivas de una red de K clases OAA, incluyendo la división precisa de las fronteras de clasificación y de poseer la cualidad de aumentar su precisión a medida que aumenta su complejidad.

  Desventajas

  - Requiere un mayor tiempo de entrenamiento que una red K clases OAA. Esto se debe a que se tienen más nodos que clases, ya que se necesita garantizar una distancia de Hamming tolerable entre las clases. Los autores concluyen que no es recomendable usar PAQ con una sola red clasificadora si la magnitud de los datos de entrenamiento es muy grande o la cantidad de clases es alta.

[7] Recordemos que M es el número de neuronas de salida y K es la cantidad de clases.

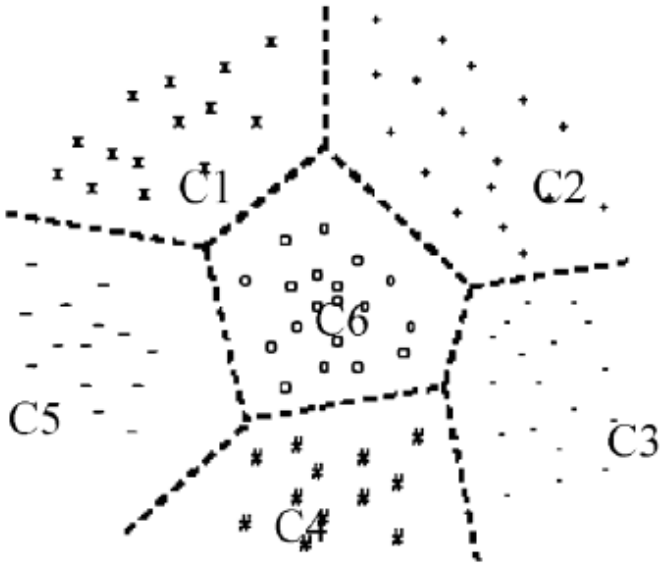


Figura 10: Frontera de clasificación ideales generadas por una sola red clasificadora para K clases. Siendo esta la clasificación OAO de una red con múltiples neuronas de salida.

2.3.3. Datasets desbalanceados

Un problema común en el mundo del aprendizaje profundo son los dataset desbalanceados. Estos datasets tienen la particularidad de que su data no está repartida equitativamente en todas las clases (por ejemplo, MNIST tiene 6000 datos por clase), si no que más bien, existe una cierta cantidad de clases que predominan en cantidad de datos de ejemplos que otras. En el caso más extremo, solamente existe una clase que tiene el 90 % de los datos de todo el dataset. Esta condición provoca que los algoritmos de aprendizaje tiendan a generar un sesgo hacia las clases predominantes y a ignorar las clases con menos datos de ejemplos, generando una pobre clasificación de las clases minoritarias. Algunas tareas que tienen que lidiar con dataset desbalanceados son la detección de derrames de petróleo, detección de fraudes y detección de enfermedades raras [11]. La presencia de data desbalanceada puede resultar en anomalías en el aprendizaje, generando paradojas de accuracy [11] las cuales consisten en la dificultad de controlar los errores tipo 1 y 2 al generar la clasificación. Un ejemplo claro de este problema es que el caso de enfermedades raras, en que el clasificador tiende a elegir siempre la clase mayoritaria y por lo tanto reconoce pocos casos en que se presenta la enfermedad.
Actualmente existen varios métodos para corregir este problema, pero veremos algunos de los más tradicionales [12]:

- **Undersampling:** Método en el que seleccionamos aleatoriamente ejemplos de la clase mayoritaria y descartamos las restantes. Dado que asumimos que cualquier muestra

aleatoria refleja con precisión la distribución de los datos, este es un enfoque ingenuo. Este es un método clásico en el que el objetivo es equilibrar las distribuciones de clase mediante la eliminación aleatoria de los ejemplos de clase mayoritaria. Esto lleva a descartar datos potencialmente útiles que podrían ser importantes para los clasificadores.

- **Oversampling:** Técnica que replica ejemplos de las clases minoritarias para que la distribución sea equilibrada. Pero un oversampling ingenuo tiene el problema de que va aumentando la probabilidad del overfitting al replicar los mismos datos de ejemplos minoritarios en el modelo.

- **Data Augmentation:** Data augmentation es un caso particular del oversampling. Esta técnica también replica las imágenes de las clases minoritarias pero se le aplica una transformación previa. Por ejemplo, podemos tomar un imagen de un cierto dataset desbalanceado y generar 5 nuevas imágenes en base a la imagen original. A estas imágenes generadas artificialmente se les incorpora ruido gaussiano, rotaciones, distintos zooms, shifts, etc. Ya que las imágenes generadas no son exactamente las mismas que la imagen original, no se tiende a generar mucho overfitting al igual que un simple oversampling.

- **Class re-weighting:** La idea de esta técnica es tener en cuenta la asimetría de los errores de costo directamente durante el entrenamiento del clasificador, es decir, le asignamos un peso de importancia a cada clase (dependiendo de que tan significativa sea) para que la función de costo sepa cuales son las clases menos significativas, por lo cual, se generara una corrección a la asignación de pesos de la red para evitar que se ignore las clases con menos datos de ejemplos.

- **Snow Ball:** Técnica que consiste en entrenar primero una red neuronal de K clases solamente con los ejemplos de las clases minoritarias para que se establezca un conjunto de conexiones con pesos de favorables a estos ejemplos. Luego se aumenta la capacidad de la red para reconocer los ejemplos de la clase mayoritaria mediante el uso de un conjunto de entrenamiento dinámico que incluye todos los ejemplos de clase minoritaria y un número creciente de ejemplos de la clase mayoritaria. De esta manera, el efecto de ignorar las clases minoritarias se puede reducir considerablemente.

Análisis de Rendimiento

El estado del arte analiza la precisión de clasificación y el tiempo de entrenamiento en tres categorías:

- Data de entrenamiento desbalanceada.

- Gran cantidad de clases.

- Dataset de entrenamiento pequeños vs grandes.

Para realizar esto, se ocuparon distintos dataset de entrenamiento con distintos sistemas de ANN, los cuales se explicarán a continuación:

- **Glass** El dataset Glass es un conjunto de imágenes que corresponden a K=6 tipos distintos de vidrios. Este dataset tiene un total de 214 datos en total y no tiene un dataset de prueba.

	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
Glass/training	70	76	17	13	9	29

Tabla 1: Distribución de datos de entrenamiento en el dataset “Glass”.

Este dataset se utilizó para evaluar el comportamiento de diferentes sistemas frente a desbalance de clases. Los autores realizaron distintas pruebas en sistemas binarios OAO, OAA, OAHO y una red de K clases clásica. Como este dataset no contiene un dataset auxiliar de prueba, los resultados obtenidos en la experimentación se generaron mediante un proceso de cross-validation con 10 folds. Los mejores resultados se observan en Tabla 2. Los autores destacan que la red OAHO pudo detectar la clase 3, la cual fue ignorada por el resto de las redes.

	Hidden Nodes	1 (%)	2 (%)	3 (%)	4 (%)	5 (%)	6 (%)	Total (%)
OAA, 6 redes	5	84.29	69.74	0	69.23	66.67	86.21	71.03
OAO, 15 redes	5	68.57	68.42	0	61.54	66.67	82.76	64.95
OAHO, 5 redes	5	85.71	57.89	17.65	69.23	55.56	82.76	67.76
OAA, 1 red	20	84.29	69.74	0	69.23	66.67	86.21	70.56

Tabla 2: Resumen de precisión en “Glass”.

- **Shuttle** Shuttle es una colección de metadata de naves espaciales , distribuidas en 7 clases distintas con 9 atributos por clase. Contiene en total 58000 datos, separados en 43500 datos de entrenamiento y 14500 datos de prueba. Este dataset también es usado para probar distintos modelos en condiciones de clases desbalanceados, ya que en Shuttle el 80 % de los datos de entrenamiento pertenecen a una única clase.

	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7
Shuttle/training data	34,108	37	132	6748	2458	6	11
Shuttle/test data	11,478	13	39	2155	809	4	2

Tabla 3: Distribución de datos de entrenamiento y prueba en el dataset “Shuttle”.

Los modelos probados sobre este dataset fueron redes binarias OAA, OAO, OAHO, una red para K clases clásica OAA y una red para K clases OAA implementando la técnica snowball. Según [11] la técnica Snowball consiste en entrenar primero una red neuronal de K clases solamente con los ejemplos de las clases minoritarias para que se establezca un conjunto de conexiones con pesos de favorables a estos ejemplos. Luego se aumenta la capacidad de la red para reconocer los ejemplos de la clase mayoritaria mediante el uso de un conjunto de entrenamiento dinámico que incluye todos los ejemplos de clase minoritaria y un número creciente de ejemplos de la clase mayoritaria. De esta manera, el efecto de ignorar las clases minoritarias se puede reducir considerablemente. Los autores destacan que en redes binarias, la técnica OAHO mejora la precisión al clasificar dataset con datos de entrenamiento desbalanceados ya que puede detectar la clase 2 que fue ignorado por las demás redes binarias y en una red para K clases, el data augmentation realizado en conjunto con la técnica snowball corrige completamente el problema. Los resultados son observables en Tabla 4.

	Hidden Nodes	1(%)	2(%)	3(%)	4(%)	5(%)	6(%)	7(%)	Total(%)
OAA, 7 redes	20	100	0	15.38	100	99.75	0	0	99.63
OAO, 21 redes	10	100	0	33.33	100	100	0	0	99.69
OAHO, 1 red	30,20,15,10,10,10	99.99	7.69	38.46	100	99.75	0	0	99.69
OAA, 1 red	30	100	0	12.82	100	99.88	0	0	99.63
OAA, 1 red con data augmentation	30	99.94	30.77	64.1	100	99.93	100	100	99.77
OAA, 1 red con ”snowball”	30	100	92.31	94.87	100	99.88	100	100	99.95

Tabla 4: Resumen de precisión en “Shuttle”.

Las redes en OAHO tienen un número variable de nodos ocultos, ya que cada red esta entrenada de forma única. La primera red NN_0 fue entrenada con 30 hidden nodes y con la clase 1 en contra de las clases 2,3,4,5,6,7. La segunda red NN_1 fue entrenada con 20 hidden nodes y con la clase 4 en contra de las clases 2,3,5,6,7. La tercera red fue entrenada con 15 hidden nodes y con la clase 5 en contra de las clases 2,3,6,7. La cuarta red NN_3 fue entrenada con 10 hidden nodes y con la clase 3 en contra de las clases 2,6,7. La quinta red NN_4 fue entrenada con 10 hidden nodes y con la clase 2 en contra de las clases 6,7. Finalmente, la sexta red NN_5 fue entrenada con 10 hidden nodes y con la clase 7 en contra de la clase 6.

- **MNIST**

  MNIST es un dataset balanceado que contiene imágenes de dígitos escritos a mano. El dataset consiste en 60.000 imágenes de entrenamiento y 10.000 de prueba. En Tabla

5 se pueden ver los resultados del accuracy global y el tiempo de entrenamiento de los sistemas. Algunas redes fueron entrenadas con la técnica Error-correcting output code (ECOC) la cual consisten es generar una codificación binaria de M bits, cumpliendo que $M > K$ siendo K la cantidad de clases. Esta codificación sirve cuando se quiere aumentar la distancia de Hamming entre la codificación de las clases. Lo destacable en este estudio es que tanto el sistema de redes binarias y la red de K clases OAA tienen resultados similares en termino de accuracy promedio, sobresaliendo OAO, 45 redes por 0.15 % extra que por sobre OAA, 1 red. Respecto a los tiempos de entrenamiento promedio de los experimentos, se tiene que considerar que los sistemas con redes binarias poseen una cantidad inferior de neuronas que las redes para K clases y cada red se entrenaba con una cantidad de epochs distinta, permitiendo entrenar las arquitecturas en poco tiempo.

	ECOC, 15 redes	OAA, 10 redes	OAO, 45 redes	OAHO, 9 redes	ECOC, 1 red	OAA, 1 red
Accuracy (%)	96.4	95.5	97.54	96.24	96.85	97.39
Hidden Nodes	15	15	20	20	300	400
Tiempo de entrenamiento (h)	6.5	4.2	7	18.1	30	23

Tabla 5: Rendimiento del dataset MNIST en distintos sistemas de redes neuronales.

- **Letter database**

  El database Letter es similar a Mnist, donde se tiene que identificar una amplia cantidad de píxeles blanco y negro como una de las 26 letras mayúsculas en el alfabeto inglés.

Class	Cantidad de Datos
A	789
B	766
C	736
D	805
E	768
F	775
G	773
H	734
Y	755
J	747
K	739
L	761
M	792
N	783
O	753
P	803
Q	783
R	758
S	748
T	796
U	813
V	764
W	752
X	787
Y	786
Z	734
Total	20000

Tabla 6: Distribución total de datos por clase del dataset Letter.

Conformado por un set de 15.000 datos de entrenamiento y 5.000 de prueba. Los experimentos fueron realizados en un computador con Pentium Centrino 1.6 GHz y 512 MB de RAM y se pueden observar sus resultados en Tabla 7. Los autores destacan que un sistema OAO tiende a calzar mucho mejor para problemas con un gran número de clases. Cuando las clases incrementan demasiado, como en este caso, el tiempo de entrenamiento de un sistema OAO se cuadriplica en comparación a una sola red para K clases, producto del overhead requerido por la gran cantidad de redes en un sistema OAO. Este comportamiento no se aprecia en el dataset Mnist de 10 clases.

	OAA, 26 redes	OAO, 325 redes	OHAO, 25 redes	OAA, 1 red
Accuracy (%)	86.34	93.96	89.58	87.38
Hidden Nodes	15	15	15	40
Tiempo de entrenamiento (h)	2	2.3	1.16	0.45

Tabla 7: Rendimiento el dataset “Letter”. en distintos sistemas de redes neuronales.

- **Iris, Wine, Vehicle y PDB**

  Iris, Wine y Vehicle son pequeños dataset de flores iris, vinos y distintos vehículos como dice su nombre en ingles. PDB es un dataset de banco de proteínas (PDB por sus siglas en inglés) desbalanceado. En la comunidad del aprendizaje automático, se tiene la preocupación que los sistemas clasificadores para dataset grandes no se comportaran bien para dataset pequeños y viceversa. Los autores estudian el comportamiento anterior al comparar dataset pequeños como Iris que tiene 150 datos divididos en 3 clases balanceadamente, Wine con 59, 71 y 48 datos de ejemplos en sus tres clases respectivamente y Vehicle, el cual posee 240 ejemplos para 3 clases y 225 ejemplos para una cuarta clase. Como no se tiene un set de pruebas, los resultados obtenidos en Tabla 8 son producto de un cross validation con 10 folds.

	OAA, 3 redes	OAO, 3 redes	OAHO, 2 redes	OAA, 1 red
Iris Hidden Nodes	10	5	10	20
Iris Accuracy (%)	96	98	96	98
Wine Hidden Nodes	5	5	5	10
Wine Accuracy (%)	98.31	97.75	98.31	98.31
Vehicle Hidden Nodes	15	15	15	40
Vehicle Accuracy (%)	84.75	83.69	84.16	83.92

Tabla 8: Precisión de clasificación de cuatro redes neuronales sobre los pequeños datasets “Iris”, “Wine” y “Vehicle”.

En el dataset PDB, se tiene la secuencia de proteínas $p = p_1, p_2, ..., p_m$ donde p_i es un aminoácido cuya estructura secundaria puede ser categorizada como helix, strand y coil. El set de entrenamiento se compone de 110.530 datos extraídos de 300 secuencias de proteínas y el set de pruebas consiste en 110.530 datos extraídos de 100 secuencias de proteínas. Cada dato consiste en 19 aminoácidos y representado con un vector de características de 95 dimensiones. La tabla 9 muestra los resultados obtenidos al trabajar con las mismas arquitecturas que en iris, wine y vehicle.

	OAA, 3 redes	OAO, 3 redes	OAHO, 2 redes	OAA, 1 red
Hidden Nodes	20	40	60 & 30	60
Accuracy (%)	65.64	80.35	48.88	65.14
Tiempo de entrenamiento (h)	21	4.32	30.6	90

Tabla 9: Accuracy de clasificación de cuatro redes neuronales sobre el gran dataset “PDB”.

En el sistema OAHO, un nodo fue entrenado con 60 hidden nodes y se realizó un modelado de clase 1 en contra de las clases 2 y 3, y el segundo nodo fue modelado con 30 hidden nodes y fue modelado con clase 2 en contra de la clase 3. Los autores destacan que el tiempo de entrenamiento esta relacionado al número de epochs de entrenamiento. Los autores usan 1000 epochs para todos los sistemas.

CAPÍTULO 3
PROPUESTA DE SOLUCIÓN

La propuesta de solución es investigar el uso de la arquitectura GAD usada originalmente para SVMs en [4] y adaptarla para que en vez de utilizar SVM utilice en cada nodo una red convolucional binaria.

La arquitectura GAD (gráfico acíclico dirigido), fue propuesta en el año 2000 [4]. La arquitectura GAD es un grafo que solamente tiene una orientación y no posee ciclos, tiene un nodo raíz y nodos hojas, por lo cual su estructura es similar a un árbol binario. La idea fundamental de esta arquitectura es aplicar la metodología de dividir y vencerás a un problema de clasificación de múltiples clases, generando K(K-1)/2 nodos, siendo K la cantidad de clases.

Cada nodo de nuestro grafo GAD será una red convolucional binaria que será entrenada para descartar una de dos clases. Parte de la información de salida de cada red serán los datos de entrada de las siguientes redes en una capa inferior. Al terminar este proceso, se genera la clasificación final en los nodos hojas. Para entrenar esta nueva arquitectura GAD, se entrenará cada nodo independientemente entre sí. Se considerarán tres formas posibles de hacer el entrenamiento:

- **Entrenamiento desde 0:** Utilizando la misma arquitectura de red para cada nodo de nuestra arquitectura GAD, se entrenará desde cero (pesos iniciales aleatorios) cada nodo, individualmente, para descartar una de dos clases. Este método derivada en el experimento 1.
- **Fine Tunning:** Cada nodo se entrenará individualmente para descartar una de dos clases, al igual que el entrenamiento desde 0, pero los pesos iniciales serán los de una red pre-entrenada cuyo propósito es clasificar el mismo dataset pero con una metodología de N = K nodos de salida. Este método derivará en el experimento 2.
- **Fine Tunning con capas congeladas:** Para este caso, los pesos iniciales de cada nodo serán otorgados por una red ya pre-entrenada al igual que en el entrenamiento GAD anterior, pero donde todas las capas de la red serán congeladas, a excepción de la última capa, por lo cual, cada nodo entrenará únicamente una capa de la red.

Como se explicó en la sección 2.3.1, el orden de la lista de clases que se le entrega a la arquitectura GAD es importante, pero como el enfoque de esta investigación es comprobar cómo se comporta el accuracy general y por clase de ciertos datasets utilizando el método GAD, una investigación sobre el orden de la lista de clases significaría escapar del objetivo principal de la memoria. Por lo tanto, se utilizará simplemente un orden aleatorio, con la misma ‘semilla” aleatoria para modificar todas las listas de clases en la experimentación.

CAPÍTULO 4
VALIDACIÓN DE LA SOLUCIÓN

En este proyecto se intenta comprobar la eficacia de dividir un problema de clasificación múltiple en varios problemas de clasificación binaria, para ello se realizarán 3 tipos de experimentos sobre 3 dataset distintos. Para comprobar si esto realmente funciona o no, se compararán los resultados obtenidos en los experimentos con dos benchmarks. El primero corresponde a usar una red única con arquitectura seleccionada del estado del arte de cada dataset. El segundo corresponde a una técnica de descomposición seleccionada del estado del arte presentado en el capítulo 2. Por último, se estudiará el comportamiento del modelo propuesto en dos problemas de clases desbalanceadas: uno generado sintéticamente y otro desbalanceado naturalmente.

4.1. Datasets

- MNIST [13]: Colección balanceada de 70,000 imágenes de dígitos del 0 al 9 escritos a mano las cuales tienen dimensiones 28x28x1 píxeles. Este dataset cuenta con una división de 60.000 imágenes de entrenamiento y 10.000 imágenes de prueba.

- CIFAR-10 [14]: Consiste en un conjunto de 60.000 imágenes RGB de 10 clases diferentes las cuales son: gatos, perros, ranas, caballos, pájaros, ciervos, aviones, automóviles, camiones y barcos. Todas las imagenes estan balanceadas. Estas imágenes tienen una dimensión de 32x32x3 píxeles y el dataset cuenta con una división de 50.000 imágenes de entrenamiento y 10.000 imágenes de prueba

- NORB [15]: Colección de Imágenes balanceadas de 108x108x2 píxeles de 50 juguetes que pertenecen a 6 categorías genéricas, las cuales son: animales de cuatro patas, figuras humanas, aviones, camiones, autos y una sexta categoría para imágenes sin objetos. Los objetos en cuestión fueron fotografiados por dos cámaras bajo 6 condiciones de iluminación, 9 ángulos de elevación (desde 30 a 70 grados, con aumentos de 5 grados) y 18 ángulos azimutales (desde 0 a 340 grados, con aumentos de 20 grados). Este dataset contiene un conjunto de entrenamiento separado en 10 folds. Cada fold contiene 29.160 imágenes de entrenamiento (6 categorías, 5 ejemplos, 6 tipos de iluminación, 9 tipos de elevación y 18 variaciones azimutales) y 29.160 imágenes de pruebas. Sin embargo, en la literatura actual, se suele utilizar solo los 2 primeros folds.

- Best Artworks of All Time (BAAT) [16]: Imágenes de obras de arte de 50 pintores famosos. Las imágenes tienen dimensiones 1280x927x3 y vienen en formato RBG. Por temas de no incrementar en exceso la cantidad de nodos del grafo GAD, se trabajó solamente con las 10 clases con mayor cantidad de imágenes, resultando en 4060 imágenes de entrenamiento perteneciente a 10 artistas diferentes.

Class	Data
Vincent van Gogh	877
Edgar Degas	702
Pablo Picasso	439
Pierre-Auguste Renoir	336
Albrecht Durer	328
Paul Gauguin	311
Francisco Goya	291
Rembrandt	262
Alfred Sisley	259
Titian	255

Tabla 10: Distribución de data de BAAT.

Como no se tiene un conjunto de prueba se dividió el dataset en dos, tomando aleatoriamente un 20 % de los datos por clase, quedando 3252 datos de entrenamiento y 808 datos de prueba.

Dataset	Datos de Entrenamiento	Datos de Prueba	Dimensiones (Filas x Columnas x Canales)	Clases
MNIST	60.000	10.000	28x28x1	10
CIFAR-10	50.000	10.000	32x32x3	10
NORB	29.160	29.160	108x108x2	6
BAAT	3252	808	1280x927x3	10

Tabla 11: Especificaciones importantes de dataset a utilizar para la validación de la memoria.

4.2. Arquitecturas y Benchmark

Para poder validar nuestra solución, se tiene que escoger una métrica y con ello, un punto de partida para todos los dataset con el objetivo de comprobar si nuestra propuesta supera o no una cierta barrera establecida. Para ello se definirán dos arquitecturas que marcarán nuestro punto de partida, la arquitectura maestra y la arquitectura básica [6]. La arquitectura maestra es el SOTA de su respectivo dataset y la arquitectura básica es un derivado simplificado de la arquitectura maestra la cual contiene capas y estructuras más tradicionales. Además de estas dos arquitecturas, se compararán los resultados con otros dos métodos de descomposición: OAHO (descomposición en cadena) y OAO (one-against-one) con mayoría de votos, ambos especificados en [5]. Esta selección se debe a que ambos métodos son de descomposición y son similares a la propuesta entregada, por lo cual se puede realizar una comparación de metodos.

4.2.1. MNIST

4.2.2. Arquitectura Maestra

La estructura maestra escogida es la red DNN-W20 [2]. El modelo está compuesto por las siguientes capas en orden de aparición:

Nombre	Neuronas	Tamaño Filtro	Fun. de Act.	Tamaño Salida
Conv2D	20	4x4	Scaled Tanh	25x25x20
MaxPooling2D	-	2x2	-	12x12x20
Conv2D	40	5x5	Scaled Tanh	8x8x40
MaxPooling2D	-	3x3	-	2x2x40
Flatten	-	-	-	160
Dense	150	-	Scaled Tanh	150
Dense	10	-	Softmax	10

Tabla 12: MNIST Arquitectura maestra, red DNN-W20.

Las imágenes son pre-procesadas aplicando una normalización a los 20x20 pixeles en los cuales se encuentra el número. Los pesos de la red son iniciados en un rango de [-0.05, 0.05] mediante una distribución uniforme. Se utilizó la misma función de activación para todas las capas ocultas, la cual es una tangente hiperbólica escalada:

$$f(x) = 1,7159 \cdot tanh\left(\frac{2x}{3}\right). \tag{1}$$

El modelo es entrenado con una tasa de aprendizaje inicial de $0,001$, la que es multiplicada por un factor de $0,993$ hasta que alcanza el valor $0,00003$, lo que lleva a la ejecución de 500 epochs. No hay uso de momentum o weight decay. Por temas de memoria se hizo uso de SGD con tamaño de mini-batch 128 en vez de utilizar Gradiente Descendiente Online como en el paper original. Se tiene que tomar en cuenta que en el trabajo original se distorsionaron ejemplos de entrenamiento escogidos aleatoriamente previo a cada epoch por medio de transformaciones, rotaciones y cambios de escala, lo que no es hecho en este trabajo.

El accuracy reportado por los autores de esta arquitectura es del $99{,}61$ % y el accuracy obteido fue de $99{,}11$ %. Por último el accuracy por clase obtenido es:

Clase	Accuracy %
0	98.68
1	99.12
2	98.50
3	98.23
4	98.67
5	99.21
6	99.16
7	98.85
8	99.14
9	99.10
prom	98.87

Tabla 13: Accuracy por clase para el dataset MNIST usando la arquitectura maestra sin descomposición.

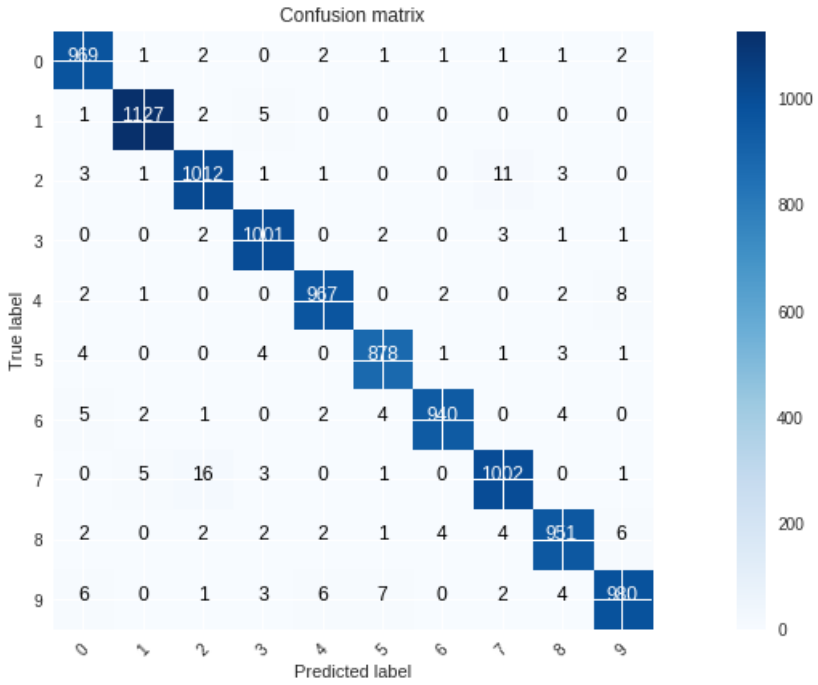


Figura 11: Matriz de confusión de las clases de MNIST para la arquitectura maestra sin descomposición.

Para la descomposición en cadena (OAHO) se mantuvo la misma cantidad de epochs, se utilizó 1 neurona de salida, función de activación sigmoid y una función de pérdida binary cross entropy para cada nodo y se obtuvo un benchmark de:

Clase	Accuracy %
0	98.37
1	99.03
2	96.22
3	97.03
4	97.97
5	96.07
6	96.86
7	95.52
8	95.58
9	94.35
prom	96.70

Tabla 14: Benchmark MNIST arquitectura maestra con descomposición en cadena.

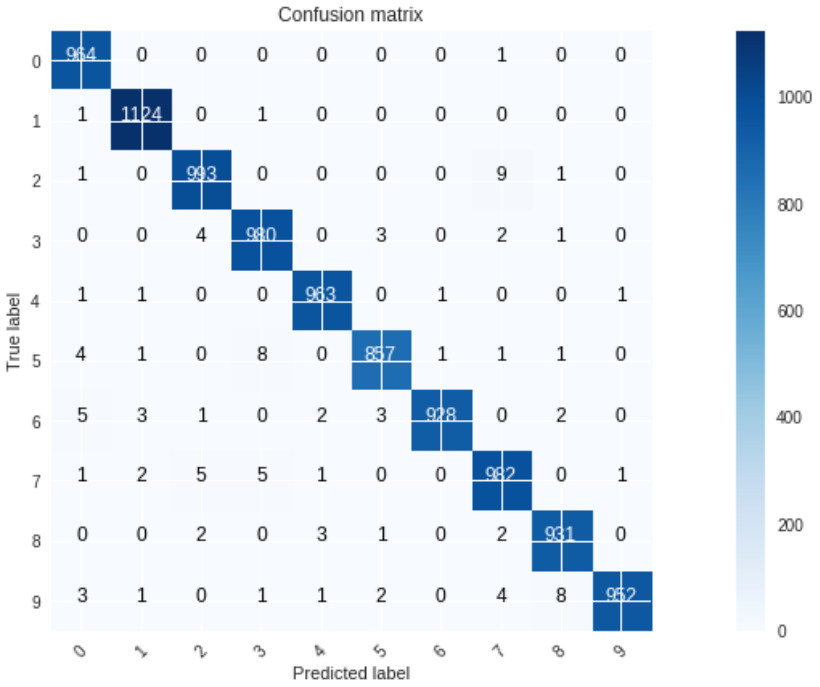
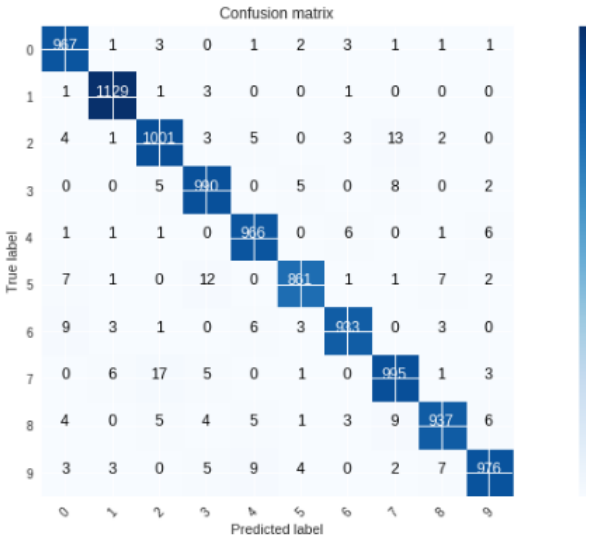


Figura 12: Matriz de confusión de las clases de MNIST para la arquitectura maestra con descomposición en cadena.

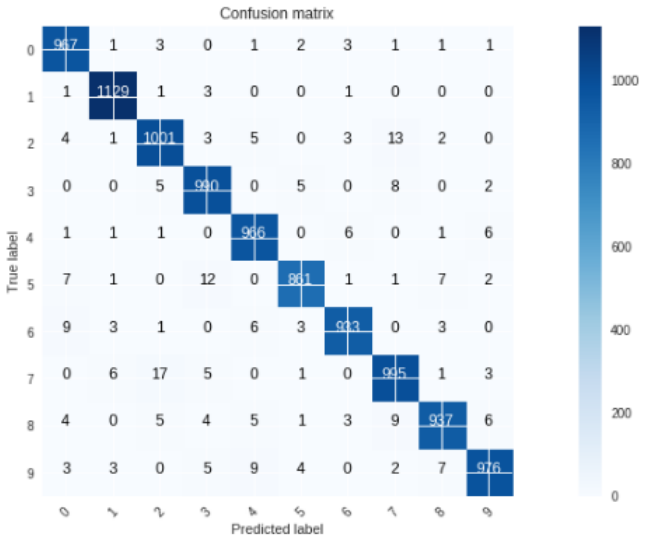
Finalmente, para la descomposición OAO con mayoría de votos, se usaron K(K-1)/2 redes convolucionales para generar la votación de cada ejemplo. En cada red se mantuvo la misma cantidad de epochs, se utilizó 1 neurona de salida, función de activación sigmoid y una función de pérdida binary cross entropy. Se realizó el entrenamiento de las tres formas anteriormente explicadas anteriormente. Luego de la votación y la corrección de empates de manera aleatoria, tenemos los siguientes resultados:

Clase	Accuracy % Training Desde 0	Accuracy % Fine Tunning	Accuracy % Fine Tunning Capas Congeladas
0	98.38	98.36	97.48
1	99.02	98.36	98.72
2	96.85	96.68	96.81
3	96.48	96.38	97.87
4	97.29	97.13	97.89
5	96.41	96.48	96.44
6	97.27	96.38	97.68
7	96.64	96.07	97.53
8	96.16	97.47	98.15
9	96.63	97.86	98.19
prom	97.11	97.12	97.68

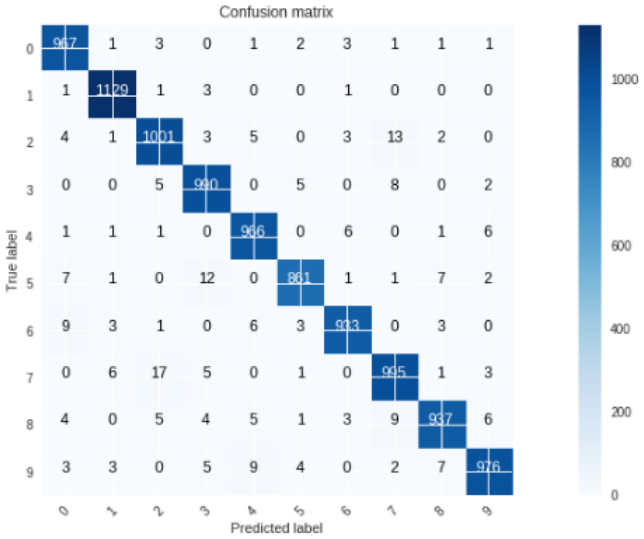
Tabla 15: Benchmark MNIST para la arquitectura maestra con descomposición OAO con mayoría de votos.



(a) Entrenamiento desde 0.



(b) Solo Fine Tunning.



(c) Fine Tunning con capas congeladas.

Figura 13: Matriz de confusión de las clases de MNIST para la arquitectura maestra con descomposición OAO con mayoría de votos.

4.2.3. Arquitectura Básica

Se utilizó una arquitectura CNN-Softmax [17] con la siguiente arquitectura:

Nombre	Neuronas	Tamaño Filtro	Fun. de Act.	Tamaño Salida
Conv2D	32	5x5	Relu	24x24x32
MaxPooling2D	-	2x2	-	12x12x32
Conv2D	64	5x5	Relu	8x8x64
MaxPooling2D	-	2x2	-	4x4x64
Flatten	-	-	-	1024
Dense	500	-	Relu	500
Dropout	-	-	-	500
Dense	10	-	Softmax	10

Tabla 16: MNIST arquitectura basica, red DNN-W20.

Las imágenes son preprocesadas mediante la técnica GCN [6]. Para entrenar este modelo, se utiliza SGD con mini-batch de 128 durante 110 epochs, incluyendo weight decay de $0,0005$, momentum fijo de $0,9$ y learning rate fijo de $0,01$. Finalmente, se utilizó una capa de Dropout con un valor de $0,5$ para proteger al modelo del sobreajuste.

En promedio, se reporta que esta arquitectura logra un $99{,}44$ % de accuracy [6] mientras que los experimentos realizados indican un rendimiento de $98{,}85$ %. Los resultados por clase se muestran a continuación:

Clase	Accuracy %
0	97.02
1	97.92
2	99.41
3	98.05
4	98.98
5	98.76
6	99.89
7	97.33
8	99.58
9	98.88
prom	98.58

Tabla 17: Accuracy por clase para el dataset MNIST usando la arquitectura básica sin descomposición.

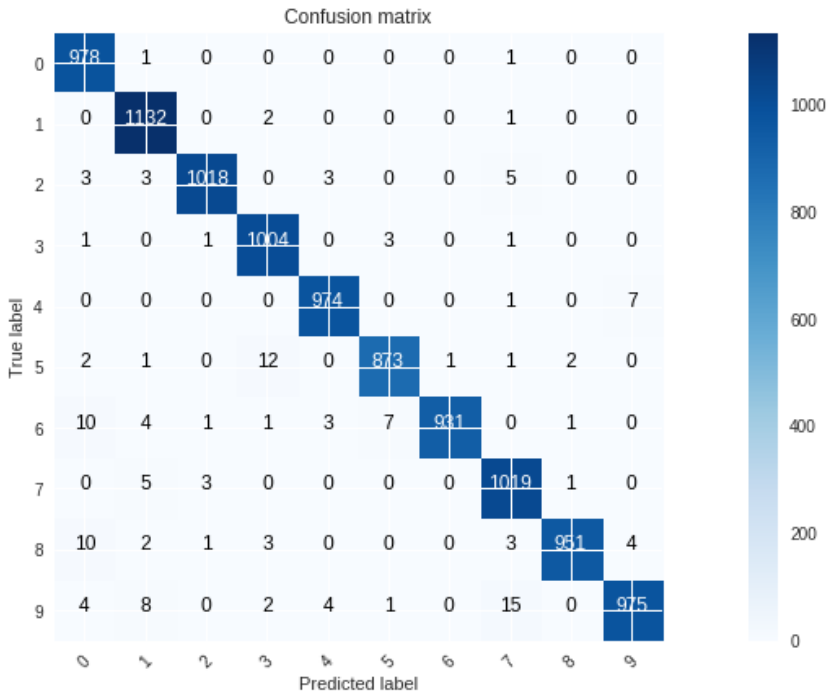


Figura 14: Matriz de confusión de las clases de MNIST para la arquitectura básica sin descomposición.

Para la descomposición en cadena de la arquitectura básica se mantuvo la cantidad de epochs, se utilizó 1 neurona de salida, función de activación sigmoid y una función de perdida binary cross entropy para cada nodo y se obtuvo un benchmark de:

Clase	Accuracy %
0	97.35
1	99.38
2	97.87
3	98.51
4	98.48
5	98.32
6	97.60
7	97.18
8	98.05
9	95.24
prom	97.08

Tabla 18: Benchmark MNIST para la arquitectura básica con descomposición en cadena.

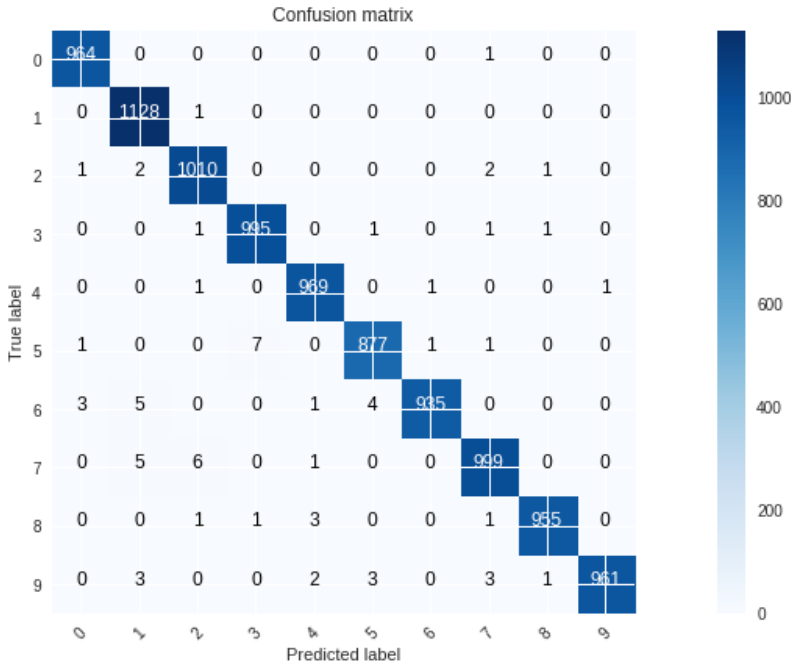
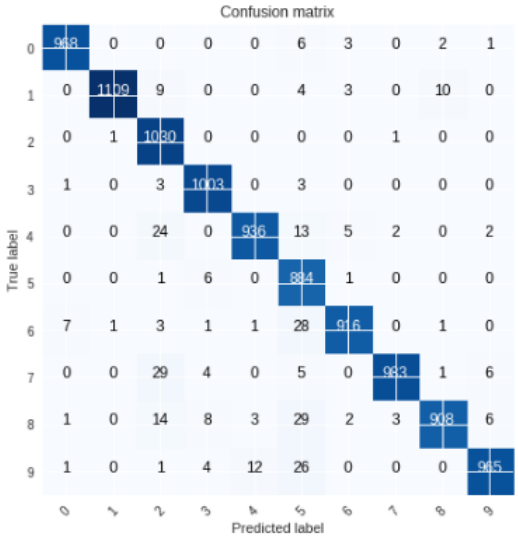


Figura 15: Matriz de confusión de las clases de MNIST para la arquitectura básica con descomposición en cadena.

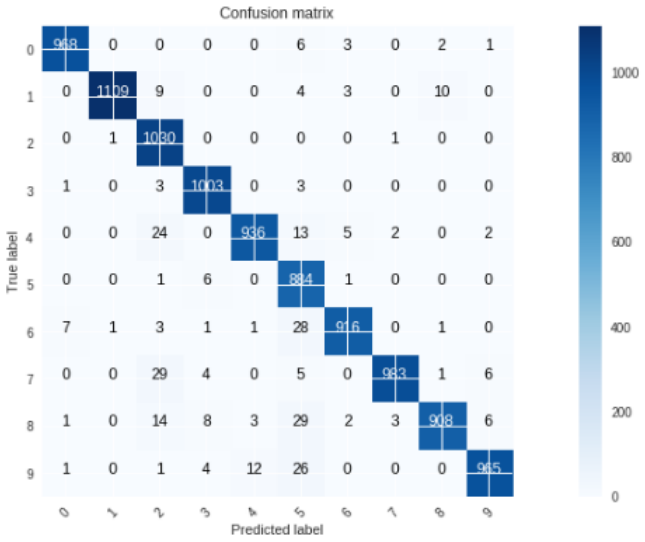
Por último, para la descomposición OAO con mayoría de votos, se usaron K(K-1)/2 redes convolucionales para generar la votación de cada ejemplo. Se utilizó 1 neurona de salida, función de activación sigmoid y una función de pérdida binary cross entropy para cada nodo. Se realizaron los tres tipos de entrenamiento, lo que resultó en los resultados de la Tabla 19:

Class	Accuracy % Training Desde 0	Accuracy % Fine Tunning	Accuracy % Fine Tunning Capas Congeladas
0	98.37	98.38	98.75
1	99.57	97.57	98.35
2	92.23	99.18	99.82
3	97.56	99.26	99.38
4	98.84	95.39	95.54
5	88.37	99.74	98.18
6	98.56	95.57	96.91
7	99.64	95.24	95.17
8	98.79	93.36	92.46
9	98.27	95.15	95.64
prom	97.02	96.88	97.02

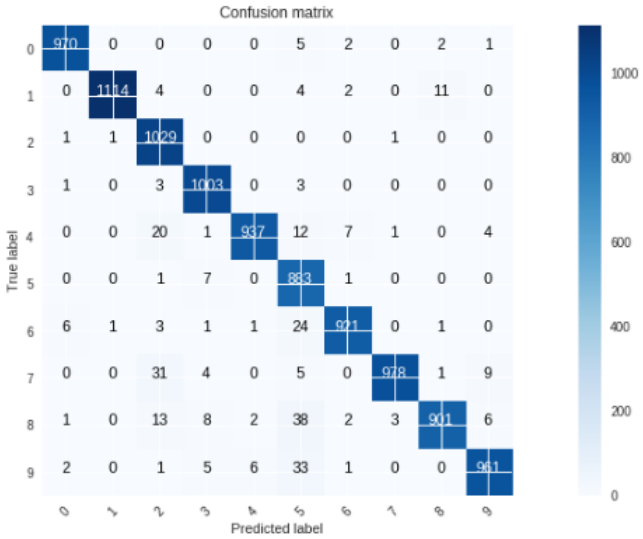
Tabla 19: Benchmark MNIST para la arquitectura básica con descomposición OAO con mayoría de votos.



(a) Entrenamiento desde 0.



(b) Solo Fine Tunning.



(c) Fine Tunning con capas congeladas.

Figura 16: Matriz de confusión de las clases de MNIST para la arquitectura básica con descomposición OAO con mayoría de votos.

4.2.4. NORB

NORB es un caso especial, ya que una de sus clases representa un objeto que no se encuentra en las otras 5 categorías, por lo cual esta se omitirá por que esta clase “de relleno” no se encuentra en los otros datasets, por lo cual se elimina para hacer más consistentes todos los experimentos a lo largo de todos los datasets. Entonces, el dataset NORB termina

finalmente con 24.300 datos de entrenamiento y 24.300 datos de prueba. Por otra parte, esto también acelera los experimentos, ya que una clase menos significa que nuestro grafo es menos profundo por lo cual se tienen que entrenar menos nodos.

4.2.5. Arquitectura Maestra

Para este dataset, se utilizó la red propuesta en [18]. Esta red toma beneficios del Average Pooling y la función de activación Relu. Los autores de [6] la denominan AP-Relu por lo cual utilizaremos la misma denominación. La red está compuesta por lo siguiente:

Nombre	Neuronas	Tamaño Filtro	Fun. de Act.	Tamaño Salida
Conv2D	64	5x5	Relu	96x96x64
AveragePooling2D	-	3x3	-	32x32x64
LRN2D	-	-	-	32x32x64
Conv2D	64	5x5	Relu	32x32x64
AveragePooling2D	-	3x3	-	10x10x64
LRN2D	-	-	-	10x10x64
Conv2D	64	5x5	Relu	10x10x64
AveragePooling2D	-	3x3	-	3x3x64
Flatten	-	-	-	576
Dense	500	-	Relu	500
Dropout	-	-	-	500
Dense	5	-	Softmax	5

Tabla 20: NORB Arquitectura maestra, AP-Relu.

Las imágenes en sí no fueron pre-procesadas pero se escalaron a un tamaño de 98x98 píxeles. En cada una de las dos primeras capas de pooling hay una capa de LRN [6]. El modelo es entrenado con SGD como método de optimización durante 250 epochs, con tamaño de mini-batch 1284, weight decay de $0,01$ y momentum de $0,9$. Se utiliza una tasa de aprendizaje inicial de $0,001$ que es multiplicada por un factor de $0,1$ cada 100 epochs.

Se reporta que esta red alcanza un accuracy total del $94{,}7\,\%$, mientras que el accuracy alcanzado experimentalmente fue del $90\,\%$ en promedio y el accuracy por clase se muestra a continuación:

Clase	Accuracy %
Animal	81.83
Human	96.20
Airplane	98.08
Truck	88.29
Car	84.12
prom	89.70

Tabla 21: Accuracy por clase para el dataset NORB usando la arquitectura maestra sin descomposición.

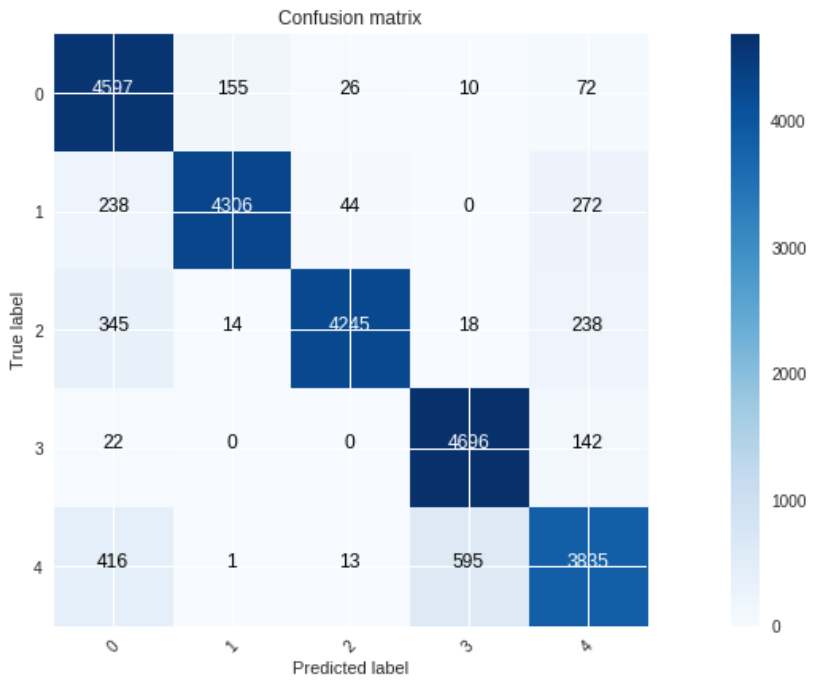


Figura 17: Matriz de confusión de las clases de NORB para la arquitectura maestra sin descomposición.

Además, el benchmark de la arquitectura maestra descompuesta en forma de cadena es:

Clase	Accuracy %
Animal	84.18
Human	85.23
Airplane	76.93
Truck	91.79
Car	59.88
prom	79.60

Tabla 22: Benchmark NORB para la arquitectura maestra con descomposición en cadena.

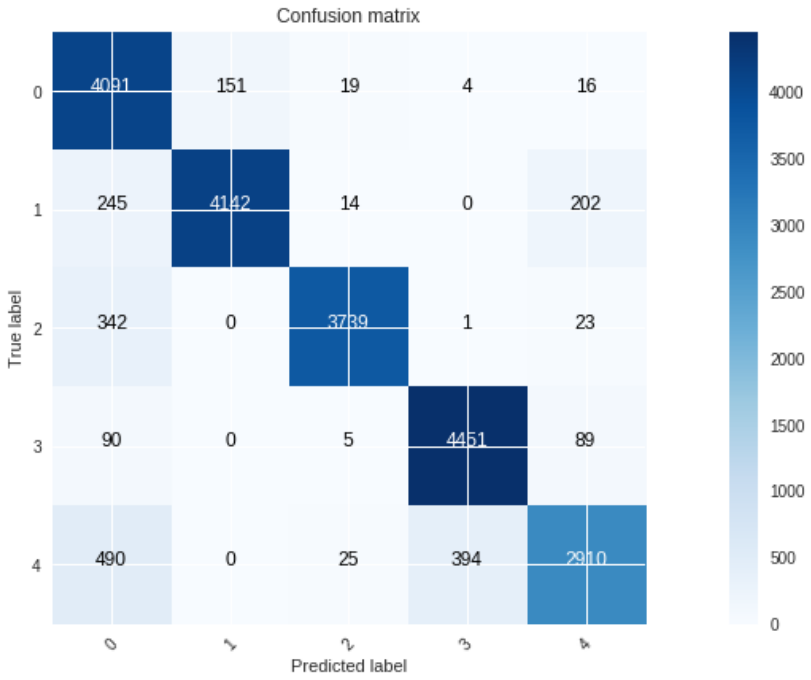
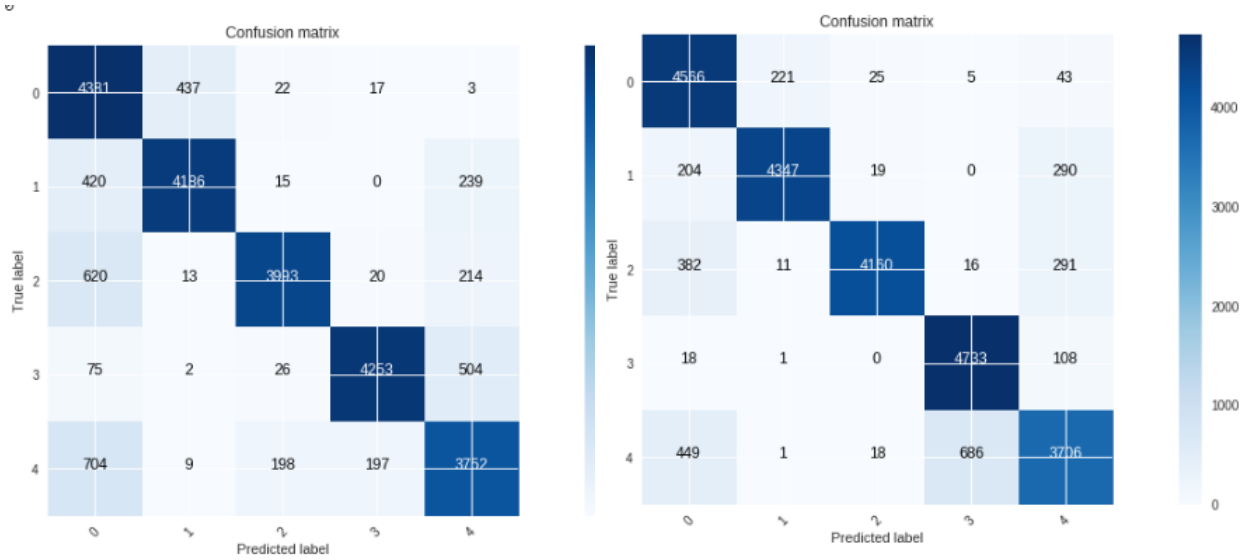


Figura 18: Matriz de confusión de las clases de NORB para la arquitectura maestra con descomposición en cadena.

Por último, al utilizar la misma arquitectura pero con descomposición OAO con mayoría de votos y realizando los tres tipos de entrenamientos posibles, tenemos un benchmark de:

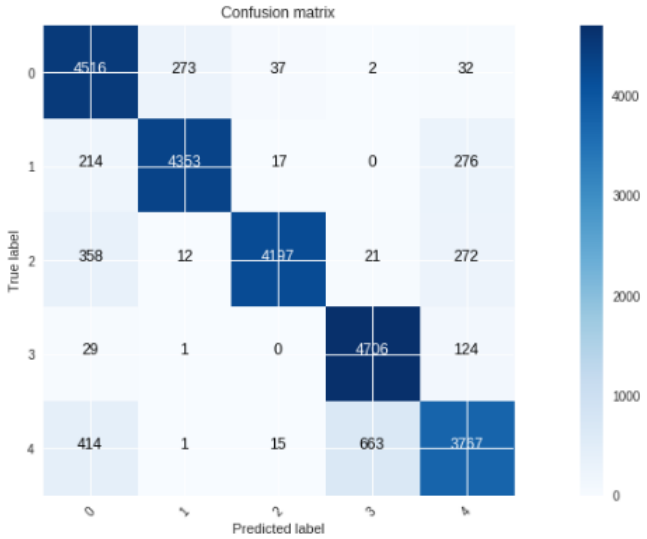
Class	Accuracy % Training Desde 0	Accuracy % Fine Tunning	Accuracy % Fine Tunning Capas Congeladas
Animal	90.47	92.58	90.79
Human	86.85	89.86	89.51
Airplane	82.13	85.14	86.48
Truck	87.68	97.93	96.37
Car	77.79	76.29	77.49
prom	84.98	88.36	88.13

Tabla 23: Benchmark NORB para la arquitectura maestra con descomposición OAO con mayoría de votos.



(a) Entrenamiento desde 0.

(b) Solo Fine Tunning.



(c) Fine Tunning con capas congeladas.

Figura 19: Matriz de confusión de las clases de NORB para la arquitectura maestra con descomposición OAO con mayoría de votos.

4.2.6. Arquitectura Básica

Se vuelve a utilizar la red propuesta en [18], pero se reemplazan las Average Pooling por Max Pooling y se cambió el pool size a una ventana de 2x2. Con estas modificaciones la red se denomina MP-Relu [6]. Las consideraciones tomadas respecto al pre-procesamiento de los datos, funciones de activación utilizadas, método de optimización, tamaño de mini-batch,

cantidad de epochs de entrenamiento, tasa de aprendizaje, weight decay y momentum se mantienen con respecto a la arquitectura maestra:

Nombre	Neuronas	Tamaño Filtro	Fun. de Act,	Tamaño Salida
Conv2D	64	5x5	Relu	96x96x64
MaxPooling2D	-	2x2	-	48x48x64
LRN2D	-	-	-	48x48x64
Conv2D	64	5x5	Relu	48x48x64
MaxPooling2D	-	2x2	-	24x24x64
LRN2D	-	-	-	24x24x64
Conv2D	64	5x5	Relu	24x24x64
MaxPooling2D	-	2x2	-	12x12x64
Flatten	-	-	-	9216
Dense	500	-	Relu	500
Dropout	-	-	-	500
Dense	5	-	Softmax	5

Tabla 24: NORB Arquitectura básica, MP-Relu.

Esta arquitectura alcanza un accuracy del $94{,}12$ % según [6], mientras que experimentalmente se alcanzó un accuracy del 89 %. El accuracy por clase es:

Clase	Accuracy %
Animal	83.03
Human	96.76
Airplane	96.46
Truck	86.57
Car	86.36
prom	89.84

Tabla 25: Accuracy por clase para el dataset NORB usando la arquitectura básica sin descomposición.

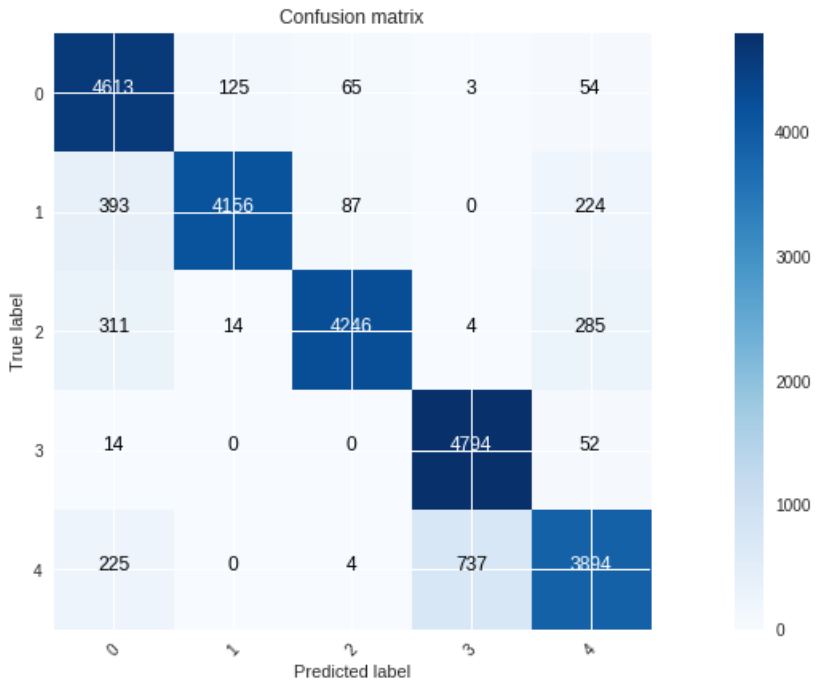


Figura 20: Matriz de confusión de las clases de NORB para la arquitectura básica sin descomposición.

El benchmark de la arquitectura básica descompuesta en forma de cadena (OAHO) es:

Clase	Accuracy %
Animal	84.18
Human	77.45
Airplane	76.85
Truck	91.56
Car	57.82
prom	77.57

Tabla 26: Benchmark NORB para la arquitectura básica con descomposición en cadena.

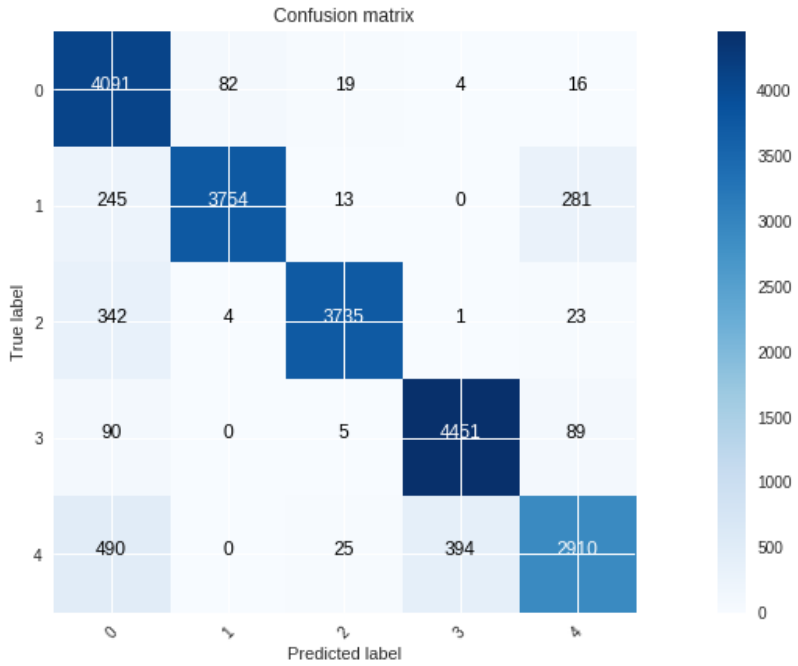
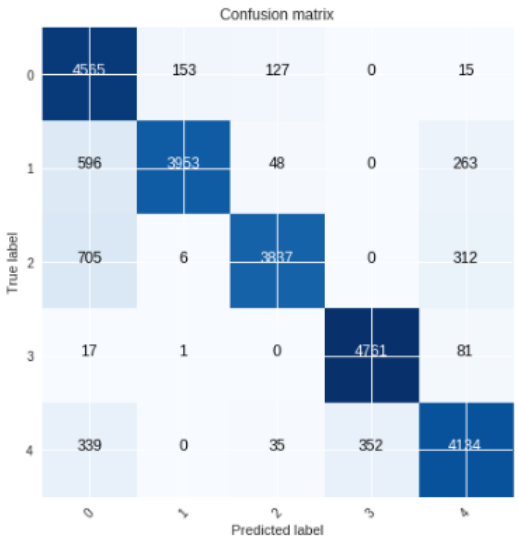


Figura 21: Matriz de confusión de las clases de NORB para la arquitectura básica con descomposición en cadena.

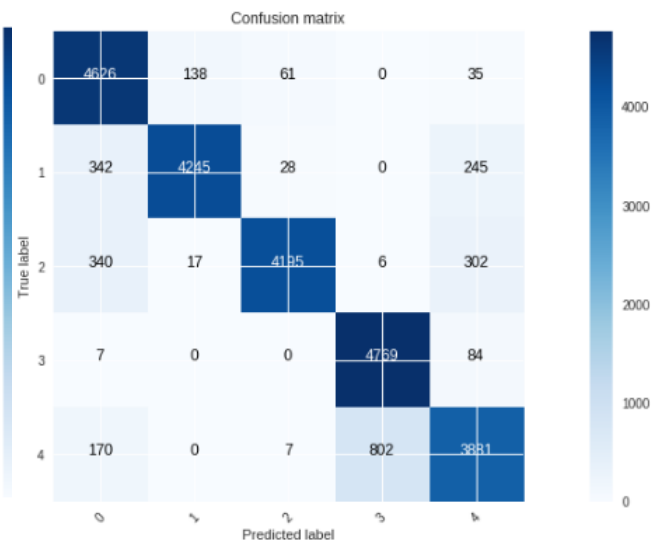
Por último, con una descomposición OAO con mayoría de votos en los tres entrenamientos anteriormente estipulados, se obtiene lo siguiente:

Class	Accuracy % Training Desde 0	Accuracy % Fine Tunning	Accuracy % Fine Tunning Capas Congeladas
Animal	93.91	94.16	93.13
Human	81.32	87.37	86.49
Airplane	78.84	86.38	85.17
Truck	97.73	96.32	96.67
Car	85.13	79.78	74.91
prom	87.39	88.80	87.27

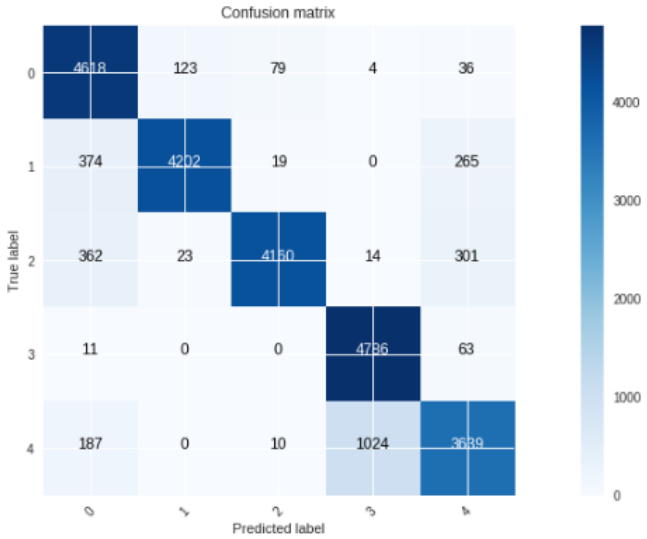
Tabla 27: Benchmark NORB para la arquitectura básica con descomposición OAO con mayoría de votos.



(a) Entrenamiento desde 0.



(b) Solo Fine Tunning.



(c) Fine Tunning con capas congeladas.

Figura 22: Matriz de confusión de las clases de NORB para la arquitectura básica con descomposición OAO con mayoría de votos.

4.2.7. CIFAR-10

4.2.8. Arquitectura Maestra

Se utilizará la arquitectura All-CNN-C propuesta en [19]:

Nombre	Neuronas	Tamaño Filtro	Fun. de Act	Pasos	Tamaño salida
Dropout	-	-	-	-	32x32x3
Conv2D	96	3x3	Relu	-	32x32x96
Conv2D	96	3x3	Relu	-	32x32x96
Conv2D	96	3x3	Relu	2x2	16x16x96
Dropout	-	-	-	-	16x16x96
Conv2D	192	3x3	Relu	-	16x16x192
Conv2D	192	3x3	Relu	-	16x16x192
Conv2D	192	3x3	Relu	2x2	8x8x192
Dropout	-	-	-	-	8x8x192
Conv2D	192	3x3	Relu	-	8x8x192
Conv2D	192	1x1	Relu	-	8x8x192
Conv2D	10	1x1	Relu	-	8x8x10
GlobalAveragePooling2D	-	-	-	-	10
Dense	10	-	Softmax	-	10

Tabla 28: CIFAR-10 arquitectura maestra, red All-CNN-C.

Las imágenes son preprocesadas aplicando GCN en primer lugar y luego ZCA whitening. El modelo es entrenado con SGD durante 350 epochs. No se indica el tamaño de mini-batch considerado, por lo que es fijado en 100. La tasa de aprendizaje se experimentó con los valores $0,25$, $0,1$, $0,05$ y $0,01$. Toda tasa es multiplicada por un factor de $0,1$ después de 200, 250 y 300 epochs. Se usó momentum fijo de $0,9$. Para regularizar el modelo, se aplica dropout de $0,2$ sobre la capa de entrada y de $0,5$ para cada capa convolucional de stride 2. El modelo también es regularizado con weight decay de $0,001$ y finalmente se tomo una taza de aprendizaje de $0,05$ la cual se multiplicó por $0,1$ en el apoch 200, 250 y 300.

El accuracy reportado por los autores de esta arquitectura fue de $90{,}82$ % y experimentalmente se llegó a un valor de $87{,}05$ % donde el accuracy por clase es:

Clase	Accuracy %
Airplane	88.01
Automobile	92.59
Bird	80.53
Cat	70.83
Deer	85.94
Dog	82.44
Frog	87.45
Horse	92.58
Ship	89.19
Truck	92.52
prom	86.21

Tabla 29: Accuracy por clase para el dataset CIFAR10 usando la arquitectura maestra sin descomposición.

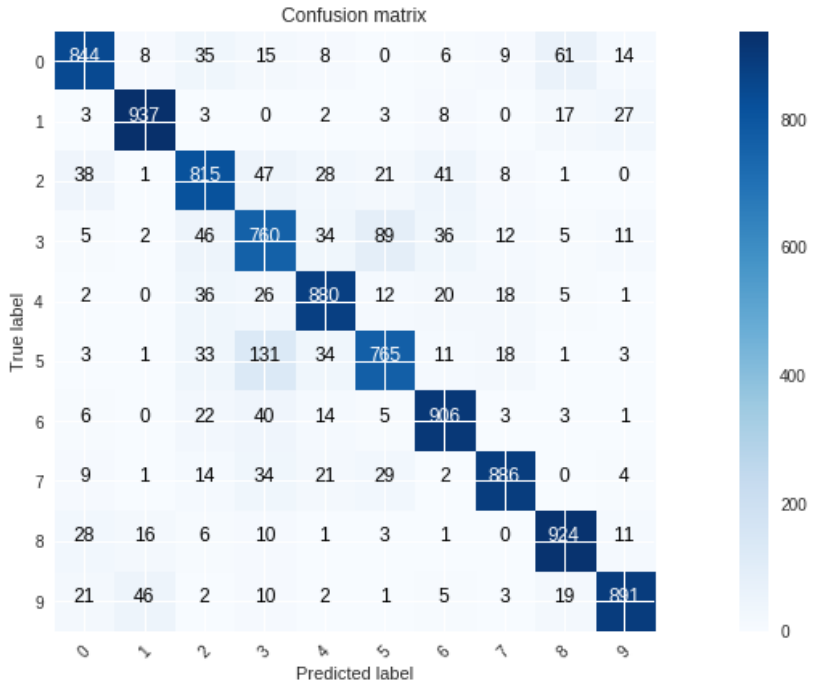


Figura 23: Matriz de confusión de las clases de CIFAR10 para la arquitectura maestra sin descomposición.

Para la descomposición en cadena de la arquitectura maestra de CIFAR-10, se mantuvo lo realizado con las otras descomposiciones en cadena lo cual dio un benchmark de:

Clase	Accuracy %
Airplane	73.90
Automobile	89.90
Bird	74.30
Cat	70.70
Deer	79.80
Dog	76.60
Frog	85.8
Horse	77.80
Ship	82.60
Truck	85.70
prom	79.71

Tabla 30: Benchmark CIFAR-10 para la arquitectura maestra con descomposición en cadena.

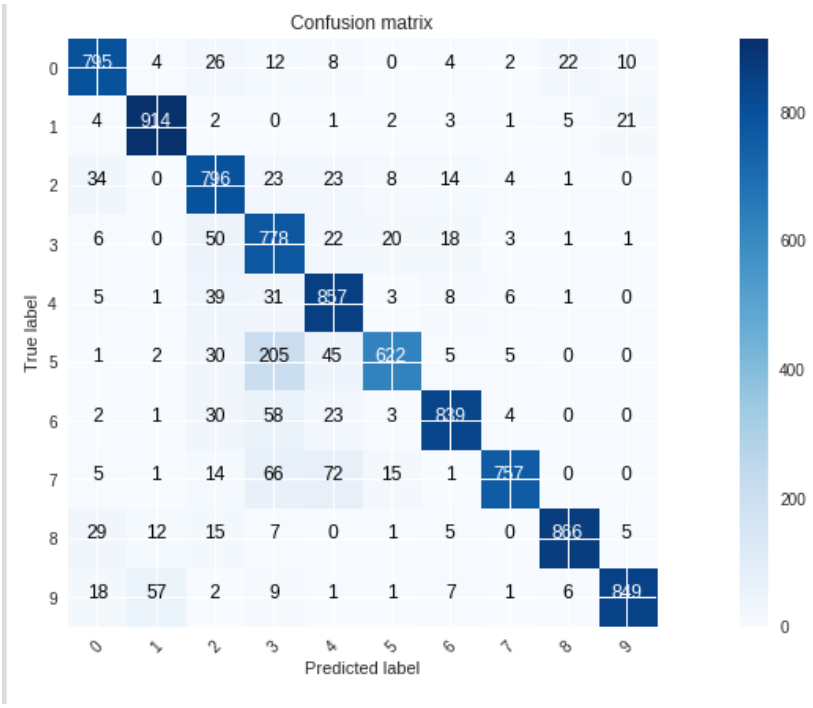
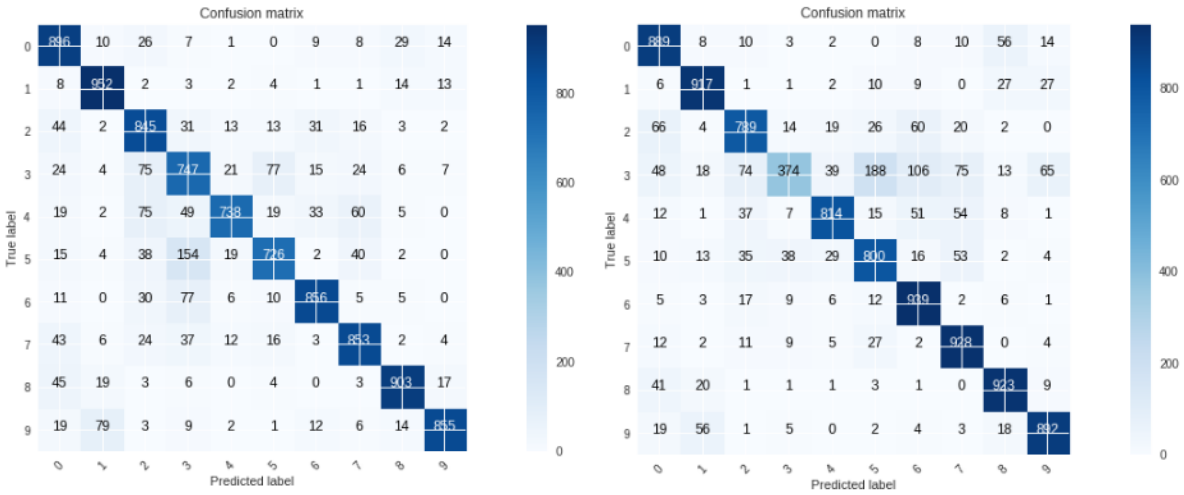


Figura 24: Matriz de confusión de las clases de CIFAR-10 para la arquitectura maestra con descomposición en cadena.

Para el benchmark de OAO con mayoría de votos, solamente se lograron entrenar los sistemas con Fine Tunning y con Fine Tunnig con capas congeladas. Los resultados son:

Clase	Accuracy % Training Desde 0	Accuracy % Fine Tunning	Accuracy % Fine Tunning Capas Congeladas
Airplane	-	89.46	88.47
Automobile	-	90.24	91.23
Bird	-	84.56	78.52
Cat	-	74.77	37.68
Deer	-	73.64	81.18
Dog	-	72.12	78.67
Frog	-	85.23	93.28
Horse	-	85.89	88.46
Ship	-	90.62	92.78
Truck	-	85.58	89.95
prom	-	83.21	82.02

Tabla 31: Benchmark CIFAR-10 para la arquitectura maestra con descomposición OAO con mayoría de votos.



(a) Solo Fine Tunning.

(b) Fine Tunning con capas congeladas.

Figura 25: Matriz de confusión de las clases de CIFAR-10 para la arquitectura maestra con des-composición OAO con mayoría de votos.

4.2.9. Arquitectura Básica

La arquitectura a utilizar se conoce como CNN+Relu, propuesta en [20]:

Name	Neuron	Filter Size	Activation	Output SIze
Conv2D	96	5x5	Relu	32x32x96
Dropout	-	-	-	32x32x96
MaxPooling2D	-	3x3	-	10x10x96
Dropout	-	-	-	10x10x96
Conv2D	128	5x5	Relu	10x10x128
Dropout	-	-	-	10x10x128
AveragePooling2D	-	3x3	-	3x3x128
Dropout	-	-	-	3x3x128
Conv2D	256	-	Relu	3x3x256
Dropout	-	-	-	3x3x256
AveragePooling2D	-	-	-	1x1x256
Dropout	-	-	-	1x1x256
Dense	2048	-	Relu	1x1x2048
Dropout	-	-	-	1x1x2048
Dense	2048	-	Relu	1x1x2048
Dropout	-	-	-	1x1x2048
Flatten	-	-	-	2048
Dense	10	-	Softmax	10

Tabla 32: CIFAR-10 arquitectura basica, red CNN+Relu.

Se utiliza el mismo pre-procesamiento de [6] en el se calcula el valor promedio de cada pixel, considerando el conjunto de entrenamiento completo y cada canal por separado. Luego, dichos valores son restados de cada imagen. Para toda capa oculta, se utiliza ReLU como función de activación. Se aplica dropout de $0,25$ previo a cada capa de pooling. Además, se aplica dropout de $0,25$, $0,25$ y $0,5$ después de cada capa de pooling, respectivamente. El modelo es entrenado durante 150 epochs por medio de SGD con tamaño de mini-batch 100. La tasa de aprendizaje cambia en epochs específicos: Es de $0,01$ desde el epoch 1, es de 0,001 desde el epoch 100, es de $0,0001$ desde el epoch 134 y es de $0,00001$ desde el epoch 142. Se aplica weight decay de $0,001$. Además, se utiliza un momentum inicial de $0,5$ que aumenta hasta un valor máximo de $0,9$ de acuerdo a la siguiente ecuación, que corresponde a una versión modificada de la original:

$$m_i = m_{max} - \left(m_{max} - m_0\right) \cdot exp(-0,06i)\,, \tag{2}$$

donde m_i es el valor del momentum en el epoch i, m_{max} es su valor máximo y m_0 es su valor inicial.

El accuracy reportado por los autores es del $87{,}44$ %, mientras que el accuracy obtenido experimentalmente fue del $84{,}41$ % por lo cual, el accuracy por clase fue de:

Class	Accuracy %
Airplane	84.58
Automobile	93.80
Bird	76.16
Cat	70.57
Deer	71.82
Dog	82.09
Frog	86.22
Horse	90.62
Ship	92.04
Truck	92.23
prom	84.01

Tabla 33: Accuracy por clase para el dataset CIFAR10 usando la arquitectura básica sin descomposición.

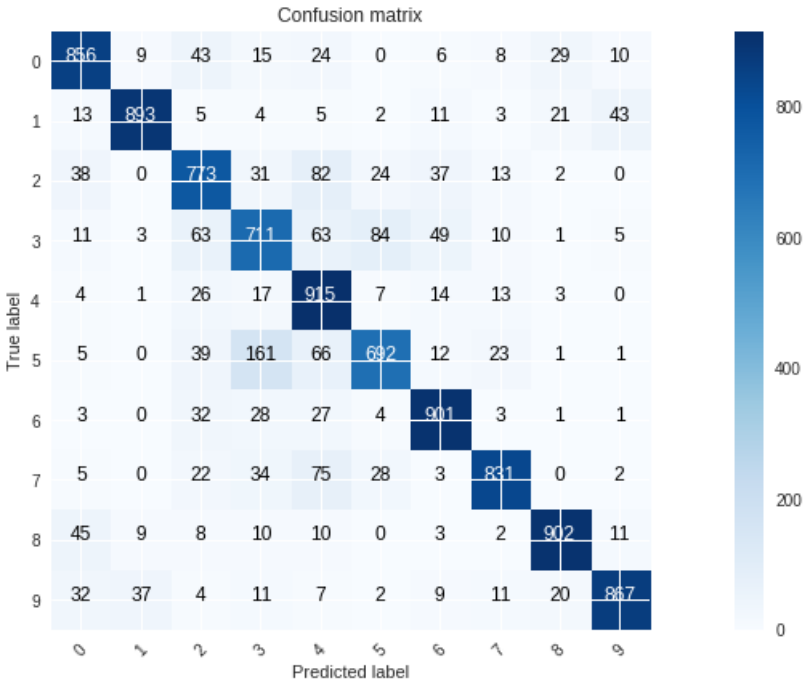


Figura 26: Matriz de confusión de las clases de CIFAR10 para la arquitectura básica sin descomposición.

Para la descomposición en cadena, se obtuvo los siguientes resultados:

Class	Accuracy %
Airplane	70.3
Automobile	65.3
Bird	67.6
Cat	43.5
Deer	77.4
Dog	51.8
Frog	82.1
Horse	74.9
Ship	77.9
Truck	71.4
prom	68.22

Tabla 34: Benchmark CIFAR-10 para la arquitectura básica con descomposición en cadena.

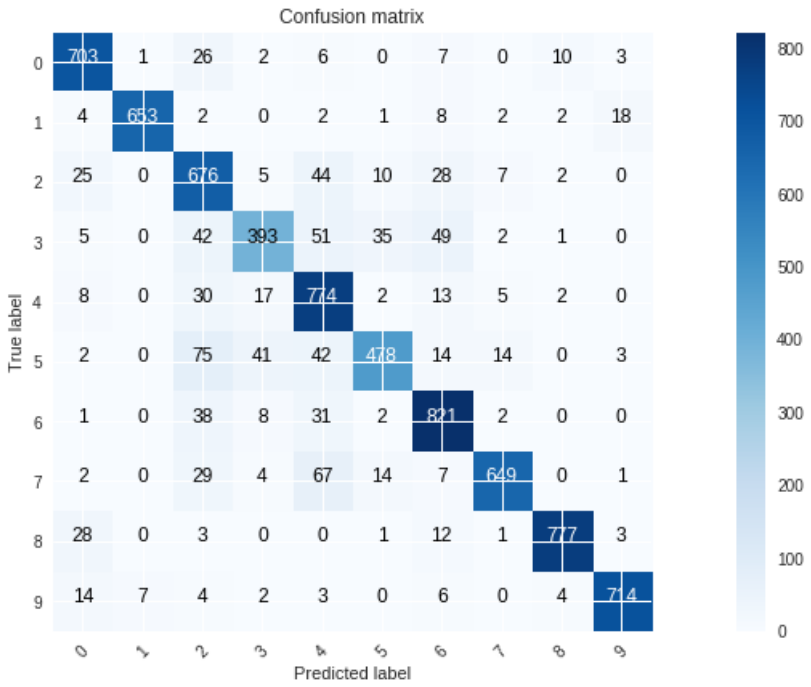
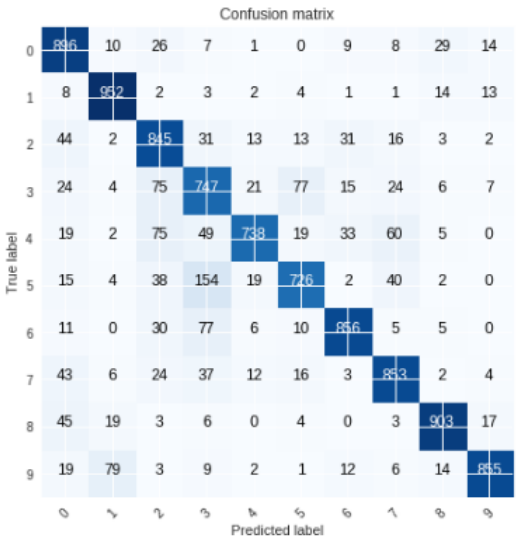


Figura 27: Matriz de confusión de las clases de CIFAR-10 para la arquitectura básica con descomposición en cadena.

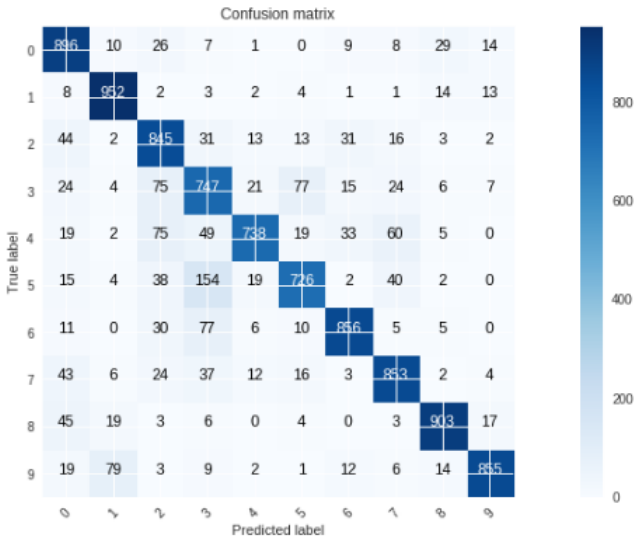
Finalmente, usando la descomposición OAO con mayoría de votos, se obtuvo lo siguiente:

Clase	Accuracy % Training Desde 0	Accuracy % Fine Tunning	Accuracy % Fine Tunning Capas Congeladas
Airplane	85.85	79.83	88.92
Automobile	87.93	88.46	91.63
Bird	84.71	75.13	78.73
Cat	71.51	66.62	37.21
Deer	71.48	71.25	81.48
Dog	72.54	80.49	80.93
Frog	83.38	79.74	80.91
Horse	85.74	83.91	92.96
Ship	87.59	91.68	92.87
Truck	85.96	93.62	89.04
prom	81.67	81.07	81.47

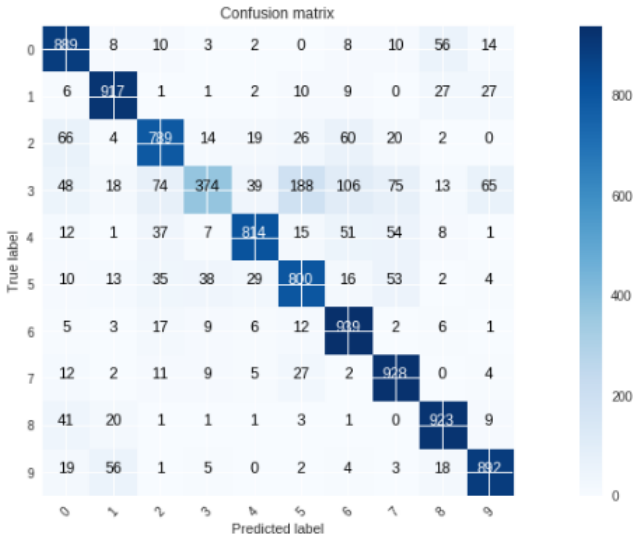
Tabla 35: Benchmark CIFAR-10 para la arquitectura básica con descomposición OAO con mayoría de votos.



(a) Entrenamiento desde 0.



(b) Solo Fine Tunning.



(c) Fine Tunning con capas congeladas.

Figura 28: Matriz de confusión de las clases de CIFAR-10 para la arquitectura básica con descomposición OAO con mayoría de votos.

4.3. Unbalanced CIFAR-10

Para estudiar como se comporta nuestra arquitectura GAD frente a dataset desbalanceado, se escogió el dataset CIFAR-10 ya que aún no se tiene un accuracy elevado por clase sobre el 95 % lo que lo hace un candidato estupendo para verificar si esta arquitectura puede aumentar el accuracy sobre las clases menos representativas. Para generar el desbalance en el

dataset, se observaron los resultados del accuracy por clase tanto de la arquitectura maestra como de la básica:

Class	Accuracy % Master	Accuracy % Basic
Airplane	88.01	84.58
Automobile	92.59	93.80
Bird	80.53	76.16
Cat	70.83	70.57
Deer	85.94	71.82
Dog	82.44	82.09
Frog	87.45	86.22
Horse	92.58	90.62
Ship	89.19	92.04
Truck	92.52	92.23
prom	86.21	84.01

Tabla 36: Accuracy por clase de la arquitectura maestra y básica de CIFAR-10 sin descomposición.

Se decidió que si una clase tiene un porcentaje de accuracy inferior al 90 % en alguno de los modelos master o basic, se reducirá el total de datos de entrenamiento y testing a un 10 % de la cantidad original, mientras que si en ambos casos, el accuracy es superior al 90 %, los datos de entrenamiento se mantienen.

Esto llega a una reducción en las clases Airplane, Bird, Cat, Deer, Dog, Frog y Ship quedando en un total de 499 datos de entrenamiento y 99 de prueba por clase, mientras que las clases Automobile, Horse y Truck quedan con 5000 datos de entrenamiento y 1000 de datos de test por clases.

Class	Train	Test
Airplane	499	99
Automobile	5000	1000
Bird	499	99
Cat	499	99
Deer	499	99
Dog	499	99
Frog	499	99
Horse	5000	1000
Ship	499	99
Truck	5000	1000

Tabla 37: Datos de entrenamiento y Prueba por clase para el dataset CIFAR10 desbalanceado.

4.3.1. Benchmark Arq. Maestra

Los resultados del accuracy experimental de la arquitectura maestra son los siguientes:

Class	Accuracy %
Airplane	36.38
Automobile	87.93
Bird	32.52
Cat	26.29
Deer	53.67
Dog	24.21
Frog	60.62
Horse	87.20
Ship	67.54
Truck	89.30
prom	56.57

Tabla 38: Accuracy por clase para el dataset Unbalanced CIFAR-10 usando la arquitectura maestra sin descomposición.

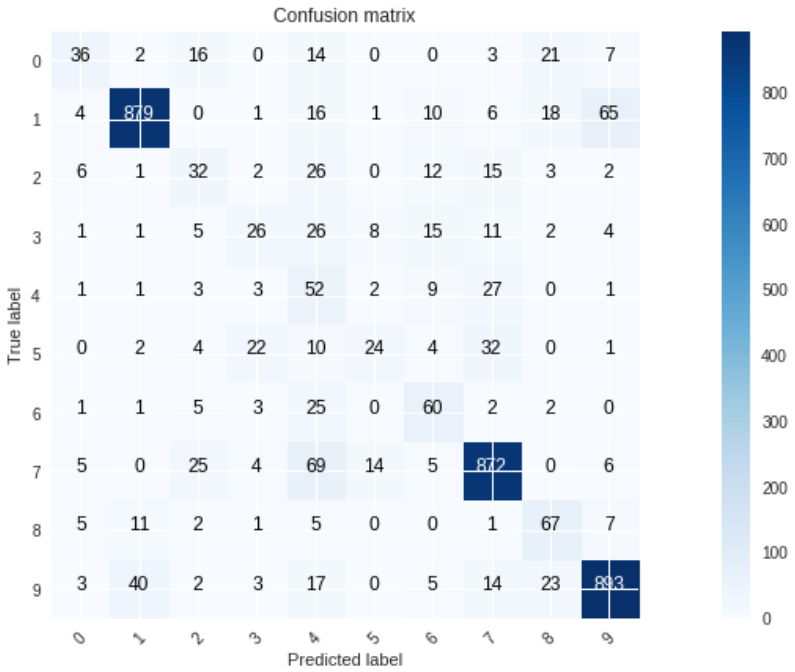


Figura 29: Matriz de confusión de las clases de Unbalanced CIFAR-10 para la arquitectura maestra sin descomposición.

Al descomponer la arquitectura maestra en una estructura encadenada, tenemos que el ac-

curacy por clase es:

Class	Accuracy %
Airplane	45.45
Automobile	87.10
Bird	44.44
Cat	26.28
Deer	65.72
Dog	33.65
Frog	69.72
Horse	85.80
Ship	74.08
Truck	85.04
prom	61.73

Tabla 39: Accuracy por clase para el dataset Unbalanced CIFAR-10 usando la arquitectura maestra con descomposición en cadena.

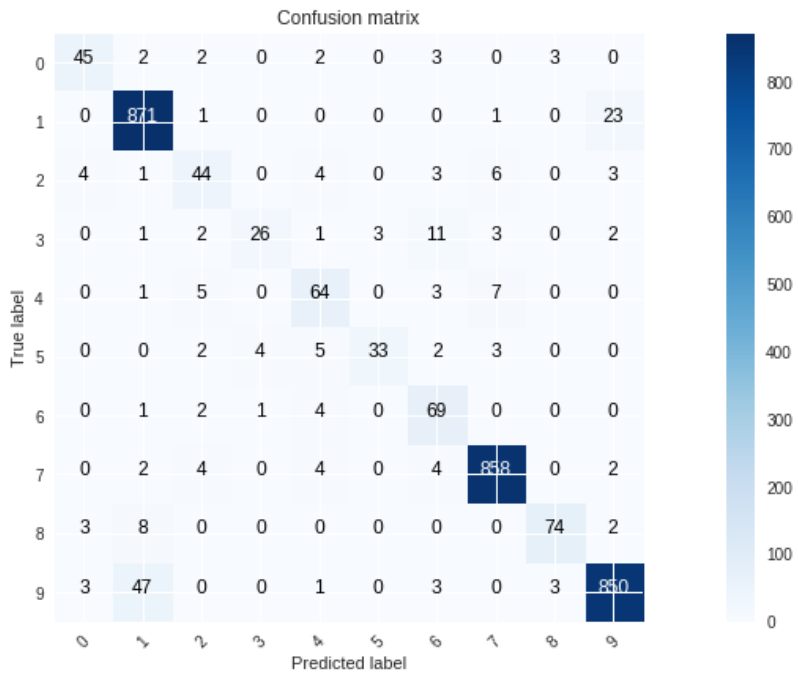
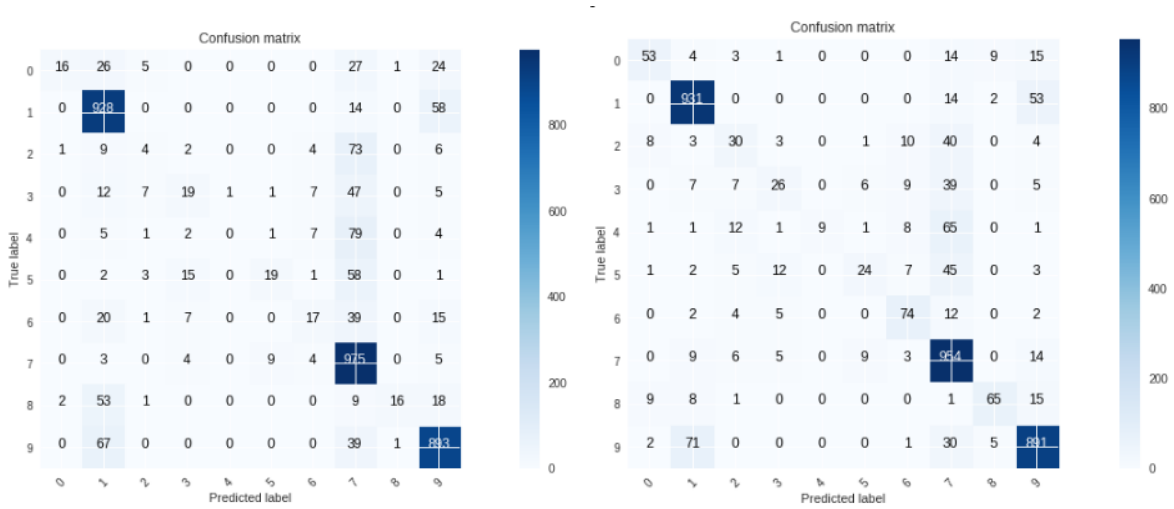


Figura 30: Matriz de confusión de las clases de Unbalanced CIFAR-10 para la arquitectura maestra con descomposición en cadena.

Por temas de experimentación que veremos más adelante, no se realizará descomposición OAO con mayoría de votos a la arquitectura maestra entrenando desde 0. Dicho esto, los resultados de las otras dos formas de experimentación son:

Clase	Accuracy % Training Desde 0	Accuracy % Fine Tunning	Accuracy % Fine Tunning Capas Congeladas
Airplane	-	16.83	53.47
Automobile	-	92.35	93.95
Bird	-	4.55	30.24
Cat	-	19.36	16.78
Deer	-	0.97	9.25
Dog	-	19.41	24.26
Frog	-	17.15	74.58
Horse	-	97.41	95.74
Ship	-	16.18	48.31
Truck	-	89.78	86.94
prom	-	37.40	53.35

Tabla 40: Benchmark Unbalanced CIFAR-10 para la arquitectura maestra con descomposición OAO con mayoría de votos.



(a) Solo Fine Tunning.

(b) Fine Tunning con capas congeladas.

Figura 31: Matriz de confusión de las clases de Unbalanced CIFAR-10 para la arquitectura maestra con des-composición OAO con mayoría de votos.

4.3.2. Benchmark Arq. Básica

Para la arquitectura básica, los experimentos nos otorgaron los siguientes resultados:

Class	Accuracy %
Airplane	39.41
Automobile	91.32
Bird	25.19
Cat	32.54
Deer	28.19
Dog	39.17
Frog	77.78
Horse	95.51
Ship	62.54
Truck	91.58
prom	58.32

Tabla 41: Accuracy por clase para el dataset Unbalanced CIFAR-10 usando la arquitectura básica sin descomposición.

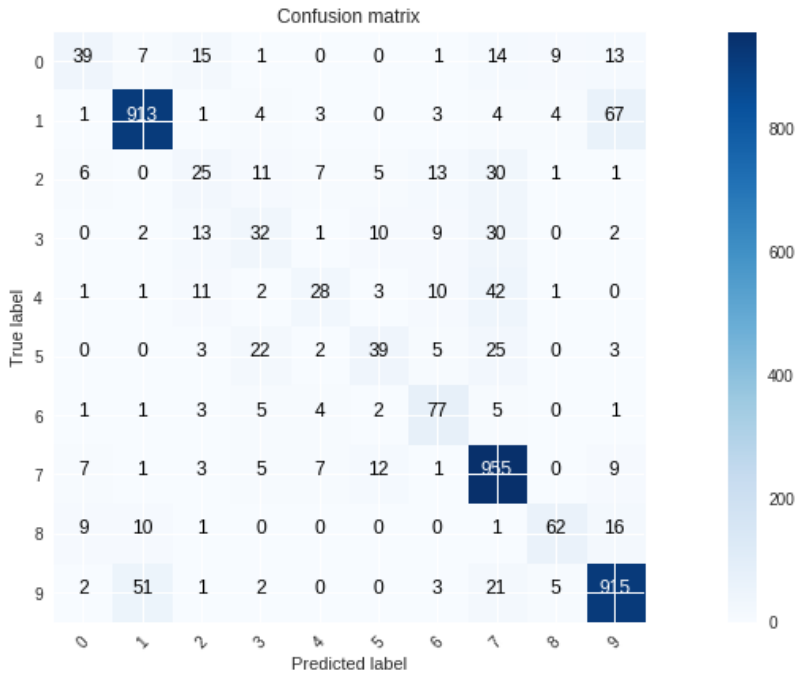


Figura 32: Matriz de confusión de las clases de Unbalanced CIFAR-10 para la arquitectura básica sin descomposición.

Al descomponer la arquitectura básica en una estructura encadenada, tenemos que el accuracy por clase es:

Class	Accuracy %
Airplane	51.49
Automobile	89.27
Bird	41.35
Cat	23.31
Deer	43.51
Dog	25.36
Frog	63.59
Horse	88.53
Ship	64.67
Truck	84.89
prom	57.60

Tabla 42: Accuracy por clase para el dataset Unbalanced CIFAR-10 usando la arquitectura básica con descomposición en cadena.

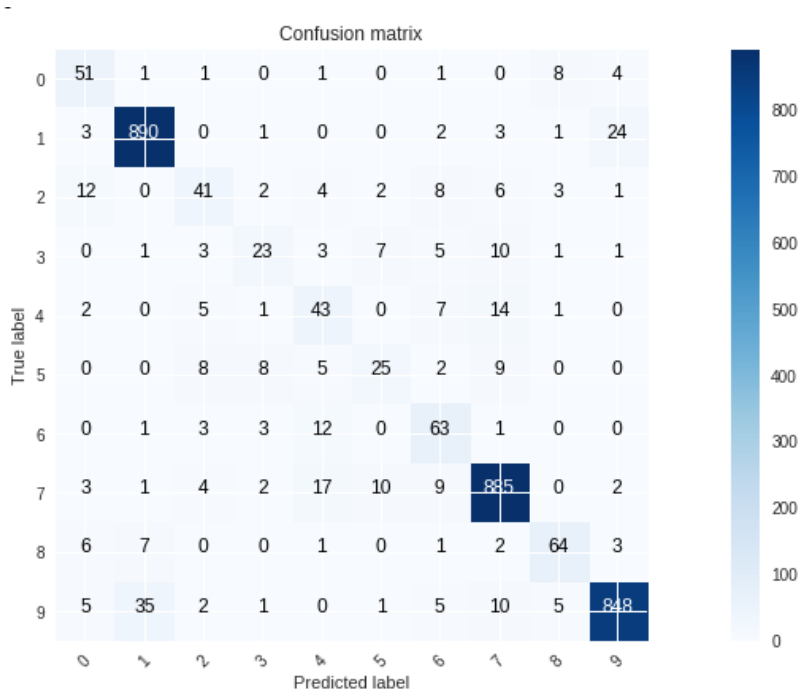
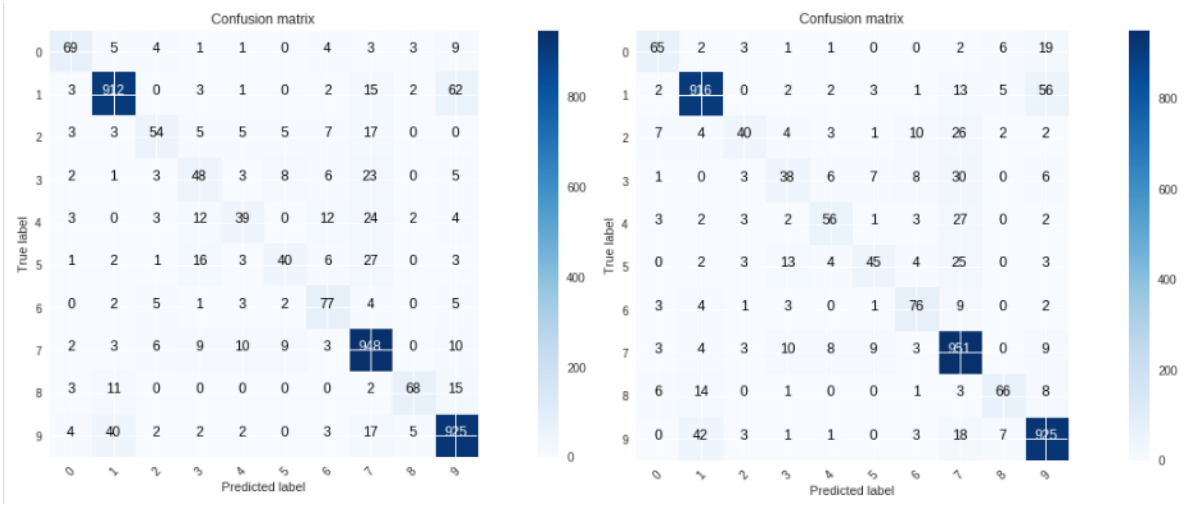


Figura 33: Matriz de confusión de las clases de Unbalanced CIFAR-10 para la arquitectura básica con descomposición en cadena.

Al igual que la arquitectura maestra, no se realizará entrenamiento desde 0 para la arquitectura básica. Dicho esto, los resultados de la descomposición son:

Clase	Accuracy % Training Desde 0	Accuracy % Fine Tunning	Accuracy % Fine Tunning Capas Congeladas
Airplane	-	69.77	49.63
Automobile	-	81.24	91.68
Bird	-	44.54	38.42
Cat	-	30.49	35.34
Deer	-	39.31	42.57
Dog	-	40.48	45.45
Frog	-	77.76	76.77
Horse	-	94.82	95.11
Ship	-	68.69	66.67
Truck	-	92.53	90.59
prom	-	63.96	63.22

Tabla 43: Benchmark Unbalanced CIFAR-10 para la arquitectura básica con descomposición OAO con mayoría de votos.



(a) Solo Fine Tunning.

(b) Fine Tunning con capas congeladas.

Figura 34: Matriz de confusión de las clases de Unbalanced CIFAR-10 para la arquitectura básica con des-composición OAO con mayoría de votos.

4.4. BAAT

BAAT es un problema que se soluciona de manera simple con un modelo pre-entrenado como ResNet50, pero para mantener la idea principal de este trabajo, se creó un modelo propio siguiendo la filosofía de combinar capas Convolucional con capas de MaxPooling, resultado

en la siguiente red:

Name	Neuron	Filter Size	Activation	Output Size
Conv2D	96	5x5	Relu	84x84x96
Dropout	-	-	-	84x84x96
MaxPooling2D	-	3x3	-	28x28x96
Dropout	-	-	-	28x28x96
Conv2D	128	5x5	Relu	28x28x128
Dropout	-	-	-	28x28x128
AveragePooling2D	-	3x3	-	9x9x128
Dropout	-	-	-	9x9x128
Conv2D	256	-	Relu	9x9x256
Dropout	-	-	-	9x9x256
AveragePooling2D	-	-	-	3x3x256
Dropout	-	-	-	3x3x256
Dense	2048	-	Relu	3x3x2048
Dropout	-	-	-	3x3x2048
Dense	2048	-	Relu	3x3x2048
Dropout	-	-	-	3x3x2048
Flatten	-	-	-	18432
Dense	10	-	Softmax	10

Tabla 44: Arquitectura Convolutional-MaxPooling para el dataset BAAT

Respecto a la red, el porcentaje de dropout está fijado en $0,25\,\%$ para los primeros cuatro dropouts y posteriormente se aumenta a $0,5\,\%$. Se realiza una regularización l2 de $0,001$ en las capas convolucionales y densas. Respecto a los datos, estos fueron pre-procesados para que solo se trabajara con 256x256x3 dimensiones por temas de memoria de la GPU y además solamente se trabajó con los 10 artistas que tienen más datos de ejemplo.

Ya que no se tiene una división nativa del conjunto de entrenamiento o de prueba, se tomó (aleatoriamente) un $20\,\%$ de los datos para prueba, resultando en la siguiente distribución:

Class	Train	Test
Vincent van Gogh	702	175
Edgar Degas	562	140
Pablo Picasso	352	87
Pierre-Auguste Renoir	269	67
Albrecht Durer	263	65
Paul Gauguin	249	62
Francisco Goya	233	58
Rembrandt	210	52
Alfred Sisley	208	51
Titian	204	51
Total	3252	808

Tabla 45: Distribución de datos de entrenamiento y prueba después de la división de datos en BAAT.

El entrenamiento fue realizado en dos partes. Primero se entrenó por 50 epochs con el optimizador Adam, usando la configuración base de keras. Posteriormente, se entrenó por otros 50 epochs con SGD. Para el entrenamiento con SGD, se inició con un learning rate de $0,1$ y se fue multiplicando el larning rate por $0,1$ cada vez que no se detectara cambios en el conjunto de validación cada 5 epochs. El resultado de este entrenamiento fue que el accuracy total alcanzado fue de 85.03 % pero al observar el accuracy por clase, tenemos un resultado menos alentador:

Class	Accuracy %
Vincent van Gogh	07.43
Edgar Degas	85.71
Pablo Picasso	35.63
Pierre-Auguste Renoir	00.00
Albrecht Durer	30.77
Paul Gauguin	00.00
Francisco Goya	18.97
Rembrandt	50.63
Alfred Sisley	68.63
Titian	64.71
prom	36.25

Tabla 46: Accuracy por clase para el dataset BAAT sin descomposición.

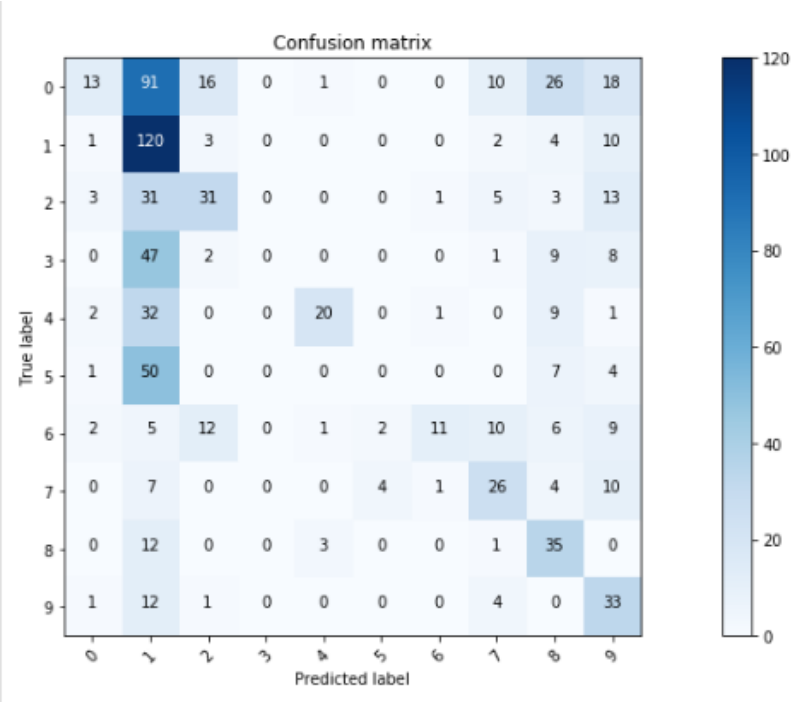


Figura 35: Matriz de confusión de las clases de BAAT para su arquitectura base sin descomposición.

La descomposición en cadena, usando la misma configuración y el mismo tipo de entrenamiento, dió el siguiente resultado:

Class	Accuracy %
Vincent van Gogh	12.53
Edgar Degas	83.24
Pablo Picasso	33.46
Pierre-Auguste Renoir	15.64
Albrecht Durer	31.46
Paul Gauguin	10.95
Francisco Goya	20.63
Rembrandt	49.23
Alfred Sisley	67.75
Titian	65.38
prom	39.03

Tabla 47: Accuracy por clase para el dataset BAAT con descomposición en cadena.

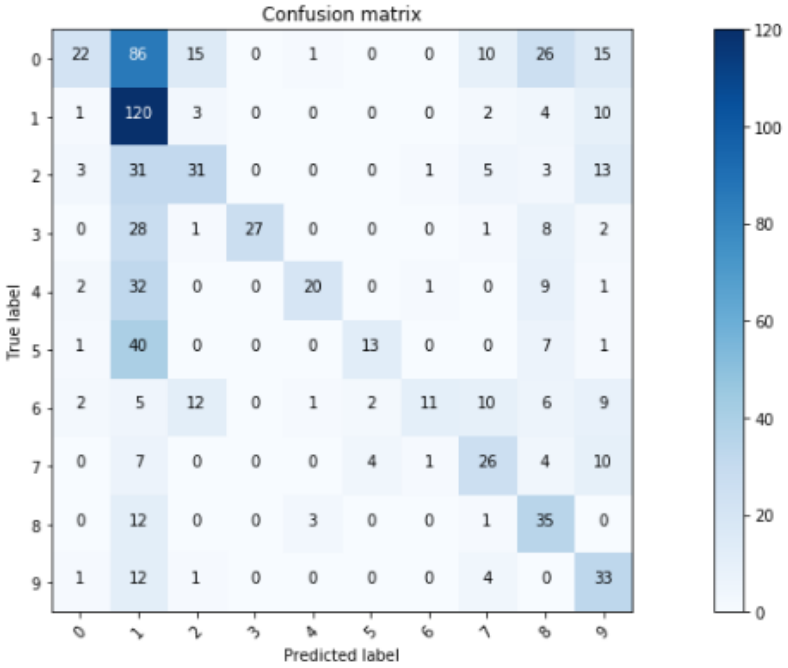
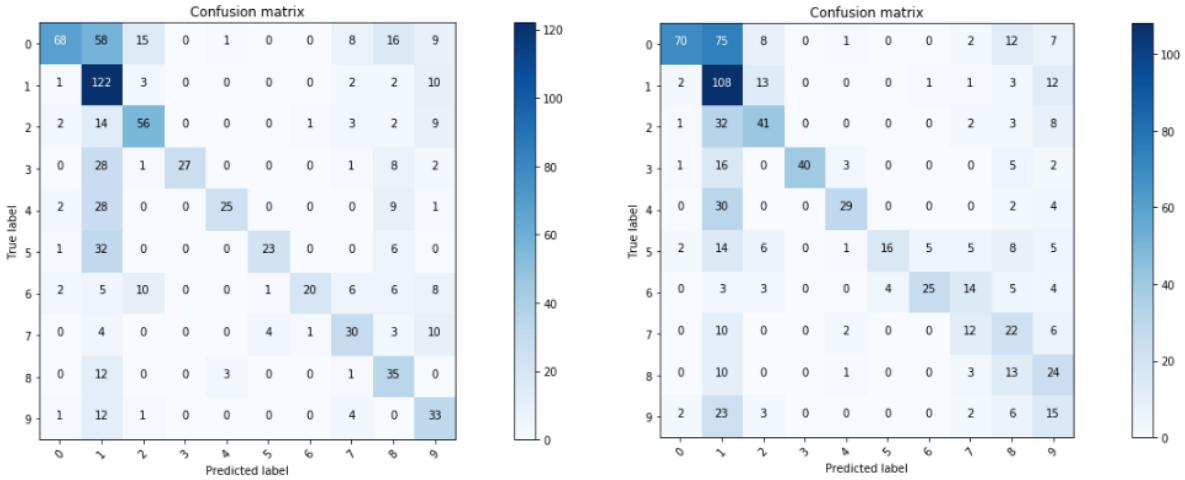


Figura 36: Matriz de confusión de las clases de BAAT para su arquitectura base con descomposición en cadena.

Para el benchmark de OAO con mayoría de votos, al igual que cifar-10, solamente se logró entrenar los sistemas con Fine Tunning y con Fine Tunning con capas congeladas:

Class	Accuracy % Training Desde 0	Accuracy % Fine Tunning	Accuracy % Fine Tunning Capas Congeladas
Vincent van Gogh	-	20.86	25.47
Edgar Degas	-	85.14	85.58
Pablo Picasso	-	38.37	40.19
Pierre-Auguste Renoir	-	20.30	23.79
Albrecht Durer	-	35.46	38.97
Paul Gauguin	-	17.10	19.68
Francisco Goya	-	24.48	26.68
Rembrandt	-	53.46	54.13
Alfred Sisley	-	68.63	69.44
Titian	-	65.71	66.18
prom	-	42.95	45.01

Tabla 48: Benchmark BAAT para la arquitectura base con descomposición OAO con mayoría de votos.



(a) Solo Fine Tunning.

(b) Fine Tunning con capas congeladas.

Figura 37: Matriz de confusión de las clases de BAAT para la arquitectura base con descomposición OAO con mayoría de votos.

4.5. Análisis Experimental de la Propuesta

Los siguientes experimentos nos permitirán corroborar la hipótesis de que dividir un problema de múltiples clases en múltiples redes es mejor que simplemente tener una sola red, por lo cual el objetivo de estos experimentos es tomar la arquitectura maestra y la estructura básica y transformarlas en una arquitectura GAD para así analizar el comportamiento del accuracy por clase. Además, para observar el comportamiento de la arquitectura en dataset desbalanceados, se tomará CIFAR-10 y se desbalanceará la cantidad de ejemplos por clase (como en los experimentos anteriores) y a su vez se experimentará con el dataset BAAT pero para este caso, solo se usará una única arquitectura base. Los experimentos son:

- **Experimento 1**: Tomar la arquitectura maestra/básica sin entrenar, con N neuronas de salida. Las redes correspondientes a cada nodo del grafo se generan luego cambiando la última capa por una capa con 1 neuronas para clasificar sus respectivas clases objetivo. Todas las capas de esta red se declaran como entrenables.
- **Experimento 2**: Tomar la arquitectura maestra/básica ya pre-entrenada con N neuronas de salida. Las redes correspondientes a cada nodo, se generan como en el caso anterior. Todas las capas de esta red se declaran como entrenables.
- **Experimento 3**: Tomar la arquitectura maestra/básica ya pre-entrenada con N neuronas de salida. Las redes correspondientes a cada nodo, se generan como en el caso anterior, pero se declara solo la última capa como entrenable.

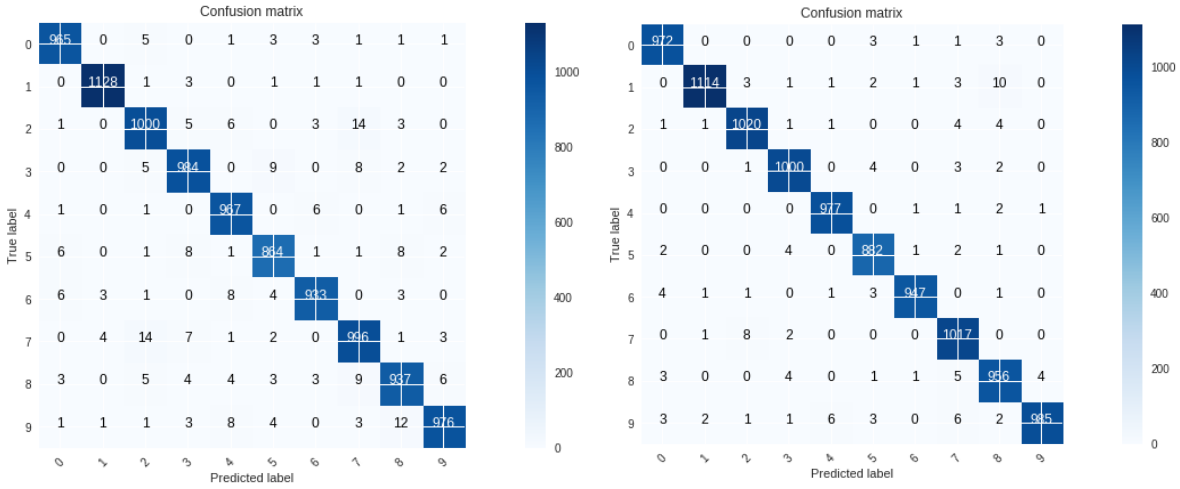
4.5.1. MNIST

En el primer experimento se entrena por completo un modelo con dos neuronas de salida, por lo cual la inicialización de pesos tanto para la arquitectura GAD Maestra como básica están en un rango de $[-0,05, 0,05]$. Ambas arquitecturas fueron entrenadas con 110 epochs, ya que por experimentación, una mayor cantidad de epochs no conlleva a un aumento significativo del accuracy. En ambas arquitecturas se usó un mini-batch de 128 para compatibilizar la forma en que la tarjeta gráfica maneja la memoria interna y para tener un pool de data eficiente en cada epoch.

Los resultados obtenidos en este experimento se muestran en la siguiente tabla:

Class	Benchmark Master	Benchmark Master Chain Decomposition	Bechmark Master OAO	Accuracy % Exp 1 Master	% de mejora respecto a Arq. Maestra Original	% de mejora respecto a Arq. Master Chain	% de mejora respecto a Arq. Master OAO
0	98.68	98.37	98.38	98.42	-0.26	0.05	0.04
1	99.12	99.03	99.02	99.30	0.18	0.27	0.28
2	98.50	96.22	96.85	97.71	-0.79	1.49	0.86
3	98.23	97.03	96.48	97.04	-1.19	0.01	0.56
4	98.67	97.97	97.29	97.99	-0.68	0.02	0.70
5	99.21	96.07	96.41	97.08	-2.13	1.01	0.67
6	99.16	96.86	97.27	98.21	-0.95	1.35	0.94
7	98.85	95.52	96.64	97.42	-1.43	1.90	0.78
8	99.14	95.58	96.16	97.80	-1.34	2.22	1.64
9	99.10	94.35	96.63	97.99	-1.11	3.64	1.36
Prom	98.87	96.70	97.11	97.90	-0.97	1.20	0.78
Class	Benchmark Basic	Benchmark Basic Chain Decomposition	Bechmark Basic OAO	Accuracy % Exp 1 Basic	% de mejora respecto a Arq. Basic Original	% de mejora respecto a Arq. Basic Chain	% de mejora respecto a Arq. Basic OAO
0	97.02	97.35	98.37	98.28	1.26	0.93	-0.09
1	97.92	99.38	99.57	99.58	1.66	0.20	0.01
2	99.41	97.87	92.23	98.63	-0.78	0.76	6.40
3	98.05	98.51	97.56	98.92	0.87	0.41	1.36
4	98.98	98.48	98.84	99.63	0.65	1.15	0.79
5	98.76	98.32	88.37	98.95	0.19	0.63	10.58
6	99.89	97.60	98.56	99.19	-0.70	1.59	0.63
7	97.33	97.18	99.64	98.28	0.95	1.10	-1.36
8	99.58	98.05	98.79	98.08	-1.50	0.03	-0.71
9	98.88	95.24	98.27	99.47	0.59	4.23	1.20
prom	98.58	97.80	97.02	98.90	0.32	1.10	1.88

Tabla 49: Resultados experimento 1 sobre el dataset MNIST.



(a) Arq. maestra GAD.

(b) Arq. básica GAD.

Figura 38: Matriz de confusión de los resultados del experimento 1 sobre el conjunto de prueba de MNIST.

En el segundo experimento, se realizó el mismo pre-procesamiento a las imágenes. Ya que contamos con un modelo pre-entrenado, cada nodo o red convolucional se entrenó solamente con 20 epochs. Esto último se debe a que realizando distintos experimentos se encontró que luego de los 20 epochs tanto el accuracy como la función de pérdida convergen.

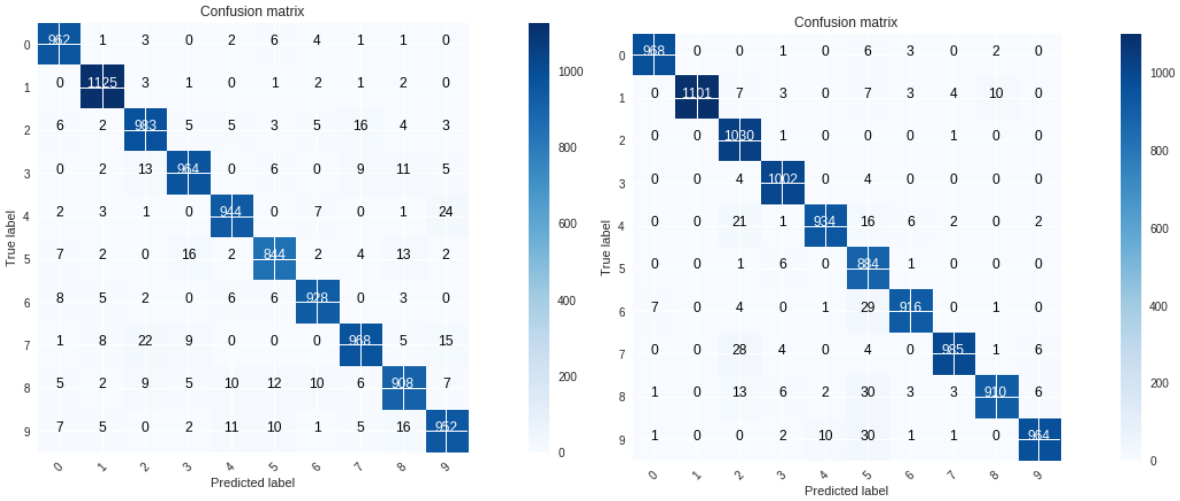
Los resultados obtenidos son:

Class	Benchmark Master	Benchmark Master Chain Decomposition	Bechmark Master OAO	Accuracy % Exp 2 Master	% de mejora respecto a Arq. Maestra Original	% de mejora respecto a Arq. Master Chain	% de mejora respecto a Arq. Master OAO
0	98.68	98.37	98.36	99.04	0.36	0.67	0.68
1	99.12	99.03	98.36	98.27	-0.85	-0.76	-0.09
2	98.50	96.22	96.68	96.95	-1.55	0.73	0.27
3	98.23	97.03	96.38	97.62	-0.61	0.59	1.24
4	98.67	97.97	97.13	97.98	-0.69	0.01	0.85
5	99.21	96.07	96.48	96.13	-3.08	0.06	-0.35
6	99.16	96.86	96.38	97.30	-1.86	0.44	0.92
7	98.85	95.52	96.07	96.23	-2.62	0.71	0.16
8	99.14	95.58	97.47	99.99	0.85	4.41	2.52
9	99.10	94.35	97.86	98.69	-0.41	4.34	0.83
prom	98.87	96.70	97.12	97.82	-1.05	1.12	0.70
Class	Benchmark Basic	Benchmark Basic Chain Decomposition	Bechmark Basic OAO	Accuracy % Exp 2 Basic	% de mejora respecto a Arq. Basic Original	% de mejora respecto a Arq. Basic Chain	% de mejora respecto a Arq. Basic OAO
0	97.02	97.35	98.38	99.08	2.06	1.73	0.70
1	97.92	99.38	97.57	99.99	2.07	0.61	2.42
2	99.41	97.87	99.18	97.96	-1.45	0.09	-1.22
3	98.05	98.51	99.26	97.66	-0.39	-0.85	-1.60
4	98.98	98.48	95.39	98.63	-0.35	0.15	3.24
5	98.76	98.32	99.74	97.52	-1.24	-0.80	-2.22
6	99.89	97.60	95.57	98.18	-1.71	0.58	2.61
7	97.33	97.18	95.24	98.90	1.57	1.72	3.66
8	99.58	98.05	93.36	98.48	-1.10	0.43	5.12
9	98.88	95.24	95.15	98.57	-0.31	3.33	3.42
prom	98.58	97.80	96.88	98.50	-0.09	0.70	1.61

Tabla 50: Resultados experimento 2 sobre el dataset MNIST.

Class	Benchmark Master	Benchmark Master Chain Decomposition	Bechmark Master OAO	Accuracy % Exp 3 Master	% de mejora respecto a Arq. Maestra Original	% de mejora respecto a Arq. Master Chain	% de mejora respecto a Arq. Master OAO
0	98.68	98.37	97.48	98.35	-0.33	-0.02	0.87
1	99.12	99.03	98.72	98.86	-0.26	-0.17	0.14
2	98.50	96.22	96.81	97.12	-1.38	0.90	0.31
3	98.23	97.03	97.87	97.62	-0.61	0.59	-0.25
4	98.67	97.97	97.89	97.87	-0.80	-0.10	-0.02
5	99.21	96.07	96.44	95.74	-3.47	-0.33	-0.70
6	99.16	96.86	97.68	97.28	-1.88	0.42	-0.40
7	98.85	95.52	97.53	96.92	-1.93	1.40	-0.61
8	99.14	95.58	98.15	99.99	0.85	4.41	1.84
9	99.10	94.35	98.19	98.68	-0.42	4.33	0.49
prom	98.87	96.70	98.19	97.84	-1.02	1.14	0.17
Class	Benchmark Basic	Benchmark Basic Chain Decomposition	Bechmark Basic OAO	Accuracy % Exp 3 Basic	% de mejora respecto a Arq. Basic Original	% de mejora respecto a Arq. Basic Chain	% de mejora respecto a Arq. Basic OAO
0	97.02	97.35	98.75	99.08	2.06	1.73	0.33
1	97.92	99.38	98.35	99.99	2.07	0.61	1.64
2	99.41	97.87	99.82	98.55	-0.86	0.68	-1.27
3	98.05	98.51	99.38	98.66	0.61	0.15	-0.72
4	98.98	98.48	95.54	99.04	0.06	0.56	3.50
5	98.76	98.32	98.18	96.48	-2.28	-1.84	-1.70
6	99.89	97.60	96.91	98.19	-1.70	0.59	1.28
7	97.33	97.18	95.17	98.99	1.66	1.81	3.82
8	99.58	98.05	92.46	98.25	-1.33	0.20	5.79
9	98.88	95.24	95.64	98.06	-0.82	2.82	2.42
prom	98.58	97.80	97.02	98.53	-0.05	0.73	1.51

Tabla 51: Resultados experimento 3 sobre el dataset MNIST.

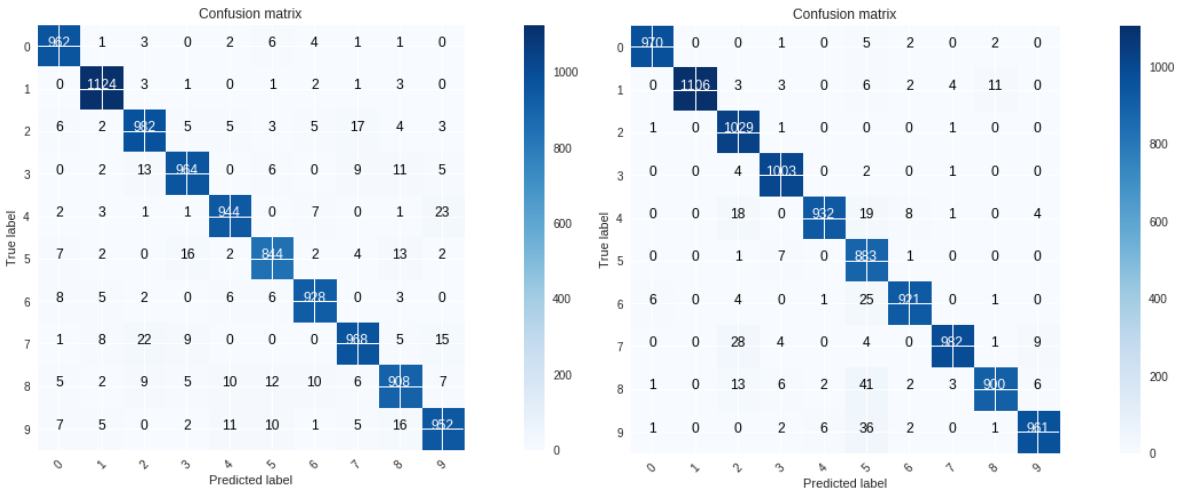


(a) Arq. maestra GAD.

(b) Arq. básica GAD.

Figura 39: Matriz de confusión de los resultados del experimento 2 sobre el conjunto de prueba de MNIST.

Para nuestro último experimento con MNIST, se volvió a entrenar con 20 epochs cada red. Los resultados experimentales son:



(a) Arq. maestra GAD.

(b) Arq. básica GAD.

Figura 40: Matriz de confusión de los resultados del experimento 3 sobre el conjunto de prueba de MNIST.

4.5.2. NORB

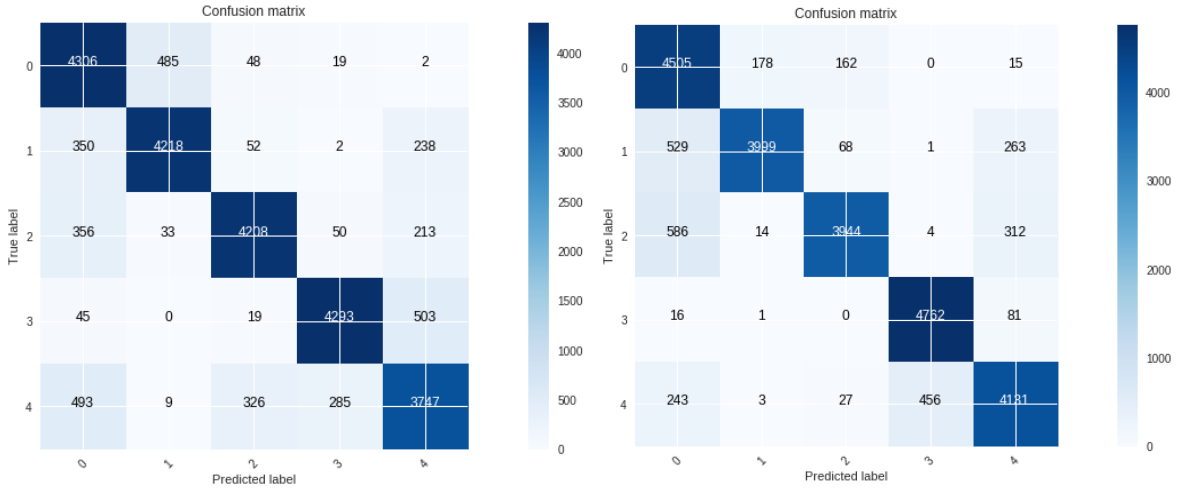
En los siguientes experimentos, se decidió mantener los hyper-parámetros del entrenamiento de la redes igual a sus estructuras maestra y básica, siento estas las siguientes:

- Imágenes escalas a 98x98 píxeles.
- Capa LRN después de las primeras capas de Max pooling/Average pooling.
- Una ventana de pooling de 3x3 para la arquitectura maestra y una de 2x2 para la arquitectura básica.
- 250 epochs.
- Mini-batch de 128.
- Weight decay de $0,01$.
- Momentum de $0,9$.
- Aprendizaje inicial de $0,001$ que es multiplicada por un factor de $0,1$ cada 100 epochs.

Después de entrenar por completo todos las redes con dos neuronas de salida, obtenemos los siguientes resultados:

Class	Benchmark Master	Benchmark Master Chain Decomposition	Bechmark Master OAO	Accuracy % Exp 1 Master	% de mejora respecto a Arq. Maestra Original	% de mejora respecto a Arq. Master Chain	% de mejora respecto a Arq. Master OAO
Animal	81.83	84.18	90.47	77.59	-4.24	-6.59	-12.88
Human	96.20	85.23	86.85	88.89	-7.31	3.66	2.04
Airplane	98.08	76.93	82.13	90.44	-7.64	13.51	8.31
Truck	88.29	91.79	87.68	92.34	4.05	0.55	4.66
Car	84.12	59.88	77.79	79.67	-4.45	19.79	1.88
prom	89.70	79.60	84.98	85.79	-3.92	6.18	0.80
Class	Benchmark Basic	Benchmark Basic Chain Decomposition	Bechmark Basic OAO	Accuracy % Exp 1 Basic	% de mejora respecto a Arq. Basic Original	% de mejora respecto a Arq. Basic Chain	% de mejora respecto a Arq. Basic OAO
Animal	83.03	84.18	93.91	76.63	-6.40	-7.55	-17.28
Human	96.76	77.45	81.32	91.33	-5.43	13.88	10.01
Airplane	96.46	76.85	78.84	93.88	-2.58	17.03	15.04
Truck	86.57	91.56	97.73	91.87	5.30	0.31	-5.86
Car	86.36	57.82	85.13	88.03	1.67	30.21	2.9
prom	89.84	77.57	87.39	88.35	-1.49	10.78	0.96

Tabla 52: Resultados experimento 1 sobre el dataset NORB.



(a) Arq. maestra GAD.

(b) Arq. básica GAD.

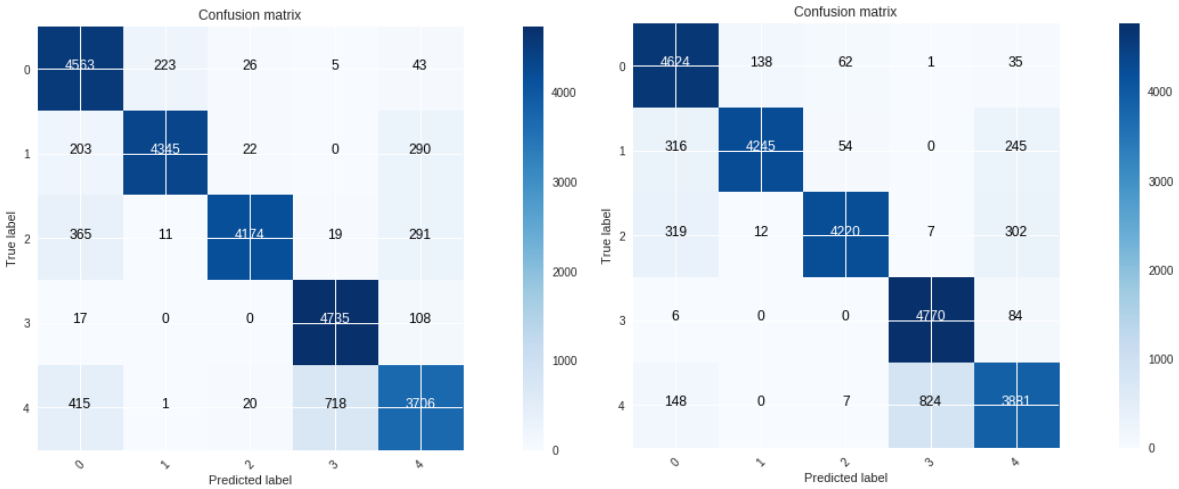
Figura 41: Matriz de confusión de los resultados del experimento 1 sobre el conjunto de prueba de NORB.

Para el segundo y tercer experimento, se mantuvieron los mismos hyper-parámetros al momento de entrenar, excepto por la cantidad de epochs. Esta cantidad fue reducida a 50 en el experimento 2 y a 10 en el experimento 3. Esta decisión fue tomada ya que al experimentar se notó que el accuracy disminuye pasado esta cantidad de epochs para ambos experimentos, debido a que la red se sobre-ajusta demasiado al trabajar con un modelo ya pre-entrenado.

Los resultados del experimento 2 son:

Class	Benchmark Master	Benchmark Master Chain Decomposition	Bechmark Master OAO	Accuracy % Exp 2 Master	% de mejora respecto a Arq. Maestra Original	% de mejora respecto a Arq. Master Chain	% de mejora respecto a Arq. Master OAO
Animal	81.83	84.18	92.58	85.02	3.19	0.84	-7.56
Human	96.20	85.23	89.86	94.87	-1.33	9.64	5.01
Airplane	98.08	76.93	85.14	98.4	0.32	21.47	13.26
Truck	88.29	91.79	97.93	96.45	8.16	4.66	-1.48
Car	84.12	59.88	76.29	83.51	-0.61	23.63	7.22
prom	89.70	79.60	88.36	91.65	1.95	12.05	3.29
Class	Benchmark Basic	Benchmark Basic Chain Decomposition	Bechmark Basic OAO	Accuracy % Exp 2 Basic	% de mejora respecto a Arq. Basic Original	% de mejora respecto a Arq. Basic Chain	% de mejora respecto a Arq. Basic OAO
Animal	83.03	84.18	94.16	85.42	2.39	1.24	-8.74
Human	96.76	77.45	87.37	96.59	-0.17	19.14	9.22
Airplane	96.46	76.85	86.38	97.17	0.71	20.32	10.79
Truck	86.57	91.56	96.32	85.15	-1.42	-6.41	-11.17
Car	86.36	57.82	79.78	85.35	-1.01	27.53	5.57
prom	89.84	77.57	88.80	89.94	0.10	12.36	1.13

Tabla 53: Resultados experimento 2 sobre el dataset NORB.



(a) Arq. maestra GAD.

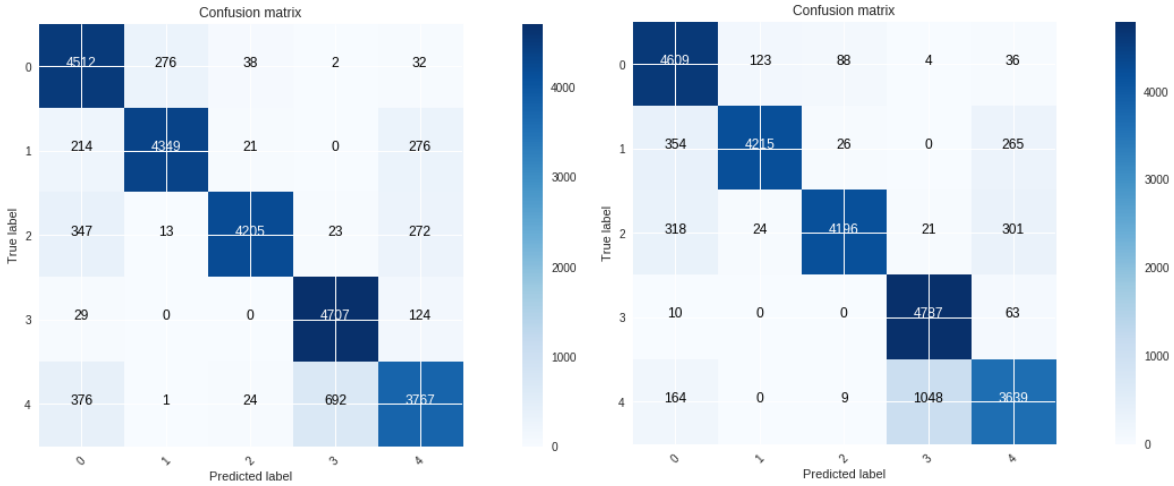
(b) Arq. básica GAD.

Figura 42: Matriz de confusión de los resultados del experimento 2 sobre el conjunto de prueba de NORB.

Finalmente, los resultados del experimento 3 son:

Class	Benchmark Master	Benchmark Master Chain Decomposition	Bechmark Master OAO	Accuracy % Exp 3 Master	% de mejora respecto a Arq. Maestra Original	% de mejora respecto a Arq. Master Chain	% de mejora respecto a Arq. Master OAO
Animal	81.83	84.18	90.79	82.37	0.54	-1.81	-8.42
Human	96.20	85.23	89.51	92.75	-3.45	7.52	3.24
Airplane	98.08	76.93	86.48	95.06	-3.02	18.13	8.58
Truck	88.29	91.79	96.37	89.78	1.49	-2.01	-6.59
Car	84.12	59.88	77.49	85.25	1.13	25.37	7.76
prom	89.70	79.60	88.13	89.04	-0.66	9.44	0.91
Class	Benchmark Basic	Benchmark Basic Chain Decomposition	Bechmark Basic OAO	Accuracy % Exp 3 Basic	% de mejora respecto a Arq. Basic Original	% de mejora respecto a Arq. Basic Chain	% de mejora respecto a Arq. Basic OAO
Animal	83.03	84.18	93.13	83.49	0.46	-0.69	-9.64
Human	96.76	77.45	86.49	92.63	-4.13	15.18	6.14
Airplane	96.46	76.85	85.17	94.15	-2.31	17.30	8.98
Truck	86.57	91.56	96.67	87.69	1.12	-3.87	-8.98
Cat	86.36	57.82	74.91	87.55	1.19	29.73	12.64
prom	89.84	77.57	87.27	89.10	-0.73	11.53	1.83

Tabla 54: Resultados experimento 3 sobre el dataset NORB.



(a) Arq. maestra GAD.

(b) Arq. básica GAD.

Figura 43: Matriz de confusión de los resultados del experimento 3 sobre el conjunto de prueba de NORB.

4.5.3. CIFAR-10

En nuestro primer experimento solamente se logró entrenar completamente la arquitectura básica como un grafo GAD. La arquitectura maestra está echa muy específicamente para resolver este problema con 10 neuronas de salida, de modo que al cambiar la cantidad de neuronas de salida y disminuir la cantidad de datos de entrada que le doy a la red, el modelo no logra aprender lo suficiente para lograr hacer una segregación de información balanceada por la red GAD, por lo cual las redes de los niveles inferiores se comportaban de una manera anómala ya que les llegaba información desbalanceada, causando pésimos resultados.

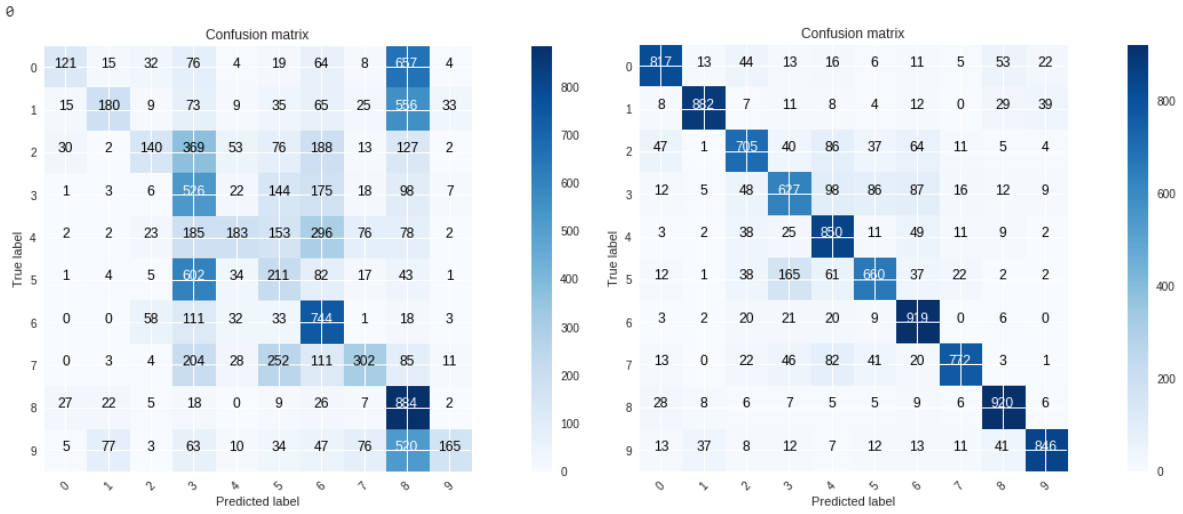
Por otro lado, la arquitectura básica si logró segregar correctamente la información. Esto se logró realizando el mismo pre-procesamiento explicado anteriormente. Además, se utilizó el

mismo cambio de learning rate y la misma cantidad de epochs.

Los resultados del primer experimento son:

Class	Benchmark Master	Benchmark Master Chain Decomposition	Bechmark Master OAO	Accuracy % Exp 1 Master	% de mejora respecto a Arq. MaestraOriginal	% de mejora respecto a Arq. Master Chain	% de mejora respecto a Arq. Master OAO
Airplane	88.01	73.90	-	-	-	-	-
Automobile	92.59	89.90	-	-	-	-	-
Bird	80.53	74.30	-	-	-	-	-
Cat	70.83	70.70	-	-	-	-	-
Deer	85.94	79.80	-	-	-	-	-
Dog	82.44	76.60	-	-	-	-	-
Frog	87.45	85.80	-	-	-	-	-
Horse	92.58	77.80	-	-	-	-	-
Ship	89.19	82.60	-	-	-	-	-
Truck	92.52	85.70	-	-	-	-	-
Prom	86.21	79.71	-	-	-	-	-
Class	Benchmark Basic	Benchmark Basic Chain Decomposition	Bechmark Basic OAO	Accuracy % Exp 1 Basic	% de mejora respecto a Arq. Basic Original	% de mejora respecto a Arq. Basic Chain	% de mejora respecto a Arq. Basic OAO
Airplane	84.58	70.3	85.85	85.46	0.88	15.16	-0.39
Automobile	93.80	65.30	87.93	92.74	-1.06	27.44	4.81
Bird	76.16	67.60	84.71	75.32	-0.84	7.72	-9.39
Cat	70.57	43.50	71.51	64.84	-5.73	21.34	-6.67
Deer	71.82	77.40	71.48	75.94	4.12	-1.46	4.46
Dog	82.09	51.80	72.54	75.77	-6.32	23.97	3.23
Frog	86.22	82.10	83.38	85.27	-0.95	3.17	1.89
Horse	90.62	74.90	85.74	90.4	-0.22	15.50	4.66
Ship	92.04	77.90	87.59	85.19	-6.85	7.29	-2.4
Truck	92.23	71.40	85.96	90.87	-1.36	19.47	4.91
Prom	84.01	68.22	81.67	82.18	-1.83	13.96	0.51

Tabla 55: Resultados experimento 1 sobre el dataset CIFAR-10.



(a) Arq. maestra GAD.

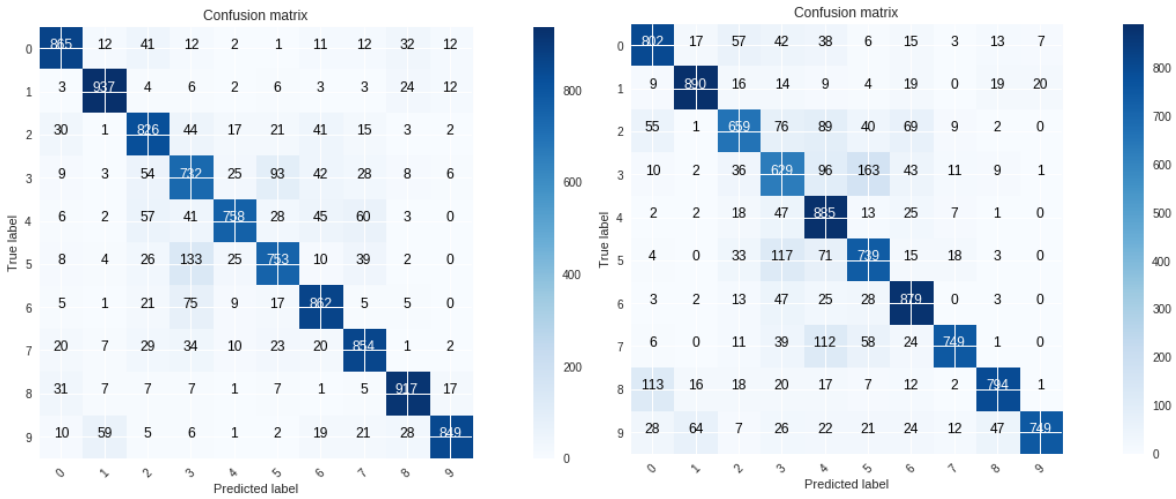
(b) Arq. básica GAD.

Figura 44: Matriz de confusión de los resultados del experimento 1 sobre el conjunto de prueba de CIFAR-10.

En nuestro segundo experimento, se realizó el mismo pre-procesamiento tanto para la arquitectura maestra como básica, pero se mantuvo un learning de $0,005$ constante par la arquitectura maestra y un $0,001$ para la arquitectura básica. Se entrenó solamente por 20 epochs, obteniendo los siguientes resultados:

Class	Benchmark Master	Benchmark Master Chain Decomposition	Bechmark Master OAO	Accuracy % Exp 2 Master	% de mejora respecto a Arq. MaestraOriginal	% de mejora respecto a Arq. Master Chain	% de mejora respecto a Arq. Master OAO
Airplane	88.01	73.90	89.46	90.64	2.63	16.74	1.18
Automobile	92.59	89.90	90.24	87.71	-4.88	-2.19	-2.53
Bird	80.53	74.30	84.56	77.20	-3.33	2.90	-7.36
Cat	70.83	70.70	74.77	77.16	6.33	6.46	2.39
Deer	85.94	79.80	73.64	89.18	3.24	9.38	15.54
Dog	82.44	76.60	72.12	79.18	-3.26	2.58	7.06
Frog	87.45	85.80	85.23	81.78	-5.67	-4.02	-3.45
Horse	92.58	77.80	85.89	81.96	-10.62	4.16	-3.93
Ship	89.19	82.60	90.62	89.64	0.45	7.04	-0.98
Truck	92.52	85.70	85.58	91.33	-1.19	5.63	5.75
Prom	86.21	79.71	83.21	84.58	-1.63	4.87	1.37
Class	Benchmark Basic	Benchmark Basic Chain Decomposition	Bechmark Basic OAO	Accuracy % Exp 2 Basic	% de mejora respecto a Arq. Basic Original	% de mejora respecto a Arq. Basic Chain	% de mejora respecto a Arq. Basic OAO
Airplane	84.58	70.30	79.83	85.95	1.37	15.65	6.12
Automobile	93.80	65.30	88.46	84.03	-9.77	18.73	-4.43
Bird	76.16	67.60	75.13	81.29	5.13	13.69	6.16
Cat	70.57	43.50	66.62	76.95	6.38	33.45	10.33
Deer	71.82	77.40	71.25	70.67	-1.15	-6.73	-0.58
Dog	82.09	51.80	80.49	75.43	-6.66	23.63	-5.06
Frog	86.22	82.10	79.74	77.80	-8.42	-4.30	-1.94
Horse	90.62	74.90	83.91	88.70	-1.92	13.80	4.79
Ship	92.04	77.90	91.68	88.90	-3.14	11.00	-2.78
Truck	92.23	71.40	93.62	88.12	-4.11	16.72	-5.5
Prom	84.01	68.22	81.07	81.78	-2.23	13.56	0.71

Tabla 56: Resultados experimento 2 sobre el dataset CIFAR-10.



(a) Arq. maestra GAD.

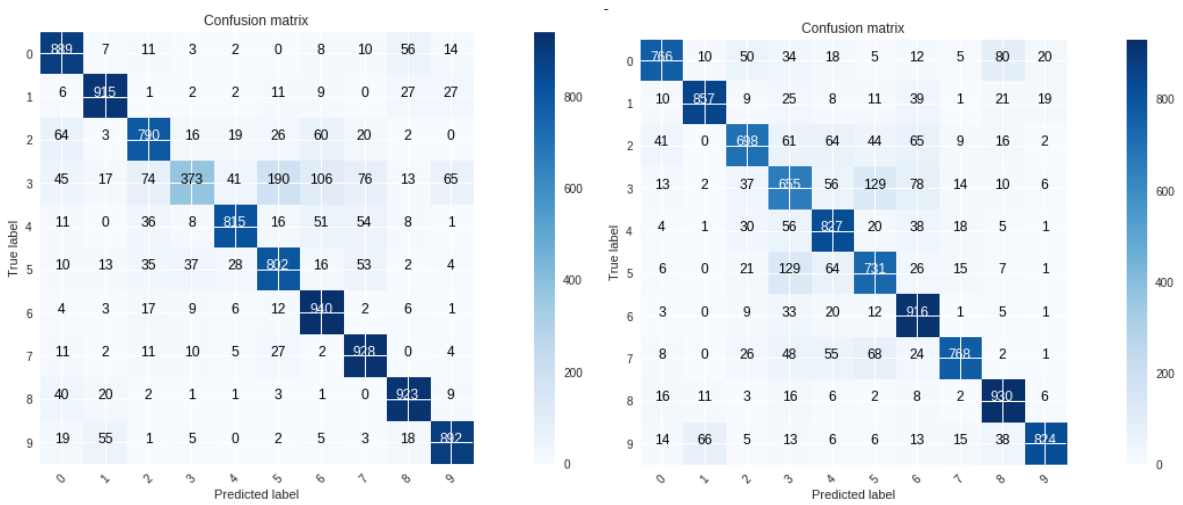
(b) Arq. básica GAD.

Figura 45: Matriz de confusión de los resultados del experimento 2 sobre el conjunto de prueba de CIFAR-10.

Finalmente, para nuestro tercer y último experimento sobre CIFAR-10, se realizaron las mismas preparaciones que el segundo experimento pero entrenando solamente la última capa de la red, lo cual entregó los siguientes resultados:

Class	Benchmark Master	Benchmark Master Chain Decomposition	Bechmark Master OAO	Accuracy % Exp 3 Master	% de mejora respecto a Arq. MaestraOriginal	% de mejora respecto a Arq. Master Chain	% de mejora respecto a Arq. Master OAO
Airplane	88.01	73.90	88.47	80.89	-7.12	6.99	-7.58
Automobile	92.59	89.90	91.23	94.41	1.82	4.51	3.18
Bird	80.53	74.30	78.52	80.78	0.25	6.48	2.26
Cat	70.83	70.70	37.68	80.39	9.56	9.69	42.71
Deer	85.94	79.80	81.18	88.68	2.74	8.88	7.5
Dog	82.44	76.60	78.67	81.65	-0.79	5.05	2.98
Frog	87.45	85.80	93.28	78.46	-8.99	-7.34	-14.82
Horse	92.58	77.80	88.46	85.98	-6.60	8.18	-2.48
Ship	89.19	82.60	92.78	83.49	-5.70	0.89	-9.29
Truck	92.52	85.70	89.95	87.71	-4.81	2.01	-2.24
Prom	86.21	79.71	82.02	84.24	-1.96	4.53	2.22
Class	Benchmark Basic	Benchmark Basic Chain Decomposition	Bechmark Basic OAO	Accuracy % Exp 3 Basic	% de mejora respecto a Arq. Basic Original	% de mejora respecto a Arq. Basic Chain	% de mejora respecto a Arq. Basic OAO
Airplane	84.58	70.3	88.92	86.10	1.52	15.80	-2.82
Automobile	93.80	65.30	91.63	96.57	2.77	31.27	4.94
Bird	76.16	67.60	78.73	78.24	2.08	10.64	-0.49
Cat	70.57	43.50	37.21	70.88	0.31	27.38	33.67
Deer	71.82	77.40	81.48	72.57	0.75	-4.83	-8.91
Dog	82.09	51.80	80.93	77.65	-4.44	25.85	-3.28
Frog	86.22	82.10	80.91	76.42	-9.80	-5.68	-4.49
Horse	90.62	74.90	92.96	86.18	-4.44	11.28	-6.78
Ship	92.04	77.90	92.87	83.06	-8.98	5.16	-9.81
Truck	92.23	71.40	89.04	98.66	6.43	27.26	9.62
Prom	84.01	68.22	81.47	82.63	-1.38	14.41	1.17

Tabla 57: Resultados experimento 3 sobre el dataset CIFAR-10.



(a) Arq. maestra GAD.

(b) Arq. básica GAD.

Figura 46: Matriz de confusión de los resultados del experimento 3 sobre el conjunto de prueba de CIFAR-10.

4.5.4. Unbalanced CIFAR-10

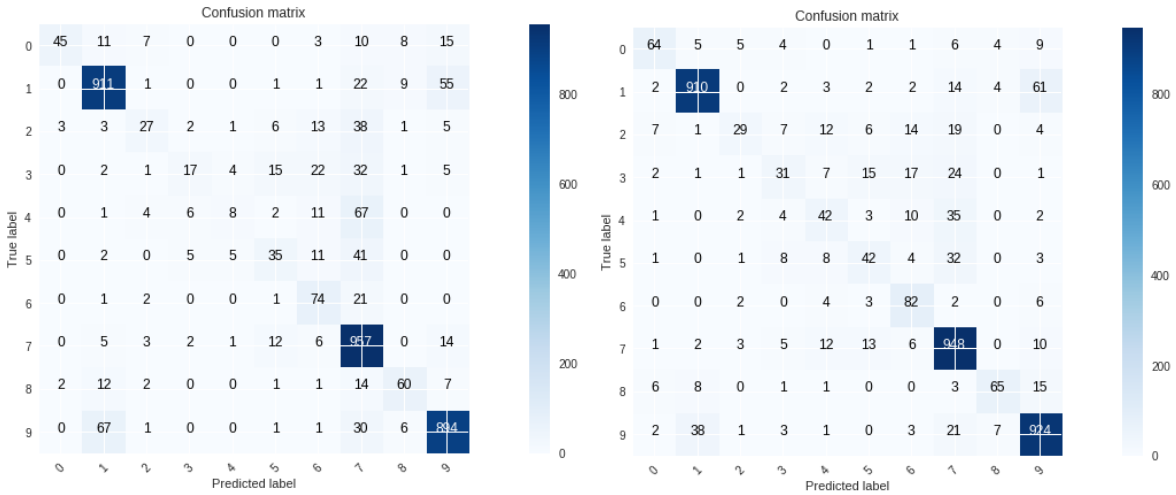
En el caso de CIFAR-10 desbalanceado, no fue posible entrenar ninguna de las dos arquitecturas desde 0: en este caso, tanto la arquitectura maestra como la arquitectura básica tuvieron problemas en uno o más de sus nodos al utilizarlas en la descomposición GAD. La arquitectura maestra ya demostró con anterioridad que no es robusta a la descomposición

en redes binarias, por lo cual una reducción de los datos de entrenamiento a cada nodo de la red maestra destruyó el rendimiento original, lo cual generó que algunos nodos simplemente no pudieran aprender absolutamente nada, causando resultados inconsistentes o clasificaciones de todos los datos de entrada a una clase única. La arquitectura básica sí demostró tener mayor resistencia a la descomposición binaria, pero al reducir los datos de entrenamiento, el comportamiento de cada nodo se vio sumamente afectado, resultando en nodos que no lograron aprender nada, generando los resultados inconsistentes. Por estas mismas causas, se omitió el benchmark con entrenamiento desde 0, ya que como la experimentación con las arquitecturas GAD no fue efectiva, no se tendría con qué comparar los resultados.

Para el segundo experimento, sí se logró entrenar la arquitectura maestra y la básica una descomposición GAD. El único cambio que se realizó fue aumentar a 100 el número de epochs de entrenamiento y se realizó una multiplicación del learning rate de $0,1$ por cada 30 epochs, obteniéndose los siguientes resultados:

Class	Benchmark Master	Benchmark Master Chain Decomposition	Bechmark Master OAO	Accuracy % Exp 2 Master	% de mejora respecto a Arq. Maestra Original	% de mejora respecto a Arq. Master Chain	% de mejora respecto a Arq. Master OAO
Airplane	36.38	45.45	16.83	49.52	13.14	4.07	32.69
Automobile	87.93	87.10	92.35	91.57	3.64	4.47	-0.78
Bird	32.52	44.44	4.55	34.44	1.92	-10.00	29.89
Cat	26.29	26.28	19.36	20.78	-5.51	-5.50	1.42
Deer	53.67	65.72	0.97	15.52	-38.15	-50.20	14.55
Dog	24.21	33.65	19.41	42.06	17.85	8.41	22.65
Frog	60.62	69.72	17.15	72.84	12.22	3.12	55.69
Horse	87.20	85.80	97.41	95.61	8.41	9.81	-1.8
Ship	67.54	74.08	16.18	57.25	-10.29	-16.83	41.07
Truck	89.30	85.04	89.78	89.71	0.41	4.67	-0.07
Prom	56.57	61.73	37.40	56.93	0.36	-4.80	19.53
Class	Benchmark Basic	Benchmark Basic Chain Decomposition	Bechmark Basic OAO	Accuracy % Exp 2 Basic	% de mejora respecto a Arq. Basic Original	% de mejora respecto a Arq. Basic Chain	% de mejora respecto a Arq. Basic OAO
Airplane	39.41	51.49	69.77	64.76	25.35	13.27	-5.01
Automobile	91.32	89.27	81.24	91.53	0.21	2.26	10.29
Bird	25.19	41.35	44.54	43.12	17.93	1.77	-1.42
Cat	32.54	23.31	30.49	32.57	0.03	9.26	2.08
Deer	28.19	43.51	39.31	50.84	22.65	7.33	11.53
Dog	39.17	25.36	40.48	44.65	5.48	19.29	4.17
Frog	77.78	63.59	77.76	82.51	4.73	18.92	4.75
Horse	95.51	88.53	94.82	95.58	0.07	7.05	0.76
Ship	62.54	64.67	68.69	66.67	4.13	2.00	-2.02
Truck	91.58	84.89	92.53	92.74	1.16	7.85	0.21
Prom	58.32	57.60	63.96	66.50	8.17	8.9	2.53

Tabla 58: Resultados experimento 2 sobre el dataset Unbalanced Cifar-10.



(a) Arq. maestra GAD.

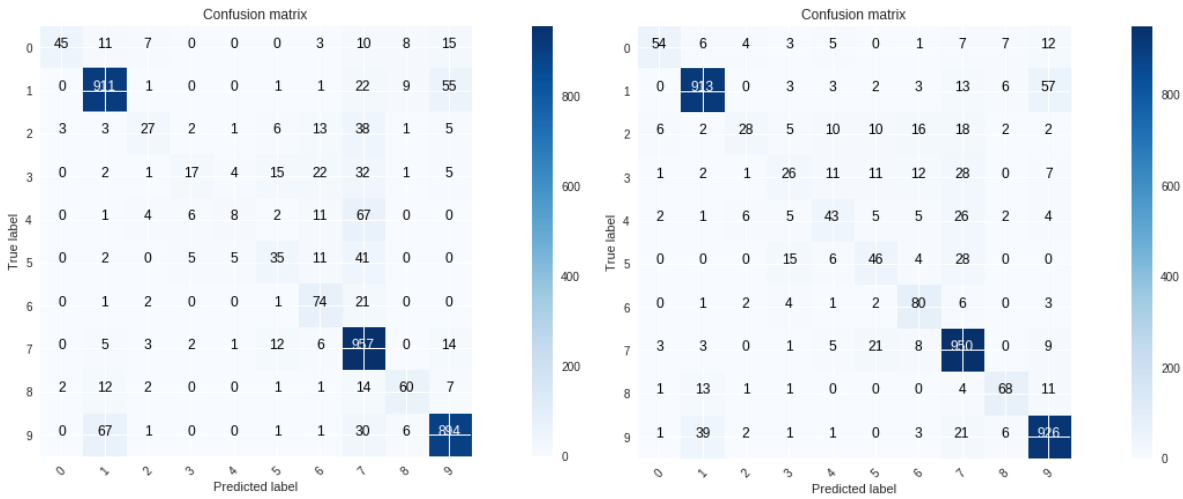
(b) Arq. básica GAD.

Figura 47: Matriz de confusión de los resultados del experimento 2 sobre el conjunto de prueba de Unbalanced CIFAR-10.

En el tercer experimento, se realizó el mismo cambio para la arquitectura maestra, resultando en:

Class	Benchmark Master	Benchmark Master Chain Decomposition	Bechmark Master OAO	Accuracy % Exp 3 Master	% de mejora respecto a Arq. Maestra Original	% de mejora respecto a Arq. Master Chain	% de mejora respecto a Arq. Master OAO
Airplane	36.38	45.45	53.47	51.27	14.89	5.82	-2.2
Automobile	87.93	87.10	93.95	91.63	3.70	4.53	-2.32
Bird	32.52	44.44	30.24	35.21	2.69	-9.23	4.97
Cat	26.29	26.28	16.78	19.36	-6.93	-6.92	2.58
Deer	53.67	65.72	9.25	15.74	-37.93	-49.98	6.49
Dog	24.21	33.65	24.26	41.85	17.64	8.20	17.59
Frog	60.62	69.72	74.58	73.87	13.25	4.15	-0.71
Horse	87.20	85.80	95.74	95.52	8.32	9.72	-0.22
Ship	67.54	74.08	48.31	56.18	-11.36	-17.90	7.87
Truck	89.30	85.04	86.94	89.35	0.05	4.31	2.41
Prom	56.57	61.73	53.35	57.00	0.43	-4.73	3.65
Class	Benchmark Basic	Benchmark Basic Chain Decomposition	Bechmark Basic OAO	Accuracy % Exp 3 Basic	% de mejora respecto a Arq. Basic Original	% de mejora respecto a Arq. Basic Chain	% de mejora respecto a Arq. Basic OAO
Airplane	39.41	51.49	49.63	55.36	15.95	3.87	5.73
Automobile	91.32	89.27	91.68	91.64	0.32	2.37	-0.04
Bird	25.19	41.35	38.42	34.27	9.08	-7.08	-4.15
Cat	32.54	23.31	35.34	33.73	1.19	10.42	-1.61
Deer	28.19	43.51	42.57	45.95	17.76	2.44	3.38
Dog	39.17	25.36	45.45	47.36	8.19	22.00	1.91
Frog	77.78	63.59	76.77	81.17	3.39	17.58	4.4
Horse	95.51	88.53	95.11	96.39	0.88	7.86	1.28
Ship	62.54	64.67	66.67	68.92	6.38	4.25	2.25
Truck	91.58	84.89	90.59	92.18	0.60	7.29	1.59
Prom	58.32	57.60	63.22	64.70	6.37	7.10	1.47

Tabla 59: Resultados experimento 3 sobre el dataset Unbalanced Cifar-10.



(a) Arq. maestra GAD.

(b) Arq. básica GAD.

Figura 48: Matriz de confusión de los resultados del experimento 3 sobre el conjunto de prueba de Unbalanced CIFAR-10.

4.5.5. BAAT

Para BAAT, al ser un dataset desbalanceado y con poca cantidad de datos, la única forma en que se logró entrenar los modelos GAD fue realizar un orden previo a la lista de clases de tal forma que en los nodos superiores se clasificara las clases con una mayor cantidad de datos de entrenamiento y en los nodos inferiores se clasificara las clases con menos datos de entrenamiento. Todo esto permite mejorar los resultados de los nodos superiores de la red. Lamentablemente, a pesar de los esfuerzos, no se logró entrenar una estructura GAD desde 0. La gran mayoría de los nodos presentaban bajo accuracy lo que provocaba un quiebre en la estructura.

Posteriormente, para el segundo experimento, sí se logro entrenar la arquitectura GAD realizando un entrenamiento con la misma arquitectura y configuración explicada en el bechmark pero entrenando en un inicio solamente con 5 epochs con optimizador Adam y posteriormente con 25 epochs con SGD para cada nodo, utilizando la lista de clases ordenadas como en el experimento 1. Los resultados obteniendo son los siguientes:

Class	Base Benchmark	Base Benchmark Chain Decomposition	Base Bechmark OAO	Accuracy % Exp 2	% de mejora respecto a Arq. Original	% de mejora respecto a Arq. Chain	% de mejora respecto a Arq. OAO
Vincent van Gogh	7.43	12.53	20.86	24.74	17.31	12.21	3.88
Edgar Degas	85.71	83.24	85.14	80.27	-5.44	-2.97	-4.87
Pablo Picasso	35.63	33.46	38.37	40.91	5.28	7.45	2.54
Pierre-Auguste Renoir	0.00	15.64	20.30	23.73	23.73	8.09	3.43
Albrecht Durer	30.77	31.46	35.46	34.96	4.19	3.50	-0.50
Paul Gauguin	0.00	10.95	17.10	23.19	23.19	12.24	6.09
Francisco Goya	18.97	20.63	24.48	22.64	3.67	2.01	-1.84
Rembrandt	50.63	49.23	53.46	55.73	5.10	6.50	2.27
Alfred Sisley	68.63	67.75	68.63	62.71	-5.92	-5.04	-5.92
Titian	64.71	65.38	65.71	63.54	-1.17	-1.84	-2.17
Prom	36.25	39.03	42.95	43.24	6.99	4.22	0.29

Tabla 60: Resultados experimento 2 sobre el dataset BAAT.

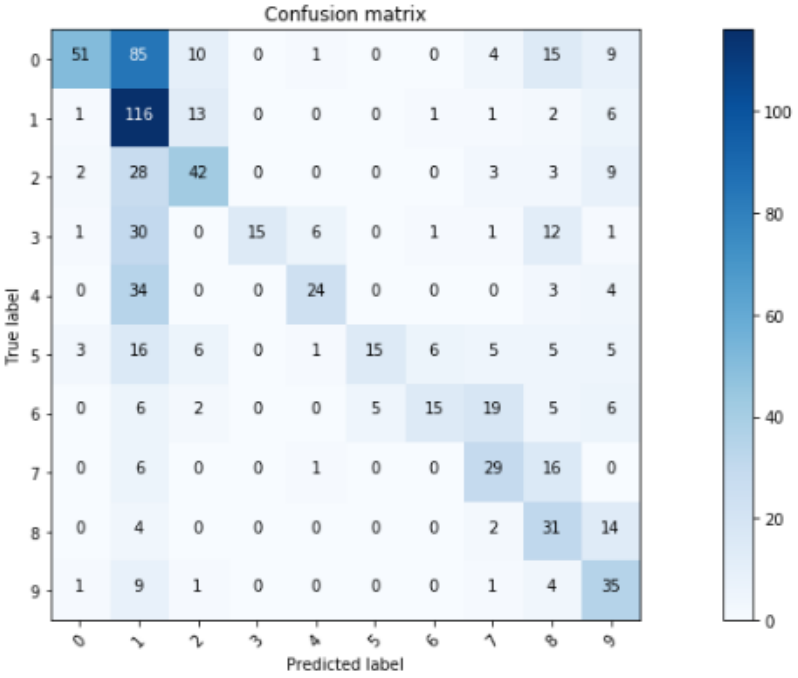


Figura 49: Matriz de confusión de las clases de BAAT para su arquitectura base con descomposición GAD entrenada con solo Fine Tunning.

Posteriormente, utilizando el mismo método pero congelando las todas las capas de la red pre-entrenada excepto la ultima, se llega al siguiente resultado en el tercer experimento:

Class	Base Benchmark	Base Benchmark Chain Decomposition	Base Bechmark OAO	Accuracy % Exp 2	% de mejora respecto a Arq. Original	% de mejora respecto a Arq. Chain	% de mejora respecto a Arq. OAO
Vincent van Gogh	7.43	12.53	25.47	30.24	22.81	17.71	4.77
Edgar Degas	85.71	83.24	85.58	80.79	-4.92	-2.45	-4.79
Pablo Picasso	35.63	33.46	40.19	41.55	5.92	8.09	1.36
Pierre-Auguste Renoir	0.00	15.64	23.79	25.32	25.32	9.68	1.53
Albrecht Durer	30.77	31.46	38.97	38.78	8.01	7.32	-0.19
Paul Gauguin	0.00	10.95	19.68	25.06	25.06	14.11	5.38
Francisco Goya	18.97	20.63	26.68	26.11	7.14	5.48	-0.57
Rembrandt	50.63	49.23	54.13	54.97	4.34	5.74	0.84
Alfred Sisley	68.63	67.75	69.44	64.94	-3.69	-2.81	-4.50
Titian	64.71	65.38	66.18	67.91	3.20	2.53	1.73
Prom	36.25	39.03	45.01	45.57	9.32	6.54	0.56

Tabla 61: Resultados experimento 3 sobre el dataset BAAT.

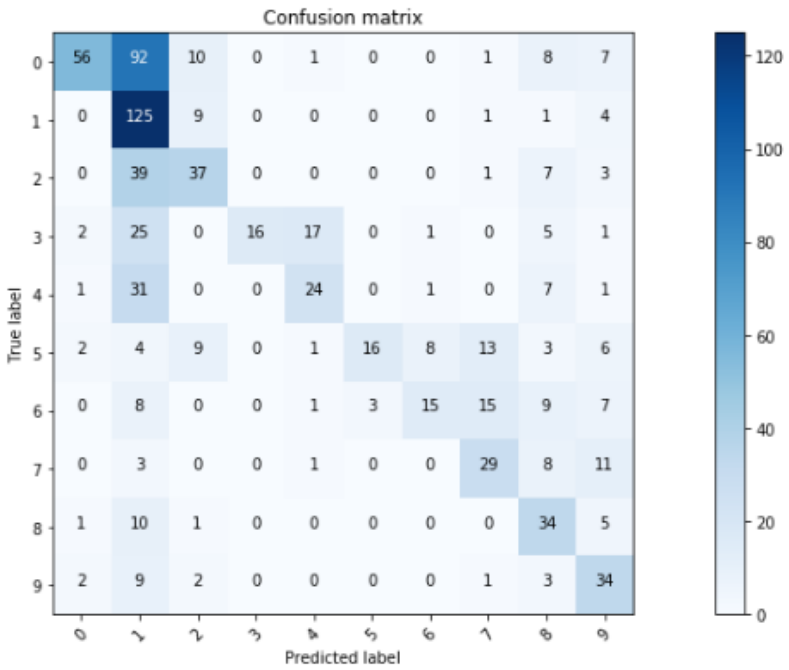


Figura 50: Matriz de confusión de las clases de BAAT para su arquitectura base con descomposición GAD entrenada mediante Fine Tunning con capas congeladas.

CAPÍTULO 5
CONCLUSIONES

Para esta sección, se concluirá de la siguiente manera:

- Conclusiones de comportamiento de la arquitectura DAG sobre los distintos dataset.
- Conclusión general de la metodología, tomando en cuenta las conclusiones individuales por dataset.
- Trabajos futuros y como poder mejorar la arquitectura.

Los experimentos realizados han demostrado que esta arquitectura funciona bien para ciertos casos y mal para otros:

5.1. MNIST

En MNIST, el método propuesto logra una accuracy promedio del $98{,}90$ % usando la arquitectura básica sin pre-entrenar, resultado ligeramente mejor que el $98{,}87$ % del método estándar (sin descomposición), que se obtiene usando la arquitectura maestra. Esto se debe a que la arquitectura básica no está tan optimizada para la metodología clásica como la arquitectura maestra, otorgando la flexibilidad necesaria para usarla en la malla GAD y haciendo el modelo más robusto en caso de que algún nodo no se entrenara lo suficiente. Comparando los métodos sobre la misma arquitectura y el mismo esquema de entrenamiento, observamos que el método propuesto empeora el desempeño del método estándar en la mayoría de las clases (con un promedio de ~ -1 %) si se usa la arquitectura maestra. Si se usa la arquitectura básica, los resultados dependen más del esquema de entrenamiento, pero, considerando el promedio sobre las diferentes clases, no se observan mejoras consistentes o significativas.

Comparando con otras técnicas de descomposición, observamos que el método propuesto mejora en todas las clases el desempeño de la arquitectura OAHO, independiente de la arquitectura o del esquema de entrenamiento que se adopte, con una mejora promedio en torno al 1 % y máximos de $3-4$ %. Con respecto a OAO, se observa una situación similar, que aparece aún más favorable cuando se considera sólo la arquitectura básica. En este último caso, se observan mejoras de 6 % o incluso 10 % en algunas clases.

La arquitectura básica de MNIST tiene mejores resultados en términos de mejorar el accuracy con una descomposición GAD que la arquitectura maestra, alcanzando en el experimento 1 un accuracy promedio 0.32 % mayor que el benchmark básico. Esto se debe a que la arquitectura básica no esta tan especializada para MNIST como la arquitectura maestra, permitiendo

la flexibilidad en la malla GAD y haciendo el modelo más robusto en caso de que algún nodo no se entrenara lo suficiente.

Se destaca que la arquitectura GAD tiene un mejor desempeño promedio que una descomposición en cadena, es más, tanto la arquitectura maestra como básica descompuesta via GAD tienen un accuracy bastante similar al benchmark de la arquitectura con N neuronas de salida, con un margen de error de $\pm$ 1 % por lo cual para este dataset, la arquitectura si mejora considerable la división clásica por cadena.

Finalmente, podemos observar que el accuracy promedio de las arquitecturas GAD supera el accuracy promedio de la descomposición OAO con mayoría de votos tanto en la arquitectura maestra como básica. Destacamos utilizando la arquitectura básica GAD se generan mejoras de hasta un 10 % para la clase 5 al usar la arquitectura básica y el sacrificio realizado en ciertas clases no supera el 2.5 %.

5.2. NORB

Los resultados de NORB fueron bastante parecidos a los resultados obtenidos en MNIST. De los tres experimentos realizados solamente el experimento 2 tuvo aumentos promedio positivos respecto a sus benchmark correspondientes el cual fue de 1.92 % para la arquitectura maestra y 0.10 para la básica. A pesar de que las variaciones en el accuracy son bastante pequeñas, como NORB es un dataset para el cual la accuracy alcanza un 89 % o más, podemos tomar estos resultados como favorables.

Las clases menos representativas de NORB sí subieron considerablemente, alcanzándose hasta un aumento del 8 % para la clase Truck en el experimento 2 de la arquitectura maestra.

Finalmente, al igual que MNIST, la arquitectura GAD superó con creces la descomposición en cadena y la descomposición OAO con mayoría de votos, generando mejoras de hasta un 12 % y 4 % en promedio respectivamente. Una observación peculiar, es el desempeño en ciertas clases se deteriora bastante al descomponer la arquitectura como GAD en comparación a OAO con mayoría de votos, pero a su vez, aumenta considerablemente en otras clases. Por ejemplo, podemos ver en los resultados del experimento 1 que la clase animal disminuye un $12{,}88$ % al comparar GAD con OAO mayoría de votos, pero a su vez, las demás clases aumentan bastante.

5.3. CIFAR-10

Para CIFAR-10, no se logró mejorar el accuracy promedio de la arquitectura GAD en comparación al benchmark del estado del arte: éste obtiene una accuracy de $86{,}21$ % usando las arquitectura maestra, que supera el $84{,}58$ % obtenido por el método propuesto, usando la

arquitectura maestra pre-entrenada. Si se analizan las clases individualmente, se logra aumentar muy poco el accuracy de clases que el método clásico clasifica de peor manera , pero a costa de un disminución muy grande del accuracy de otras clases.

Por otro lado, se logró superar el benchmark promedio de una descomposición en cadena en comparación con la arquitectura GAD, aumentando en total un 13.95 % para la arquitectura básica en el primer experimento, un 4.87 % y un 13.56 % de aumento en el segundo experimento para la arquitectura maestra y básica respectivamente y un 4.53 % y un 14.41 % para la arquitectura maestra y básica respectivamente en el tercer experimento. También se logró incrementar el accuracy promedio en todos los experimentos respecto a la descomposición OAO con mayoría de votos, incluso algunas clases aumentaron su accuracy en más de 40 %

5.4. Unbalanced CIFAR-10

Para CIFAR-10 desbalanceado, se obtuvieron resultados interesantes. Primero, podemos notar que, para el primer experimento, no se logró entrenar ninguna de las dos arquitecturas GAD. Para la arquitectura maestra este resultado era de esperarse, ya que se demostró con anterioridad en CIFAR-10, que la arquitectura no es flexible al momento de dividirla en un grafo GAD: la arquitectura muestra dificultades para aprender el problema correspondiente a cada nodo del grafo, posiblemente por la disminución que se produce del número de datos de entrenamiento de clases que pueden ya ser minoritarias. Es interesante destacar que la arquitectura básica tampoco se logró entrenar, a pesar de que demostró flexibilidad a la división, en los experimentos de CIFAR-10 balanceado.

Esta situación cambia cuando se usan los esquema de entrenamiento correspondientes a los experimentos 2 y 3, basados en pre-entrenamiento. El método propuesto logra una accuracy promedio de $66{,}50$ % usando la arquitectura básica con la tercera configuración, lo que supera el $58{,}32$ %, correspondiente al mejor resultado del método clásico (sin descomposición).

Tanto para el segundo como para el tercer experimento, la arquitectura básica se desempeñó mucho mejor que la arquitectura maestra en el método propuesto. Para el segundo experimento, se aumentó alrededor de un 8 % el accuracy promedio alcanzado por el método clásico. Para el tercer experimento, este aumento es de alrededor de un 6 %. Si utiliza en cambio la arquitectura maestra, las mejoras se reducen notablemente, aunque en este caso también lo hace el rendimiento promedio del método clásico que cae de $58{,}32$ % a $56{,}57$ %. Usando la arquitectura básica, el método propuesto mejora la clasificación de todas las clases, con máximos de mejora que superan el 20 %. Las mejoras menos importantes (< 1 %) se producen justamente sobre las clases mejor representadas en el dataset (Automobile, Horse, Truck). Comparando con otros métodos de descomposición, vemos que el $66{,}50$ % que se alcanza con la arquitectura básica y la tercera configuración de entrenamiento, supera tanto a OAHO como a OHO, que logran un $61{,}73$ % y un $63{,}96$ % respectivamente. Si se fuerza a la red a usar la arquitectura maestra, se pierde alrededor de 5 % de accuracy en promedio

respecto de OAHO, pero usando la arquitectura básica el rendimiento promedio es $7-9\,\%$ superior al de este método, con máximos de hasta $20\,\%$ en algunas clases. Las mejoras son respecto a OAO ocurren independientemente de la arquitectura utilizada pero son más importantes cuando se usa la primera, alcanzando incluso un promedio de $19{,}53\,\%$ usando la primera configuración de entrenamiento.

5.5. BAAT

BAAT fue uno de los dataset más interesantes de trabajar. Primero, podemos notar que la metodología tradicional no clasifica correctamente ninguna instancia de las clases Pierre-Auguste Renoir y Paul Gauguin. En cambio, la accuracy sobre estas clases supera el $20\,\%$ usando el método propuesto, con una mejora promedio de alrededor de $7\,\%$ si se utiliza el primer esquema de entrenamiento y de alrededor de $9\,\%$ si se utiliza el segundo. En general, todos los métodos de descomposición mejorar por sobre la metodología tradicional (sin descomposición). Usando el primer esquema de entrenamiento, OAHO obtiene una accuracy del $15{,}64\,\%$ y $10{,}95\,\%$ respectivamente sobre las clases patológicas antes mencionadas, mientras que OAO logra un $20{,}30\,\%$ y $24{,}48\,\%$ respectivamente. En promedio, el método propuesto incrementa la accuracy de OAHO de un $4{,}22\,\%$ ó un $6{,}54\,\%$, dependiendo del esquema de entrenamiento. Con respecto a OAO, los márgenes de mejoráas son más estrechos, aunque se observan para ambas configuraciones del entrenamiento.

5.6. Conclusiones Generales

En esta memoria, se presentan resultados preliminares correspondientes a la aplicación de un método de descomposición denominado GAD para el entrenamiento de redes neuronales en problemas de clasificación con múltiples categorías. Este método consiste en organizar una serie de redes binarias como nodos de un grafo de decisión. Los resultados se compararon con la metodología tradicional consistente en el entrenamiento de una única red neuronal y con dos métodos de descomposición propuestos en la literatura. El desempeño se evaluó utilizando la accuracy por clase, la accuracy promedio por clase y la mejora neta obtenida (differencia de accuracy) con respecto a los métodos de referencia.

Nuestros experimentos sobre dataset balanceados, MNIST, NORB y CIFAR-10, nos permiten concluir que la metodología puede incrementar levemente la accuracy promedio obtenida por el método tradicional, con una atenta selección de la arquitectura de red a utilizar y la metodología de entrenamiento. Es probable que la mejora no sea sustancialmente grande porque se trata de problemas sobre los cuales los resultados ya son muy buenos. Sin embargo, considerando los enormes cuidados que hay que tener para que la metodología de descomposición propuesta funcione, nuestros resultados sugieren que su uso, en problemas balanceados, sólo se justifica si realmente se requiere aumentar el accuracy de clases menos representativas en un par de decimales. Se presume que la incapacidad de aumentar nota-

blemente el accuracy en dataset ya estudiados, es que las arquitecturas utilizadas están muy optimizadas para resolver el problema es específico. Un re-planteamiento del problema de decisión puede producir resultados más negativos que positivos sobre estos modelos.

Otra conclusión importante, obtenida sobre estos primeros 3 problemas, es que la metodología propuesta supera, en la mayoría de los casos y en ocasiones con márgenes muy amplios, la accuracy promedio de las dos técnicas de descomposición usadas como referencia: OAHO y OAO. Esta observación es importante porque, aún si los márgenes de mejora son estrechos con respecto a la metodología tradicional, el método propuesto puede limitar los daños que produce la aplicación de otras técnicas de descomposición a problemas donde éste no es el enfoque apropiado.

Avanzando en la investigación, se formuló la hipótesis de que los métodos de descomposición podrían ser más útiles en problemas desbalanceados. Por este motivo, se estudió el desempeño de los métodos en dos dataset de esta naturaleza: una versión desbalanceada de CIFAR-10 y un dataset naturalmente desbalanceado denominado BAAT.

Respecto a CIFAR-10 desbalanceado, podemos notar que la arquitectura GAD básica tiene un excelente rendimiento, aumentando un 7 % y un 6 % el accuracy promedio de la arquitectura básica y superando por creces los benchmark tanto básicos como maestros de cada experimento. Estos resultados nos indican que la arquitectura GAD tiene un gran uso para problemas con data desbalanceada, si es que se está dispuesto a pagar el tiempo de cómputo que se requiere para poder entrenar la gran cantidad de redes. La estructura maestra no se comportó muy bien en nuestro dataset desbalanceado. Se presume que al ser una estructura tan rígida, la descomposición en GAD y la reducción de los datos de entrenamiento destruyera por completo la arquitectura.

Al observar los resultados con BAAT, podemos decir rápidamente que si no se tienen demasiados ejemplos de entrenamiento, intentar entrenar una estructura GAD desde 0 es bastante difícil, por lo cual, no se recomienda este método de entrenamiento si la cantidad de datos es pequeña. Por otro lado, el método GAD es una excelente técnica para potenciar el accuracy de las clases minoritarias si se tiene el tiempo para poder entrenar una red base y posteriormente realizar la división GAD.

Respecto a los resultados de los grafos GAD tanto maestro como básico en comparación a sus descomposiciones en cadena o en OAO con mayoría de votos, podemos decir que la descomposición GAD tiene mejores resultados tanto en arquitecturas maestras como básicas. El grafo GAD tiende a mejorar el accuracy por clase y aveces el accuracy promedio en comparación a la descomposición en cadena. Mientras que el grafo GAD siempre tiende a mejorar el accuracy general en comparación a la descomposición OAO con mayoría de votos, tiende a aumentar algunas clases explosivamente pero a costa de reducir otras clases de forma abrupta. Por lo cual, en conclusión, el método GAD en general se comporta mejor que las otras dos descomposiciones, pero se tiene que tomar en cuenta los efectos sobre el accuracy por clase al momento de utilizar la arquitectura.

Tomando todo esto en cuenta, se considera que la descomposición GAD tendría un mayor impacto en problemas con data desbalanceada y que se encuentren en un estancamiento en terminó de accuracy. Además, a partir de los resultados del dataset BAAT, se aconseja utilizar un modelo ya pre-entrenado y realizar la descomposición GAD mediante un entrenamiento de solo fine-tunning o fine-tunning con capas congeladas en el caso de que no se posea muchos datos de entrenamiento.

5.7. Trabajos Futuros

Como trabajos futuros, el más plausible a investigar es cómo funciona la división GAD con modelos sumamente sencillos y con esto detectar si se puede igualar un modelo complicado y denso en corporación a una arquitectura bastante simple pero dividida como grafo GAD. La complejidad de esto radica en que la arquitectura de red neuronal a utilizar de todos modos debe ser sintonizada explícitamente a un problema específico y este es un proceso computacionalmente exhaustivo.

Otra posibilidad es un estudio más profundo sobre el orden en que se utilizan las clases en el grafo. En general, en esta memoria se utilizó una lista aleatoria, pero el orden en como el grafo compara las clases es sumamente importante, punto demostrado en el estudio del dataset BAAT, donde la única forma de lograr entrenar un modelo GAD fue modificar el criterio por defecto y ordenar la lista de tal forma que las clases mayoritarias se clasificaran primero.

Finalmente, sería interesante estudiar si este modelo sirve para encontrar clases que se opacan entre si, ya que como cada nodo compara dos clases distintas, se puede estudiar cada nodo del grafo y encontrar cuáles son los nodos de división crítica, o en cuál nodo se genera una división de la información de manera sesgada y detectar las clases que son difíciles para la red o detectar las clases que se opacan entre sí.

REFERENCIAS BIBLIOGRÁFICAS

[1] Thad Hughes and Keir Mierle. Recurrent neural networks for voice activity detection. pages 7378–7382. IEEE, May 2013.

[2] Dan Cireşan, Ueli Meier, and Juergen Schmidhuber. Multi-column Deep Neural Networks for Image Classification. arXiv:1202.2745 [cs], February 2012. arXiv: 1202.2745.

[3] S. M. Kamruzzaman, Ahmed Ryadh Hasan, Abu Bakar Siddiquee, and Md Ehsanul Hoque Mazumder. Medical diagnosis using neural network. arXiv:1009.4572 [cs], September 2010. arXiv: 1009.4572.

[4] John C. Platt, Nello Cristianini, and John Shawe-Taylor. Large Margin DAGs for Multiclass Classification. In S. A. Solla, T. K. Leen, and K. Müller, editors, Advances in Neural Information Processing Systems 12, pages 547–553. MIT Press, 2000.

[5] Guobin Ou and Yi Lu Murphey. Multi-class pattern classification using neural networks. Pattern Recognition, 40(1):4–18, January 2007.

[6] Vergara Sebastián. EVALUACIÓN DE FRAMEWORKS DE SOFTWARE PARA LA IMPLEMENTACIÓN DE REDES NEURONALES ARTIFICIALES. 2018.

[7] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Internal Representations by Error Propagation. Technical Report ICS-8506, CALIFORNIA UNIV SAN DIEGO LA JOLLA INST FOR COGNITIVE SCIENCE, CALIFORNIA UNIV SAN DIEGO LA JOLLA INST FOR COGNITIVE SCIENCE, September 1985.

[8] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5(4):115–133, December 1943.

[9] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. Neural Networks, 2(5):359–366, January 1989.

[10] Creating a movie recommender using Convolutional Neural Networks.

[11] Yi L. Murphey, Hong Guo, and Lee A. Feldkamp. Neural Learning from Unbalanced Data. Applied Intelligence, 21(2):117–128, September 2004.

[12] Baptiste Rocca. Handling imbalanced datasets in machine learning, January 2019.

[13] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, November 1998.

[14] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[15] Y. LeCun, Fu Jie Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004., volume 2, pages II–104 Vol.2, June 2004.

[16] Best Artworks of All Time.

[17] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-Supervised Nets. arXiv:1409.5185 [cs, stat], September 2014. arXiv: 1409.5185.

[18] Zenglin Shi, Yangdong Ye, and Yunpeng Wu. Rank-based pooling for deep convolutional neural networks. Neural Networks: The Official Journal of the International Neural Network Society, 83:21–31, November 2016.

[19] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for Simplicity: The All Convolutional Net. arXiv:1412.6806 [cs], December 2014. arXiv: 1412.6806.

[20] Forest Agostinelli, Matthew Hoffman, Peter Sadowski, and Pierre Baldi. Learning Activation Functions to Improve Deep Neural Networks. arXiv:1412.6830 [cs, stat], December 2014. arXiv: 1412.6830.