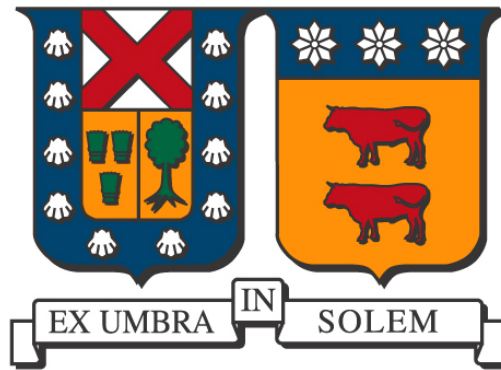


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



“HYPERBOLIC EMBEDDINGS FOR EFFICIENT
LOW-DIMENSIONAL IMAGE REPRESENTATIONS IN
COMPUTER VISION TASKS”

IAN PAUL ROBERTS VALENZUELA

TESIS PARA OPTAR AL TÍTULO DE MAGÍSTER EN CIENCIAS DE LA
INGENIERÍA INFORMÁTICA

PROFESOR GUÍA	MAURICIO ARAYA LÓPEZ
EXAMINADOR INTERNO	RICARDO ÑANCULEF
EXAMINADOR EXTERNO	JUAN PABLO VASCONEZ

ABRIL - 2026



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☐ Memoria o trabajo de título; ☒ Tesis de Postgrado;

Título del trabajo: Hyperbolic Embeddings for Efficient Low-Dimensional Image Representations in Computer Vision Tasks

Nombre del candidato(a): Ian Paul Roberts Valenzuela

Carrera / Grado: Magíster en Ciencias de la Ingeniería Informática

Campus: Casa Central Valparaíso ; **Departamento:** Departamento de Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Mauricio Araya López, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 12/03/2026

Firma:

Estudiante o Candidato(a):

Fecha: 20/03/2026

Firma:

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Resumen

El espacio hiperbólico se ha convertido en una alternativa prometedora al espacio euclidiano para representaciones de datos de alta dimensión, gracias a su capacidad natural para representar estructuras jerárquicas. Si bien la mayoría de los trabajos previos sobre estas representaciones hiperbólicas se han centrado en dominios basados en texto, donde las jerarquías son explícitas, este trabajo investiga su aplicación a las representaciones de imágenes, donde las relaciones jerárquicas son implícitas. Aprovechamos la capacidad de los *embeddings* de Poincaré para descubrir relaciones clase-subclase en conjuntos de datos de imágenes, aplicándolo a las tareas de *Matrioshka Embeddings* y *Deep Clustering*. Los resultados experimentales en conjuntos de datos como CIFAR10/100 e Imagenette muestran que los *embeddings* hiperbólicos superan a sus contrapartes euclidianas en *Matrioshka Embeddings*, particularmente en tareas *downstream* de baja dimensión donde las restricciones de dimensionalidad son críticas. Sin embargo, en *Deep Clustering*, el enfoque hiperbólico presenta desafíos de estabilidad que limitaron su aplicabilidad. Por último, analizamos las limitaciones de los enfoques hiperbólicos actuales en el dominio de la imagen y discutimos futuras direcciones de investigación para el aprendizaje de representación escalable y consciente de la jerarquía.

Palabras clave: visión por computador, embeddings, matrioshka embeddings, deep clustering, espacio hiperbólico, geometría no-euclidiana

Abstract

Hyperbolic space has emerged as a promising alternative to Euclidean space for high-dimensional data representations, due to its natural ability to represent hierarchical structures. While most prior work on hyperbolic representations has focused on text-based domains, where hierarchies are explicit, this work investigates their application to image representations, where hierarchical relationships are implicit. We leverage the ability of Poincaré embeddings to discover class–subclass relationships in image datasets, applying them to the tasks of Matrioshka Embeddings and Deep Clustering. Experimental results on datasets such as CIFAR10/100 and Imagenette show that hyperbolic embeddings outperform their Euclidean counterparts in Matrioshka Embeddings, particularly in low-dimensional downstream tasks where dimensionality constraints are critical. However, in Deep Clustering, the hyperbolic approach presents stability challenges that limited its applicability. Finally, we analyze the limitations of current hyperbolic approaches in the image domain and discuss future research directions for scalable, hierarchy-aware representation learning.

Keywords: computer vision, embeddings, matrioshka embeddings, deep clustering, hyperbolic space, non-euclidean geometry

Índice general

1. Introducción	1
1.1. Objetivos	2
1.1.1. General	2
1.1.2. Específicos	2
1.2. Estructura de la tesis	2
2. Estado del Arte	5
2.1. Aprendizaje de Representaciones Visuales	5
2.2. Geometrías No Euclidianas	5
2.2.1. Espacio Hiperbólico	5
2.3. Redes Neuronales Hiperbólicas	6
2.4. Embeddings Hiperbólicos en Tareas de Visión	7
2.5. Matrioshka Representation Learning	7
2.6. Deep Clustering	8
2.7. Arquitecturas Base	9
2.7.1. ResNet	9
2.7.2. EfficientNetV2	10
3. Metodología	11
3.1. Hipótesis	11
3.1.1. Hipótesis Específicas	11
3.2. Materiales	12
3.2.1. Stack Tecnológico	12
3.2.2. Ambiente de Desarrollo	12
3.3. HypTorch	12
3.3.1. Capa ToPoincaré	12

3.3.2. Capa FromPoincaré	13
3.3.3. Capa Lineal Hiperbólica (HypLinear)	13
3.3.4. Capa de Regresión Logística Multiclase Hiperbólica (HMLR)	13
3.3.5. Distancia Hiperbólica	14
3.4. Datasets	14
3.4.1. CIFAR10	14
3.4.2. CIFAR100	16
3.4.3. Imagenette	16
4. Hyperbolic Matrioshka Embeddings	19
4.1. Arquitectura	20
4.2. Entrenamiento	21
4.3. Evaluación	22
4.4. Resultados	22
4.4.1. CIFAR10 - ResNet50	22
4.4.2. CIFAR10 - EfficientNetV2	26
4.4.3. CIFAR100 - ResNet50	30
4.4.4. CIFAR100 - EfficientNetV2	34
4.4.5. Imagenette - ResNet50	38
4.4.6. Imagenette - EfficientNetV2	42
4.5. Resumen de Resultados	46
4.6. Análisis de la Curvatura Aprendida	47
4.7. Análisis Exploratorio: Pesos Preentrenados	48
4.7.1. Configuración	48
4.7.2. Resultados	48
5. Extensiones a Recuperación de Imágenes	51
5.1. Recuperación de Imágenes con Embeddings Hiperbólicos	51
5.1.1. Métrica de Evaluación: HmAP	52
5.1.2. Experimentos	52
5.1.3. Resultados	53
5.2. Hyperbolic DeepCluster	54
5.2.1. Arquitectura	55
5.2.2. Entrenamiento	55
5.2.3. Evaluación	56
5.2.4. Resultados	56

6. Conclusiones	59
6.1. Conclusiones Generales	59
6.2. Conclusiones Específicas	60
6.3. Trabajo Futuro	61
A. Figuras de Entrenamiento	63
A.1. CIFAR10 - ResNet50	63
A.1.1. CIFAR10 - EfficientNetV2	69
A.1.2. CIFAR100 - ResNet50	76
A.1.3. CIFAR100 - EfficientNetV2	82
A.1.4. Imagenette - ResNet50	88
A.1.5. Imagenette - EfficientNetV2	94
B. Figuras DeepCluster	101

Índice de figuras

2.1. Bloque residual ResNet.	10
3.1. Ejemplo de clases CIFAR10.	15
3.2. Jerarquía de CIFAR10	15
3.3. Ejemplo de clases CIFAR100.	16
3.4. Jerarquía de CIFAR100.	17
3.5. Ejemplos de clases Imagenette.	17
4.1. Arquitecturas de los tres modelos evaluados.	20
4.2. Método <i>Matrioshka Embeddings</i> .	21
4.3. Resultados entrenamiento ResNet50 euclidiano sobre CIFAR10.	23
4.4. Resultados entrenamiento ResNet50 hiperbólico sobre CIFAR10.	23
4.5. Resultados entrenamiento ResNet50 híbrido sobre CIFAR10.	24
4.6. Comparación entrenamiento ResNet50 sobre CIFAR10 para $d = 4$.	25
4.7. Comparación entrenamiento ResNet50 sobre CIFAR10 para $d = 8$.	25
4.8. Comparación entrenamiento ResNet50 sobre CIFAR10 para $d = 16$.	26
4.9. Resultados entrenamiento EfficientNetV2 euclidiano sobre CIFAR10.	27
4.10. Resultados entrenamiento EfficientNetV2 hiperbólico sobre CIFAR10.	27
4.11. Resultados entrenamiento EfficientNetV2 híbrido sobre CIFAR10.	28
4.12. Comparación entrenamiento EfficientNetV2 sobre CIFAR10 para $d = 4$.	29
4.13. Comparación entrenamiento EfficientNetV2 sobre CIFAR10 para $d = 256$.	29
4.14. Resultados entrenamiento ResNet50 euclidiano sobre CIFAR100.	30
4.15. Resultados entrenamiento ResNet50 hiperbólico sobre CIFAR100.	31
4.16. Resultados entrenamiento ResNet50 híbrido sobre CIFAR100.	31
4.17. Comparación entrenamiento ResNet50 sobre CIFAR100.	32
4.18. Comparación validación ResNet50 sobre CIFAR100.	32

4.19. Comparación entrenamiento ResNet50 sobre CIFAR100 para $d = 32$.	33
4.20. Comparación entrenamiento ResNet50 sobre CIFAR100 para $d = 128$.	34
4.21. Resultados entrenamiento EfficientNetV2 euclidiano sobre CIFAR100.	35
4.22. Resultados entrenamiento EfficientNetV2 hiperbólico sobre CIFAR100.	35
4.23. Resultados entrenamiento EfficientNetV2 híbrido sobre CIFAR100.	36
4.24. Comparación entrenamiento EfficientNetV2 sobre CIFAR100.	37
4.25. Comparación entrenamiento EfficientNetV2 sobre CIFAR100.	37
4.26. Resultados entrenamiento ResNet50 euclidiano sobre Imagenette.	38
4.27. Resultados entrenamiento ResNet50 hiperbólico sobre Imagenette.	38
4.28. Resultados entrenamiento ResNet50 híbrido sobre Imagenette.	39
4.29. Comparación entrenamiento ResNet50 sobre Imagenette.	40
4.30. Comparación validación ResNet50 sobre Imagenette.	40
4.31. Comparación entrenamiento ResNet50 sobre Imagenette para $d = 4$.	41
4.32. Comparación entrenamiento ResNet50 sobre Imagenette para $d = 256$.	41
4.33. Resultados entrenamiento EfficientNetV2 euclidiano sobre Imagenette.	42
4.34. Resultados entrenamiento EfficientNetV2 hiperbólico sobre Imagenette.	43
4.35. Resultados entrenamiento EfficientNetV2 híbrido sobre Imagenette.	43
4.36. Comparación entrenamiento EfficientNetV2 sobre Imagenette.	44
4.37. Comparación entrenamiento EfficientNetV2 sobre Imagenette.	44
4.38. Comparación entrenamiento EfficientNetV2 sobre Imagenette para $d = 4$.	45
4.39. Comparación entrenamiento EfficientNetV2 sobre Imagenette para $d = 256$.	45
5.1. Resultados entrenamiento EfficientNetV2 sobre CIFAR100.	57
A.1. Resultados entrenamiento ResNet50 euclidiano sobre CIFAR10.	63
A.2. Resultados entrenamiento ResNet50 hiperbólico sobre CIFAR10.	64
A.3. Resultados entrenamiento ResNet50 híbrido sobre CIFAR10.	64
A.4. Comparación entrenamiento ResNet50 sobre CIFAR10.	65
A.5. Comparación validación ResNet50 sobre CIFAR10.	65
A.6. Comparación entrenamiento ResNet50 sobre CIFAR10 para $d = 4$.	66
A.7. Comparación entrenamiento ResNet50 sobre CIFAR10 para $d = 8$.	66
A.8. Comparación entrenamiento ResNet50 sobre CIFAR10 para $d = 16$.	67
A.9. Comparación entrenamiento ResNet50 sobre CIFAR10 para $d = 32$.	67
A.10. Comparación entrenamiento ResNet50 sobre CIFAR10 para $d = 64$.	68
A.11. Comparación entrenamiento ResNet50 sobre CIFAR10 para $d = 128$.	68
A.12. Comparación entrenamiento ResNet50 sobre CIFAR10 para $d = 256$.	69

A.13. Resultados entrenamiento EfficientNetV2 euclidiano sobre CIFAR10.	70
A.14. Resultados entrenamiento EfficientNetV2 hiperbólico sobre CIFAR10.	70
A.15. Resultados entrenamiento EfficientNetV2 híbrido sobre CIFAR10.	71
A.16. Comparación entrenamiento EfficientNetV2 sobre CIFAR10.	71
A.17. Comparación entrenamiento EfficientNetV2 sobre CIFAR10.	72
A.18. Comparación entrenamiento EfficientNetV2 sobre CIFAR10 para $d = 4$.	72
A.19. Comparación entrenamiento EfficientNetV2 sobre CIFAR10 para $d = 8$.	73
A.20. Comparación entrenamiento EfficientNetV2 sobre CIFAR10 para $d = 16$.	73
A.21. Comparación entrenamiento EfficientNetV2 sobre CIFAR10 para $d = 32$.	74
A.22. Comparación entrenamiento EfficientNetV2 sobre CIFAR10 para $d = 64$.	74
A.23. Comparación entrenamiento EfficientNetV2 sobre CIFAR10 para $d = 128$.	75
A.24. Comparación entrenamiento EfficientNetV2 sobre CIFAR10 para $d = 256$.	75
A.25. Resultados entrenamiento ResNet50 euclidiano sobre CIFAR100.	76
A.26. Resultados entrenamiento ResNet50 hiperbólico sobre CIFAR100.	76
A.27. Resultados entrenamiento ResNet50 híbrido sobre CIFAR100.	77
A.28. Comparación entrenamiento ResNet50 sobre CIFAR100.	77
A.29. Comparación validación ResNet50 sobre CIFAR100.	78
A.30. Comparación entrenamiento ResNet50 sobre CIFAR100 para $d = 4$.	78
A.31. Comparación entrenamiento ResNet50 sobre CIFAR100 para $d = 8$.	79
A.32. Comparación entrenamiento ResNet50 sobre CIFAR100 para $d = 16$.	79
A.33. Comparación entrenamiento ResNet50 sobre CIFAR100 para $d = 32$.	80
A.34. Comparación entrenamiento ResNet50 sobre CIFAR100 para $d = 64$.	80
A.35. Comparación entrenamiento ResNet50 sobre CIFAR100 para $d = 128$.	81
A.36. Comparación entrenamiento ResNet50 sobre CIFAR100 para $d = 256$.	81
A.37. Resultados entrenamiento EfficientNetV2 euclidiano sobre CIFAR100.	82
A.38. Resultados entrenamiento EfficientNetV2 hiperbólico sobre CIFAR100.	82
A.39. Resultados entrenamiento EfficientNetV2 híbrido sobre CIFAR100.	83
A.40. Comparación entrenamiento EfficientNetV2 sobre CIFAR100.	83
A.41. Comparación entrenamiento EfficientNetV2 sobre CIFAR100.	84
A.42. Comparación entrenamiento EfficientNetV2 sobre CIFAR100 para $d = 4$.	84
A.43. Comparación entrenamiento EfficientNetV2 sobre CIFAR100 para $d = 8$.	85
A.44. Comparación entrenamiento EfficientNetV2 sobre CIFAR100 para $d = 16$.	85
A.45. Comparación entrenamiento EfficientNetV2 sobre CIFAR100 para $d = 32$.	86
A.46. Comparación entrenamiento EfficientNetV2 sobre CIFAR100 para $d = 64$.	86
A.47. Comparación entrenamiento EfficientNetV2 sobre CIFAR100 para $d = 128$.	87

A.48.Comparación entrenamiento EfficientNetV2 sobre CIFAR100 para $d = 256$.	87
A.49.Resultados entrenamiento ResNet50 euclidiano sobre Imagenette.	88
A.50.Resultados entrenamiento ResNet50 hiperbólico sobre Imagenette.	88
A.51.Resultados entrenamiento ResNet50 híbrido sobre Imagenette.	89
A.52.Comparación entrenamiento ResNet50 sobre Imagenette.	89
A.53.Comparación validación ResNet50 sobre Imagenette.	90
A.54.Comparación entrenamiento ResNet50 sobre Imagenette para $d = 4$.	90
A.55.Comparación entrenamiento ResNet50 sobre Imagenette para $d = 8$.	91
A.56.Comparación entrenamiento ResNet50 sobre Imagenette para $d = 16$.	91
A.57.Comparación entrenamiento ResNet50 sobre Imagenette para $d = 32$.	92
A.58.Comparación entrenamiento ResNet50 sobre Imagenette para $d = 64$.	92
A.59.Comparación entrenamiento ResNet50 sobre Imagenette para $d = 128$.	93
A.60.Comparación entrenamiento ResNet50 sobre Imagenette para $d = 256$.	93
A.61.Resultados entrenamiento EfficientNetV2 euclidiano sobre Imagenette.	94
A.62.Resultados entrenamiento EfficientNetV2 hiperbólico sobre Imagenette.	94
A.63.Resultados entrenamiento EfficientNetV2 híbrido sobre Imagenette.	95
A.64.Comparación entrenamiento EfficientNetV2 sobre Imagenette.	95
A.65.Comparación entrenamiento EfficientNetV2 sobre Imagenette.	96
A.66.Comparación entrenamiento EfficientNetV2 sobre Imagenette para $d = 4$.	96
A.67.Comparación entrenamiento EfficientNetV2 sobre Imagenette para $d = 8$.	97
A.68.Comparación entrenamiento EfficientNetV2 sobre Imagenette para $d = 16$.	97
A.69.Comparación entrenamiento EfficientNetV2 sobre Imagenette para $d = 32$.	98
A.70.Comparación entrenamiento EfficientNetV2 sobre Imagenette para $d = 64$.	98
A.71.Comparación entrenamiento EfficientNetV2 sobre Imagenette para $d = 128$.	99
A.72.Comparación entrenamiento EfficientNetV2 sobre Imagenette para $d = 256$.	99
B.1. Resultados entrenamiento ResNet50 sobre CIFAR10.	101
B.2. Resultados entrenamiento EfficientNetV2 sobre CIFAR10.	102
B.3. Resultados entrenamiento ResNet50 sobre CIFAR100.	103
B.4. Resultados entrenamiento EfficientNetV2 sobre CIFAR100.	104

Índice de tablas

4.1. Accuracy (%) sobre test para todas las configuraciones de <i>Matrioshka Embeddings</i>	46
4.2. Valores de curvatura aprendida al final del entrenamiento (experimentos sin pesos pre- entrenados).	47
4.3. Accuracy (%) sobre test con pesos preentrenados de ImageNet.	49
5.1. HmAP@10 para modelos euclidiano (E) e hiperbólico (H) entrenados en CIFAR10, a diferentes niveles de granularidad.	53
5.2. HmAP@10 para modelos euclidiano (E) e hiperbólico (H) entrenados en CIFAR100, a diferentes niveles de granularidad.	54
5.3. Variaciones de hiperparámetros evaluadas para el modelo hiperbólico.	56

Capítulo 1

Introducción

Encontrar formas de representar datos es una de las principales tareas en el aprendizaje automático e inteligencia artificial moderna. El aprendizaje de representaciones tiene como objetivo agrupar elementos semánticamente similares, a la vez que se separan aquellos semánticamente diferentes. Una versión simple de esta tarea es clasificar imágenes según diferentes categorías y crear una medida de distancia que pueda utilizarse para estimar la similitud entre diferentes imágenes. Esto se logra mediante *embeddings* [15], que son vectores de dimensionalidad relativamente baja en los que traducimos datos no estructurados. Esto nos permite simplificar datos complejos y de alta dimensión en vectores densos que podemos analizar o comparar de manera factible.

La generación de estos *embeddings* es una tarea desafiante y está influenciada por factores como la arquitectura de la red neuronal, las técnicas de preprocesamiento, la dimensionalidad del espacio de *embeddings*, y las operaciones matemáticas que pueden aplicarse sobre los *embeddings* para soportar aplicaciones como búsqueda por similitud. Este último punto es de especial interés para este trabajo, ya que se ha realizado investigación significativa recientemente sobre las aplicaciones de diferentes geometrías [24, 31] al espacio de *embeddings* (i.e., esférica, hiperbólica).

La geometría hiperbólica es una rama de la geometría no euclidiana que estudia espacios con curvatura constante negativa. En el contexto del aprendizaje automático, el modelo más empleado para representar este tipo de espacios es el modelo de la bola de Poincaré, el cual describe el espacio hiperbólico como la bola unitaria abierta dotada de una métrica Riemanniana específica. En este modelo, la distancia hiperbólica entre dos puntos crece de manera exponencial con respecto a su distancia euclidiana conforme se aproximan al borde de la bola. Esta propiedad geométrica hace posible representar de forma natural estructuras jerárquicas [13, 26], ya que los conceptos más generales pueden ubicarse cerca del centro, mientras que los más específicos se disponen hacia los bordes, evitando así distorsiones significativas en la representación.

Muchos datasets de imágenes exhiben estructuras inherentemente jerárquicas: categorías generales que se subdividen en subcategorías más específicas [7]. El espacio euclidiano presenta limitaciones fundamentales para representar eficientemente estas jerarquías, especialmente cuando la dimensionalidad de los *embeddings* se reduce. El espacio hiperbólico surge como una alternativa natural para acomodar este tipo de estructuras, ya que el volumen crece exponencialmente con el radio, lo que permite representar árboles o jerarquías profundas en pocas dimensiones, mientras que en el espacio euclidiano se requerirían muchas más dimensiones para mantener las mismas relaciones de proximidad [24, 25]. A pesar del éxito de los *embeddings* hiperbólicos en procesamiento de lenguaje natural, su aplicación en tareas de visión por computador permanece relativamente inexplorada. Esta brecha es particularmente crítica en contextos donde la reducción dimensional es necesaria por restricciones computacionales o de memoria.

1.1. Objetivos

1.1.1. General

Investigar y validar el uso de *embeddings* hiperbólicos para representaciones de imágenes, desarrollando metodologías que aprovechen la geometría no euclidiana para capturar estructuras jerárquicas en datos visuales, con aplicaciones específicas en *Matrioshka Embeddings* y *Deep Clustering*.

1.1.2. Específicos

1. Implementar y validar un framework de capas neuronales hiperbólicas que pueda ser integrado con PyTorch para el procesamiento de imágenes.
2. Desarrollar un método de entrenamiento para *Matrioshka Embeddings* en el espacio hiperbólico que permita representaciones de dimensionalidad variable.
3. Explorar la adaptación de algoritmos de Deep Clustering al espacio hiperbólico, evaluando su viabilidad y estabilidad.
4. Evaluar comparativamente el rendimiento de *embeddings* hiperbólicos versus euclidianos en tareas downstream bajo diferentes dimensionalidades.

1.2. Estructura de la tesis

La presente tesis se encuentra estructurada de la siguiente forma. En el Capítulo 2 se introduce brevemente el estado del arte y se discute la base teórica para las herramientas hiperbólicas a utilizar.

En el Capítulo 3 se presentan las hipótesis de la tesis, junto con el ambiente de trabajo y los conjuntos de datos a utilizar. Los experimentos son presentados en los Capítulos 4 y 5, en donde se detallan las implementaciones y los resultados obtenidos. Finalmente, en el Capítulo 6 se discuten los resultados y el trabajo futuro.

Capítulo 2

Estado del Arte

2.1. Aprendizaje de Representaciones Visuales

El aprendizaje de representaciones (*representation learning*) busca transformar datos de alta dimensionalidad en vectores compactos (*embeddings*) que preserven la estructura semántica de los datos originales. El trabajo de [15] introdujo la *contrastive loss*, sentando las bases para que pares de ejemplos similares queden cercanos en el espacio de *embeddings*, mientras que pares disimilares se mantengan alejados.

Con la llegada de las redes convolucionales profundas, se demostró que las representaciones aprendidas de forma supervisada en ImageNet [7] y otros datasets masivos podían ser transferidas exitosamente a tareas de visión *downstream* [33].

2.2. Geometrías No Euclidianas

El espacio euclidiano es la elección implícita de prácticamente toda arquitectura de red neuronal clásica. Sin embargo, diversas estructuras de datos presentan propiedades topológicas que no se acomodan naturalmente a esta geometría plana. Durante los últimos años, se han explorado alternativas con curvatura constante, como el espacio esférico (curvatura positiva) y el espacio hiperbólico (curvatura negativa), cada uno con ventajas específicas según la naturaleza de los datos. Este trabajo se centra en el espacio hiperbólico debido a su capacidad para representar estructuras jerárquicas.

2.2.1. Espacio Hiperbólico

La geometría hiperbólica es una geometría no euclidiana que estudia espacios de curvatura negativa constante. El espacio hiperbólico es particularmente adecuado para modelar datos jerárquicos, ya

que las jerarquías pueden representarse como árboles taxonómicos donde la clase raíz contiene todos los elementos y las clases secundarias se extienden hacia los nodos hoja. El espacio hiperbólico puede integrar de forma natural estas estructuras con una distorsión arbitrariamente baja [13, 26], gracias a que su volumen se expande exponencialmente, de manera análoga al crecimiento exponencial del número de hojas de un árbol conforme aumenta la profundidad [24].

Existen múltiples modelos isométricos del espacio hiperbólico: el modelo del hiperboloide (o de Minkowski), el modelo del disco de Klein y el modelo de la bola de Poincaré. Este último es el más empleado en aprendizaje automático por su intuición geométrica y su capacidad para definir operaciones diferenciables [11].

En el modelo de Poincaré, el espacio n -dimensional queda representado como la bola unitaria abierta $\mathbb{D}_c^n = \{x \in \mathbb{R}^n : c\|x\|^2 < 1, c \geq 0\}$ dotada de la métrica Riemanniana:

$$g_x^{\mathbb{D}} = \left(\frac{2}{1 - \|x\|^2} \right)^2 g^E,$$

donde g^E es la métrica euclidiana estándar.

2.3. Redes Neuronales Hiperbólicas

El trabajo de [24] demostró el potencial del espacio hiperbólico para el aprendizaje de representaciones jerárquicas. Utilizando el disco de Poincaré, representaron grafos de WordNet como *embeddings*, obteniendo representaciones que capturaban relaciones taxonómicas con gran precisión en bajas dimensiones, superando a su equivalente euclidiano.

A partir de este trabajo, [11] generalizaron las operaciones de las redes neuronales al espacio hiperbólico mediante el formalismo de *giroespacios*, definiendo capas lineales y clasificadores hiperbólicos, así como funciones de activación que respetan la geometría. Esta contribución es fundamental, ya que hace posible construir redes profundas completamente hiperbólicas. Posteriormente, [27] extendieron este marco mediante la optimización de la cantidad de parámetros y la generalización de operaciones para separar o concatenar vectores.

En el contexto de grafos, [6] propusieron HGCN (*Hyperbolic Graph Convolutional Networks*), integrando la convolución en grafos dentro del modelo hiperboloide del espacio hiperbólico para capturar mejor las relaciones jerárquicas presentes en los datos.

2.4. Embeddings Hiperbólicos en Tareas de Visión

La aplicación de la geometría hiperbólica a datos visuales es un área de investigación más reciente y menos saturada que su contraparte en NLP. [18] fueron de los primeros en explorar sistemáticamente *Hyperbolic Image Embeddings*, mostrando que las representaciones hiperbólicas superan a las euclidianas en tareas de recuperación de imágenes y clasificación *few-shot*, especialmente en conjuntos de datos con estructura taxonómica como CUB y CIFAR100.

[9] integraron *Vision Transformers* con *embeddings* hiperbólicos en *Hyperbolic Vision Transformers*, estableciendo nuevos resultados de referencia en aprendizaje métrico para visión. Sus experimentos muestran que los *embeddings* hiperbólicos mantienen un rendimiento superior o comparable al euclidiano en bajas dimensiones, lo que sugiere que la geometría hiperbólica permite representar estructuras jerárquicas de manera más eficiente cuando la capacidad del *embedding* es limitada.

[8] extendieron el uso de *embeddings* hiperbólicos al aprendizaje multimodal con MERU, un modelo contrastivo que aprende representaciones conjuntas de imágenes y texto en un espacio hiperbólico. Aprovechando las propiedades geométricas de los espacios con curvatura negativa para capturar estructuras jerárquicas en conceptos visuales y lingüísticos, MERU aprende un espacio de representación más interpretable donde los conceptos más generales tienden a situarse cerca del origen y los más específicos hacia la periferia. En comparación con modelos como CLIP, sus resultados son competitivos en tareas de clasificación y recuperación.

En nuestro trabajo previo [25], investigamos la capacidad de los *embeddings* hiperbólicos para recuperar relaciones jerárquicas latentes en conjuntos de datos de imágenes sin acceso explícito a la jerarquía durante el entrenamiento. Utilizando una arquitectura VGG16 con capas hiperbólicas en las últimas etapas, demostramos que los *embeddings* de Poincaré pueden descubrir relaciones clase-subclase de manera no supervisada. Para evaluar la calidad de las representaciones, propusimos HmAP (*Hierarchy-Aware mean Average Precision*), una extensión de mAP que considera la estructura jerárquica al medir la relevancia de los elementos recuperados. Los resultados experimentales en CIFAR10 y CIFAR100 mostraron que los *embeddings* hiperbólicos superan a sus contrapartes euclidianas en tareas de recuperación de imágenes, con ventajas especialmente pronunciadas en datasets con mayor número de clases y cuando la dimensionalidad del *embedding* es limitada. Este trabajo sienta las bases para la presente tesis, extendiendo la aplicación de geometría hiperbólica a *Matrioshka Embeddings* y *Deep Clustering*.

2.5. Matrioshka Representation Learning

Un problema práctico en el despliegue de sistemas de recuperación de imágenes a gran escala es la necesidad de adaptar la dimensionalidad de los *embeddings* a las restricciones de cómputo y memoria

disponibles en cada contexto. La solución tradicional consiste en entrenar modelos separados para cada dimensionalidad objetivo, lo cual es costoso e impráctico. [21] propusieron *Matryoshka Representation Learning* (MRL) como una solución a este problema.

La idea central de MRL es entrenar un único modelo que produzca *embeddings* con representaciones anidadas: los primeros d componentes del *embedding* de dimensión D deben ser, por sí solos, un *embedding* de calidad razonable para la tarea, para cualquier $d \leq D$. Esto se logra optimizando simultáneamente la función de pérdida en múltiples dimensiones: $\{m_1, m_2, \dots, m_L\}$ con $m_1 < m_2 < \dots < m_L = D$:

$$\mathcal{L}_{\text{MRL}} = \sum_{l=1}^L c_l \cdot \mathcal{L} \left(W^{(m_l)} \cdot z_{1:m_l}, y \right),$$

donde z es el *embedding* completo y $W^{(m_l)}$ es un clasificador lineal específico para la dimensionalidad m_l .

Los experimentos de [21] sobre ImageNet-1k con modelos ResNet y ViT demostraron que los *embeddings* MRL de dimensión d obtienen un rendimiento comparable al de modelos entrenados específicamente para esa dimensionalidad d , ofreciendo flexibilidad adaptativa sin sacrificar calidad.

2.6. Deep Clustering

El *Deep Clustering* integra el aprendizaje de representaciones con el agrupamiento no supervisado en un marco unificado.

[30] propusieron DEC (*Deep Embedded Clustering*), un método para aprender de forma conjunta un espacio de representación latente y las asignaciones de *clusters* mediante redes neuronales profundas. En DEC se preentrena un *autoencoder* para obtener una representación latente inicial, y luego se refina iterativamente esa representación y las asignaciones mediante una distribución tipo t -Student y la minimización de la divergencia de Kullback-Leibler respecto a una distribución objetivo auxiliar. Este trabajo estableció el paradigma de *clustering* profundo en el espacio latente, influyendo en numerosos métodos posteriores. [14] extendieron DEC con DCEC (*Deep Convolutional Embedded Clustering*), incorporando módulos convolucionales para datos de imagen y un término de reconstrucción que preserva la estructura local del espacio latente.

[3] propusieron *DeepCluster*, un enfoque que alterna entre la asignación de pseudo-etiquetas mediante k -means sobre las características extraídas por una red y el entrenamiento de dicha red para predecir estas pseudo-etiquetas. Este esquema iterativo demostró que es posible aprender características visuales útiles en un contexto no supervisado y que las representaciones y el *clustering* pueden mejorar mutuamente. *DeepCluster* alcanzó rendimientos competitivos para aprendizaje no supervisado de características, influyendo en la investigación posterior sobre métodos alternados y auto-supervisados

basados en pseudo-etiquetas.

Trabajos posteriores a [3] identificaron y arreglaron limitaciones del método. [4] propusieron SwAV, que reemplaza el paso de k -means con asignaciones a prototipos aprendidos, que obliga a que las asignaciones sean balanceadas mediante el algoritmo de Sinkhorn-Knopp. Esto elimina el ciclo de retroalimentación negativa de *DeepCluster*, donde pseudo-etiquetas de baja calidad producen *embeddings* poco discriminativos y viceversa, y añade una regularización implícita sobre la distribución de asignaciones, evitando el colapso a un único cluster. [5] propusieron DINO, que abandona el clustering por completo: en su lugar, emplea una arquitectura maestro-alumno donde el maestro se actualiza como un promedio exponencial del alumno, sin requerir pseudo-etiquetas ni ninguna forma explícita de *clustering*. Aplicado sobre *Vision Transformers*, DINO muestra que los *tokens* de clase aprendidos exhiben propiedades de segmentación semántica emergentes, sugiriendo que la estructura del espacio de representación captura relaciones jerárquicas sin supervisión directa. Ambos enfoques representan alternativas más estables a *DeepCluster* que habrían merecido exploración en el contexto hiperbólico.

Investigaciones como la de [23], que exploró el *clustering* jerárquico en espacios hiperbólicos mediante representaciones continuas de árboles, sugieren que la geometría hiperbólica puede mejorar la calidad de los *clusters* en datos con relaciones taxonómicas. No obstante, la integración de *clustering* hiperbólico con redes neuronales profundas permanece como una brecha significativa en la literatura.

2.7. Arquitecturas Base

A continuación, se describen las arquitecturas de redes neuronales convolucionales utilizadas como extractores de características en los experimentos de esta tesis.

2.7.1. ResNet

ResNet [16] corresponde a una familia de redes neuronales convolucionales que emplean bloques residuales para abordar el desafío del entrenamiento de redes profundas. A diferencia de una red neuronal clásica, en la que la entrada es transformada por capas convolucionales y luego por la función de activación, en un bloque residual, la entrada se suma a la salida de las capas convolucionales, creando un “atajo” que permite que el gradiente fluya de mejor manera y evita el problema del gradiente desvaneciente.

La salida de un bloque residual puede denotarse por:

$$H(x) = F(x) + x \quad (2.1)$$

La arquitectura de un bloque residual puede verse en la Figura 2.1

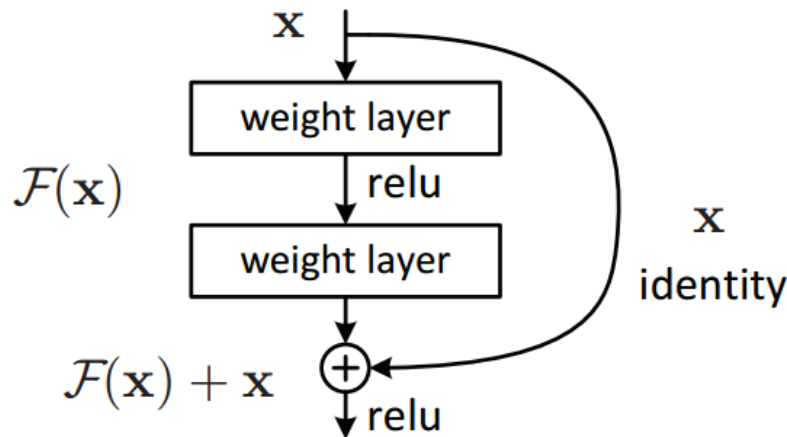


Figura 2.1: Bloque residual ResNet.

Una red ResNet consiste en la composición de múltiples bloques residuales, pudiendo identificar modelos como ResNet18, ResNet34, ResNet50 o ResNet101 según la cantidad de capas.

2.7.2. EfficientNetV2

EfficientNetV2 [29] es una familia de redes convolucionales diseñadas para lograr una alta precisión con una eficiencia computacional mejorada mediante la combinación de escalado de modelos y optimización de diseño. A diferencia de las redes tradicionales que aumentan la cantidad de capas o parámetros para mejorar el rendimiento, EfficientNetV2 utiliza escalado compuesto, equilibrando la profundidad (cantidad de capas), el ancho (cantidad de canales por capa) y la resolución (tamaño de la imagen) para optimizar la red para tareas específicas.

Una de sus principales contribuciones es la introducción de los bloques Fused-MBConv, que reemplazan las convoluciones separables en profundidad en las primeras etapas por una convolución regular 3×3 fusionada con la expansión, mejorando la eficiencia del entrenamiento. En etapas posteriores, la red emplea bloques residuales invertidos de estilo MobileNetV2. Esta combinación permite procesar conjuntos de datos a gran escala con menores requisitos computacionales y de memoria.

El diseño de EfficientNetV2 pone énfasis en el entrenamiento progresivo, que comienza con resoluciones de imagen más pequeñas y las aumenta gradualmente durante el entrenamiento, mejorando aún más tanto la precisión como la eficiencia.

La familia incluye los modelos EfficientNetV2-S, EfficientNetV2-M y EfficientNetV2-L, que reflejan diferentes configuraciones de tamaño y complejidad computacional.

Capítulo 3

Metodología

3.1. Hipótesis

El uso de *embeddings* hiperbólicos en lugar de euclidianos mejora el rendimiento y/o tiempo de cómputo de aplicaciones de visión por computador con datos semánticamente complejos, debido a la capacidad natural del espacio hiperbólico para representar eficientemente estructuras jerárquicas.

3.1.1. Hipótesis Específicas

1. En el contexto de *Matrioshka Embeddings*, las representaciones hiperbólicas preservan una mayor cantidad de información útil para tareas *downstream* al reducir su dimensionalidad (i.e., al trunca el *embedding*), en comparación con sus equivalentes euclidianas.
2. Los métodos de *Deep Clustering* que operan en el espacio hiperbólico producirán agrupaciones que respetan mejor las jerarquías semánticas naturales de los datos visuales, lo cual será evaluado mediante el estudio y aplicación de métricas diseñadas para capturar la estructura jerárquica.
3. La ventaja relativa de *embeddings* hiperbólicos sobre los euclidianos será más pronunciada en datasets con mayor estructura jerárquica (e.g., CIFAR100 vs CIFAR10) y cuando la dimensión del *embedding* es especialmente limitada.

3.2. Materiales

3.2.1. Stack Tecnológico

Para el desarrollo de los experimentos, se utiliza Python con la biblioteca de aprendizaje automático PyTorch. Las versiones hiperbólicas de las capas para redes neuronales serán provistas por HypTorch, una biblioteca desarrollada durante la implementación de esta tesis, descrita en la Sección [3.3](#).

3.2.2. Ambiente de Desarrollo

A continuación, se presentan los detalles del *hardware* y *software* utilizados durante el desarrollo del sistema.

Hardware

- 16 GB de memoria RAM
- 200 GB de almacenamiento de disco duro (HDD)
- GPU NVIDIA GeForce RTX 4060 8GB

Software

- Python 3.13
 - PyTorch
 - Lightning
 - FAISS
 - Matplotlib

3.3. HypTorch

HypTorch es una biblioteca desarrollada como parte de esta tesis que provee capas neuronales para trabajar con geometría hiperbólica en PyTorch. El código se encuentra disponible en [GitHub](#). A continuación se describen sus componentes principales.

3.3.1. Capa ToPoincaré

Esta capa, sin parámetros entrenables, transforma un *embedding* euclidiano en uno hiperbólico. Sea $v \in \mathbb{R}^n$ una representación de datos extraída de una capa de red neuronal. La capa ToPoincaré aplica

la función exponencial [18] evaluada en el origen, definida como la función biyectiva $\exp_0^c : T_0\mathbb{D}_c^n = \mathbb{R}^n \rightarrow \mathbb{D}_c^n$:

$$\exp_0^c(v) = \tanh(\sqrt{c}\|v\|) \frac{v}{\sqrt{c}\|v\|} \quad (3.1)$$

3.3.2. Capa FromPoincaré

Esta capa, sin parámetros entrenables, transforma un *embedding* hiperbólico en uno euclidiano. Sea $\mathbf{y} \in \mathbb{D}_c^n$ un punto en el disco de Poincaré. La capa FromPoincaré aplica el mapa logarítmico en el origen, definido como la función inversa del mapa exponencial:

$$\log_0^c(\mathbf{y}) = \frac{2}{\sqrt{c}} \tanh^{-1}(\sqrt{c}\|\mathbf{y}\|) \frac{\mathbf{y}}{\|\mathbf{y}\|} \quad (3.2)$$

Esta capa es útil para arquitecturas híbridas donde se requiere extraer representaciones del espacio hiperbólico para procesarlas con capas euclidianas convencionales, o para aplicar algoritmos que operan en el espacio euclidiano (como k -means) sobre *embeddings* hiperbólicos.

3.3.3. Capa Lineal Hiperbólica (HypLinear)

Esta capa implementa la transformación lineal en el espacio hiperbólico, análoga a la capa `nn.Linear` de PyTorch. Utiliza la multiplicación matriz-vector de Möbius y la adición de Möbius para los *bias*.

Dada una matriz de pesos $\mathbf{M} \in \mathbb{R}^{m \times n}$ y un vector de entrada $\mathbf{x} \in \mathbb{D}_c^n$, la multiplicación matriz-vector de Möbius se define como [11]:

$$\mathbf{M} \otimes_c \mathbf{x} = \frac{1}{\sqrt{c}} \tanh\left(\frac{\|\mathbf{M}\mathbf{x}\|}{\|\mathbf{x}\|} \tanh^{-1}(\sqrt{c}\|\mathbf{x}\|)\right) \frac{\mathbf{M}\mathbf{x}}{\|\mathbf{M}\mathbf{x}\|} \quad (3.3)$$

Si se utiliza un sesgo $\mathbf{b} \in \mathbb{R}^m$, este se proyecta primero al espacio hiperbólico mediante el mapa exponencial y luego se aplica la adición de Möbius:

Esta capa permite construir redes neuronales profundas que operan completamente en el espacio hiperbólico, preservando la geometría en cada transformación.

3.3.4. Capa de Regresión Logística Multiclase Hiperbólica (HMLR)

Esta capa reformula la regresión logística multiclase euclidiana (MLR) desde la perspectiva de las distancias a hiperplanos separadores, como en [22]. Dadas K clases, la MLR ajusta un hiperplano separador para cada una de estas clases, modelando las probabilidades a posteriori como:

$$p(y = k \mid \mathbf{x}) \propto \exp((\langle \mathbf{a}_k, \mathbf{x} \rangle - b_k)), \text{ donde } b_k \in \mathbb{R}; \mathbf{x}, \mathbf{a}_k \in \mathbb{R}^n, \forall k \in \{1, \dots, K\}.$$

Gracias al Teorema 5 de [11], se formula la versión hiperbólica. Dados $p_k \in \mathbb{D}_c^n$ y $a_k \in T_{p_k} \mathbb{D}_c^n \setminus \{0\}$:

$$p(y = k \mid x) \propto \exp \left(\frac{\lambda_{p_k}^c \|a_k\|}{\sqrt{c}} \sinh^{-1} \left(\frac{2\sqrt{c} \langle -p_k \oplus_c x, a_k \rangle}{(1 - c \| -p_k \oplus_c x \|^2) \|a_k\|} \right) \right) \quad \forall x \in \mathbb{D}_c^n, \quad (3.4)$$

donde la adición de Möbius se define como

$$x \oplus_c y := \frac{(1 + 2c \langle x, y \rangle + c \|y\|^2)x + (1 - c \|x\|^2)y}{1 + 2c \langle x, y \rangle + c^2 \|x\|^2 \|y\|^2}. \quad (3.5)$$

Los parámetros a_k se optimizan mediante la reparametrización $a_k = (\lambda_0^c / \lambda_{p_k}^c) a'_k$ donde $a'_k \in T_0 \mathbb{D}_c^n = \mathbb{R}^n$, permitiendo optimizar a'_k como un parámetro euclidiano [11]. Los parámetros p_k , que residen en el espacio hiperbólico, requieren gradientes riemannianos [10]: $\nabla_{p_k}^R \mathcal{L} = (\lambda_{p_k}^c)^{-2} \nabla_{p_k} \mathcal{L}$, y se actualizan a lo largo de la geodésica correspondiente mediante $\exp_0^c$ [2].

3.3.5. Distancia Hiperbólica

Para $x, y \in \mathbb{D}_c^n$, la función de distancia inducida se define como:

$$d_c(x, y) = \frac{2}{\sqrt{c}} \tanh^{-1} (\sqrt{c} \|(-x) \oplus_c y\|). \quad (3.6)$$

Observamos que $\lim_{c \rightarrow 0} d_c(x, y) = 2\|x - y\|$, es decir, para $c \rightarrow 0$ se recupera la geometría euclidiana, y para $c = 1$ se obtiene la distancia geodésica en el espacio hiperbólico.

3.4. Datasets

3.4.1. CIFAR10

El primer dataset utilizado en los experimentos es CIFAR10 [20], disponible públicamente. Consiste en imágenes a color de baja resolución (32×32 píxeles), organizadas en 10 clases mutuamente excluyentes que presentan una jerarquía de pocos niveles: animales (pájaro, gato, ciervo, perro, rana y caballo) y medios de transporte (avión, automóvil, barco y camión). Algunos ejemplos representativos pueden observarse en la Figura 3.1

El dataset cuenta con un total de 60.000 imágenes distribuidas de forma balanceada (6.000 por clase), divididas en un conjunto de entrenamiento de 50.000 imágenes y uno de prueba de 10.000 imágenes. Esta jerarquía se extiende de forma más granular en CIFAR100, donde cada superclase agrupa subclases más específicas.

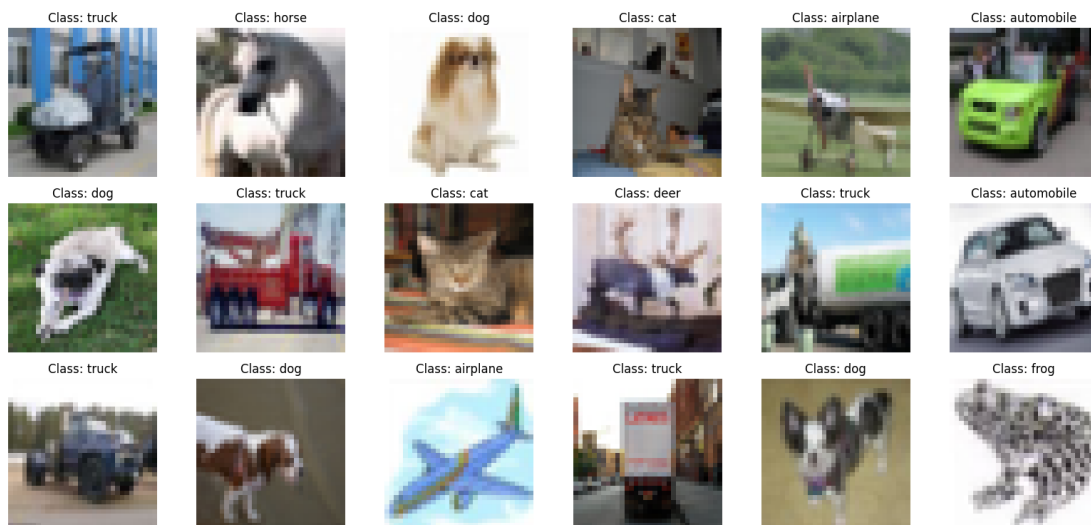


Figura 3.1: Ejemplo de clases CIFAR10.



Figura 3.2: Jerarquía de CIFAR10

3.4.2. CIFAR100

El segundo dataset es CIFAR100 [20], que comparte las características técnicas de CIFAR10: 60.000 imágenes en color de 32×32 píxeles, con la misma división de entrenamiento y prueba. La diferencia fundamental radica en su granularidad: el dataset está organizado en 100 clases, agrupadas en 20 superclases que cubren categorías como animales, plantas, objetos, paisajes, entre otras. Esto genera una estructura jerárquica de muchos niveles, visible en la Figura 3.4.

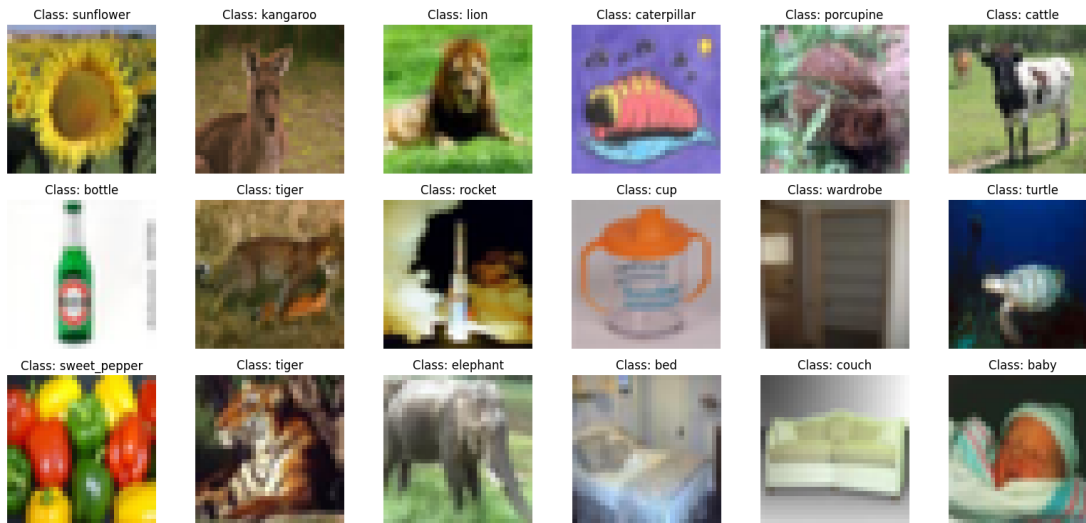


Figura 3.3: Ejemplo de clases CIFAR100.

3.4.3. Imagenette

El tercer dataset es Imagenette, un subconjunto de ImageNet desarrollado con el objetivo de proveer un benchmark más accesible para la investigación en visión por computador. A diferencia de CIFAR10 y CIFAR100, Imagenette contiene imágenes en resolución completa y variable, introduciendo desafíos adicionales propios de escenarios más realistas. El dataset está organizado en 10 clases fácilmente distinguibles entre sí: tenca, springer inglés, reproductor de cassette, motosierra, iglesia, corno francés, camión de basura, bomba de bencina, pelota de golf y paracaídas. Algunos ejemplos representativos pueden observarse en la Figura 3.5.

A diferencia de los datasets CIFAR, Imagenette no presenta una jerarquía explícita entre sus clases, siendo estas independientes entre sí.

Capítulo 4

Hyperbolic Matrioshka Embeddings

El objetivo central de los experimentos presentados en este capítulo es evaluar la viabilidad y el beneficio de incorporar geometría hiperbólica en el marco de *Matrioshka Representation Learning*.

Para ello, se propone reemplazar las capas lineales del espacio de *embedding* por capas lineales hiperbólicas, que operan sobre el disco de Poincaré, siguiendo las formulaciones introducidas en *Hyperbolic Neural Networks* [11] y extendidas al dominio visual en *Hyperbolic Image Embeddings* [18]. La motivación de esta propuesta se presenta en dos argumentos complementarios.

- **Jerarquía implícita en las representaciones Matrioshka:** Una propiedad fundamental de MRL es que las primeras dimensiones del *embedding* están optimizadas para capturar la información más general y discriminativa de la entrada, mientras que las dimensiones subsiguientes incorporan información progresivamente más específica. Esta estructura induce de manera natural una jerarquía en el espacio de representación: las dimensiones iniciales actúan como una descripción de alto nivel, similar a las categorías más generales en una jerarquía, y las posteriores refinan dicha descripción hacia niveles más específicos. El espacio hiperbólico ha sido ampliamente estudiado como geometría natural para la representación de estructuras jerárquicas, dado que su curvatura negativa constante permite embeder árboles con distorsión arbitrariamente baja. En este sentido, existe una conexión geométrica entre la organización de la información en MRL y las propiedades métricas del espacio hiperbólico.
- **Eficiencia geométrica en baja dimensión:** Una de las ventajas prácticas más relevantes de MRL es la posibilidad de utilizar *sub-embeddings* de baja dimensionalidad sin una degradación severa del rendimiento. Sin embargo, el espacio euclidiano presenta limitaciones conocidas en este régimen: el fenómeno de *crowding* implica que, a medida que la dimensión disminuye, el volumen disponible resulta insuficiente para representar estructuras complejas sin incurrir en distorsión sig-

nificativa. El espacio hiperbólico mitiga este problema de manera natural: dado que su volumen crece exponencialmente con la distancia al origen, a diferencia del crecimiento polinomial del espacio euclidiano, es capaz de alojar una cantidad mucho mayor de información semántica en una dimensión equivalente. Esta propiedad sugiere que los *sub-embeddings* de menor dimensión dentro de las representaciones Matrioshka podrían beneficiarse especialmente de una geometría hiperbólica, donde la capacidad representacional no se ve tan severamente comprometida.

Ambos argumentos convergen en la misma hipótesis: la sustitución del espacio euclidiano por el disco de Poincaré en el entrenamiento de representaciones Matrioshka debería producir *embeddings* de mayor calidad en los niveles de baja dimensionalidad, preservando o mejorando el rendimiento en dimensiones superiores. Para contrastar esta hipótesis, se diseñaron experimentos sobre los conjuntos de datos CIFAR10, CIFAR100 e Imagenette.

4.1. Arquitectura

Para los experimentos se definen tres modelos: euclidiano, hiperbólico e híbrido. Los tres comparten la misma arquitectura base, con la diferencia de que el hiperbólico y el híbrido incorporan versiones hiperbólicas de ciertas capas. Para el modelo hiperbólico, primero se pasa el *embedding* mediante la capa ToPoincare. La capa densa corresponde a su versión hiperbólica (HypLinear) y utiliza clasificadores hiperbólicos (HyperbolicMLR). La versión híbrida emplea una capa densa euclidiana con un clasificador hiperbólico. Las Figuras 4.1a, 4.1b y 4.1c ilustran la arquitectura de cada modelo.

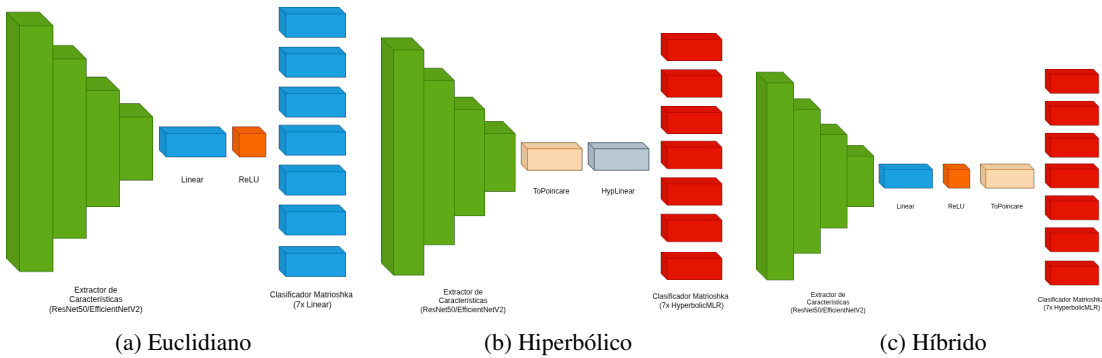


Figura 4.1: Arquitecturas de los tres modelos evaluados.

Como extractores de características se utilizan ResNet50 y EfficientNetV2-S. Las características extraídas se pasan por una capa densa de 256 neuronas, seguida de una activación ReLU en los modelos euclidiano e híbrido. El modelo hiperbólico omite esta activación, ya que opera directamente sobre el espacio de Poincaré. Finalmente, el *embedding* resultante se ingresa al clasificador Matrioshka, ilustrado

en la Figura 4.2. La función de pérdida del modelo corresponde a la suma de las funciones de pérdida (CrossEntropy) evaluadas en cada dimensión.

Desde el punto de vista del costo computacional, la incorporación de geometría hiperbólica no introduce una carga extra significativa sobre el entrenamiento. HypLinear y HyperbolicMLR tienen exactamente la misma cantidad de parámetros que sus equivalentes euclidianos de las mismas dimensiones; el único parámetro adicional de los modelos hiperbólico e híbrido es la curvatura c . Dado que el tiempo de entrenamiento está dominado por el extractor de características, cuya arquitectura es idéntica en los tres modelos, y que la diferencia de parámetros en las capas de *embedding* y clasificación es de un único escalar, el costo computacional por época es esencialmente el mismo para los tres modelos, haciendo que la adopción de geometría hiperbólica sea prácticamente gratuita en términos de tiempo de entrenamiento.

En estos experimentos, el extractor de características se entrena desde cero (sin pesos preentrenados de ImageNet) para realizar una comparación justa entre las arquitecturas. En la Sección 4.7 se presenta un análisis exploratorio que considera el uso de pesos preentrenados.

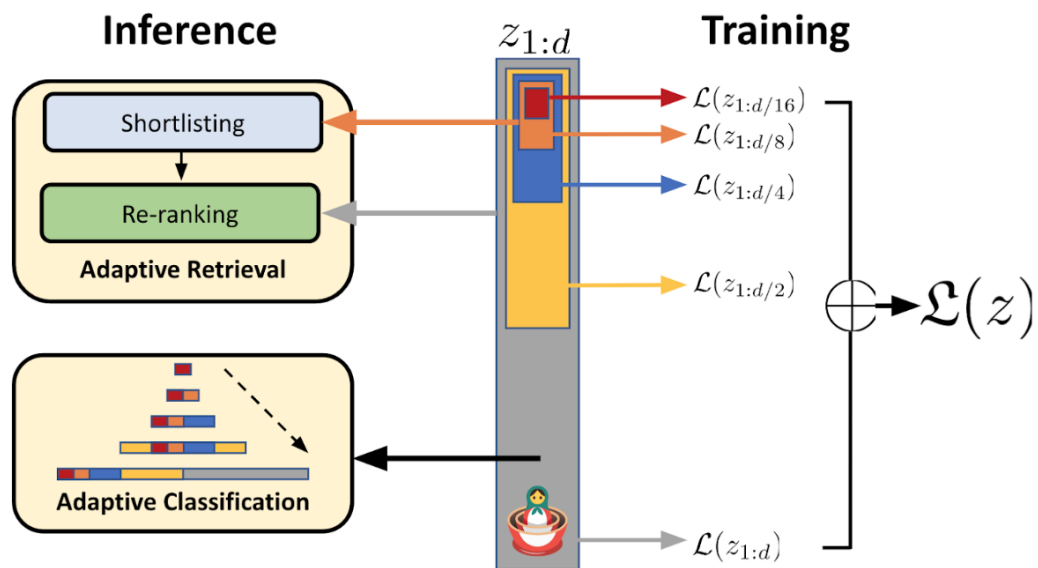


Figura 4.2: Método *Matrioshka Embeddings*.

4.2. Entrenamiento

Cada modelo se entrena por un máximo de 100 épocas con un batch size de 64, utilizando el optimizador AdamW con un *learning rate* inicial de 0.001 y weight decay de 0.0001. Se emplea un scheduler ReduceLROnPlateau que reduce el *learning rate* cuando la pérdida de validación no mejora durante 5 épocas. Adicionalmente, se implementa detención temprana (*early stopping*) con una paciencia de 10 épocas, evaluando la función de pérdida sobre el conjunto de validación.

Como única técnica de aumentación de datos se utiliza `RandomHorizontalFlip`. Para el método Matrioshka se consideran las dimensiones $\{4, 8, 16, 32, 64, 128, 256\}$. En los modelos hiperbólicos, la curvatura inicial es $c = 0.05$ y se permite que el modelo la optimice durante el entrenamiento. Este valor difiere del empleado en el trabajo previo [25] ($c = 0.0005$), que fue determinado mediante *grid search* con curvatura fija. Dado que en estos experimentos la curvatura actúa como parámetro entrenable, su valor inicial cumple únicamente el rol de punto de partida para la optimización, por lo que su elección precisa es menos crítica que en el caso de curvatura fija.

4.3. Evaluación

Durante el entrenamiento se monitorean la pérdida y *accuracy* sobre los conjuntos de entrenamiento y validación, tanto de forma global como por cada dimensión del *embedding*. La selección del mejor modelo se realiza según la pérdida de validación. Los resultados finales se reportan sobre el conjunto de test, evaluando la *accuracy* global y por dimensión.

4.4. Resultados

A continuación, se presentan los resultados obtenidos para cada combinación de dataset y arquitectura. Para cada experimento, se incluyen las curvas de entrenamiento de los modelos, comparaciones de pérdida entre ellos y un análisis detallado por dimensión del *embedding*. En esta sección se muestran únicamente las figuras más relevantes para el análisis, mientras que la totalidad de estas se encuentra disponible en el Anexo A.

4.4.1. CIFAR10 - ResNet50

Las Figuras 4.3, 4.4 y 4.5 presentan los resultados de entrenamiento por dimensión para cada modelo.

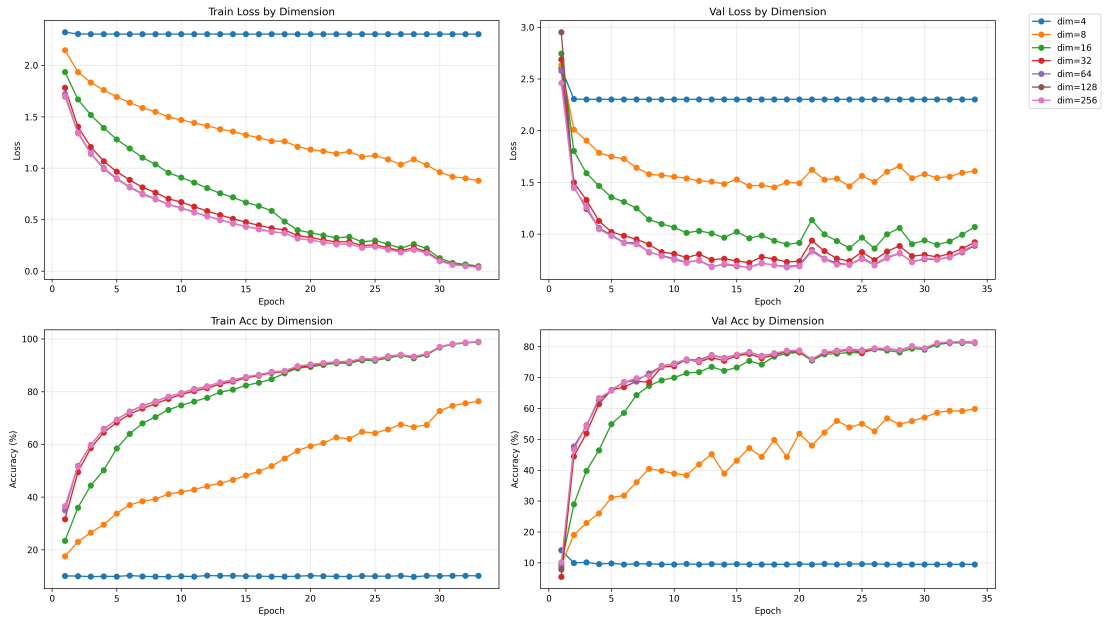


Figura 4.3: Resultados entrenamiento ResNet50 euclidiano sobre CIFAR10.

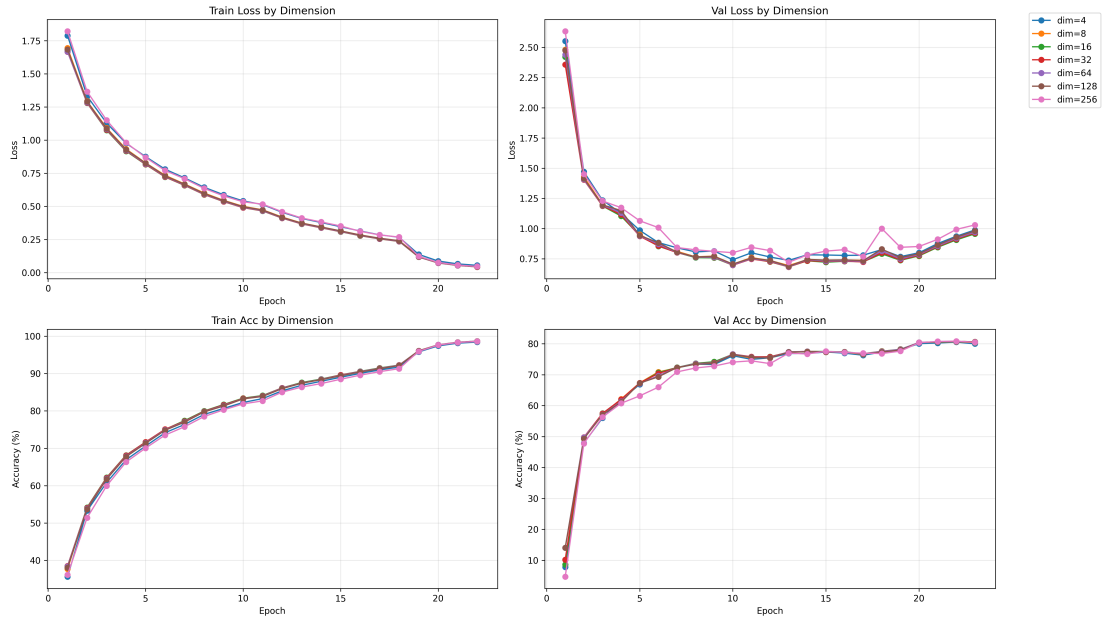


Figura 4.4: Resultados entrenamiento ResNet50 hiperbólico sobre CIFAR10.

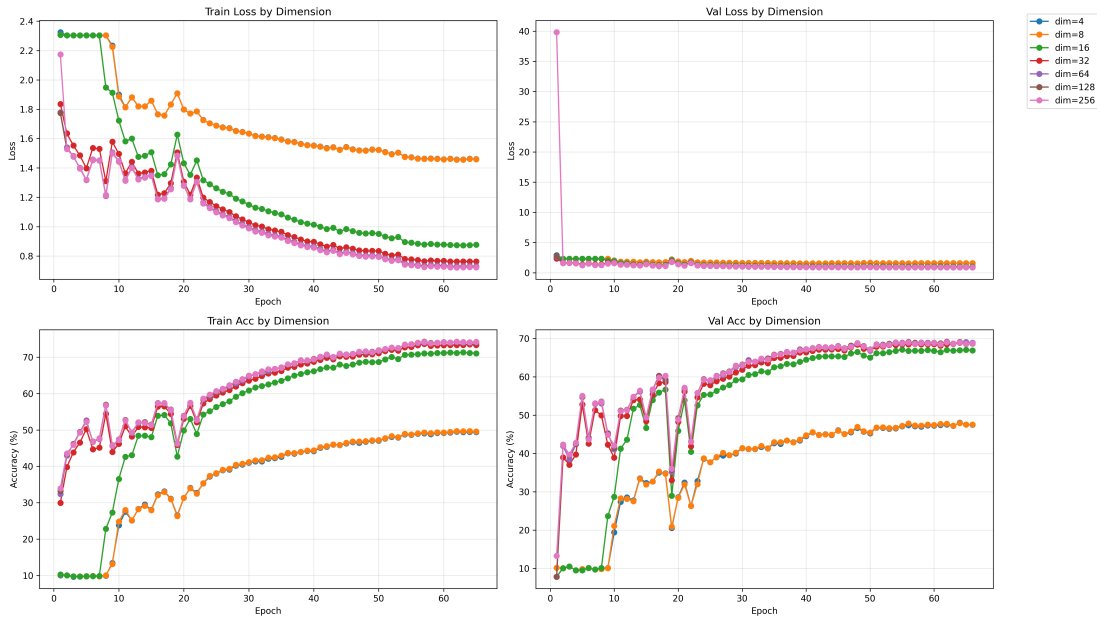
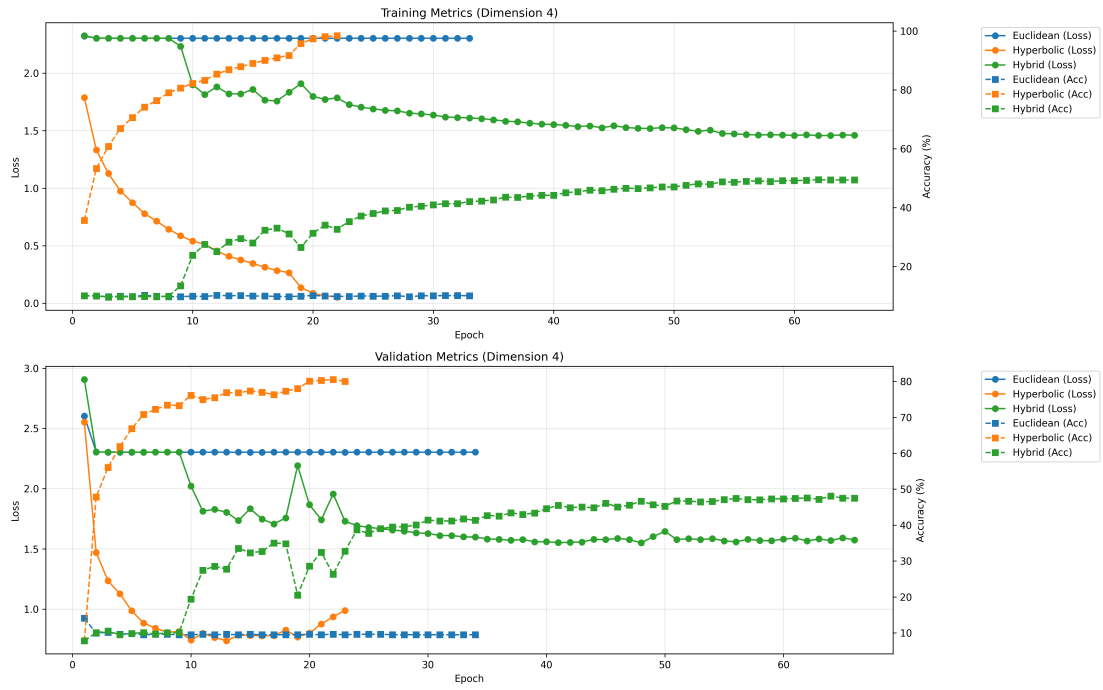
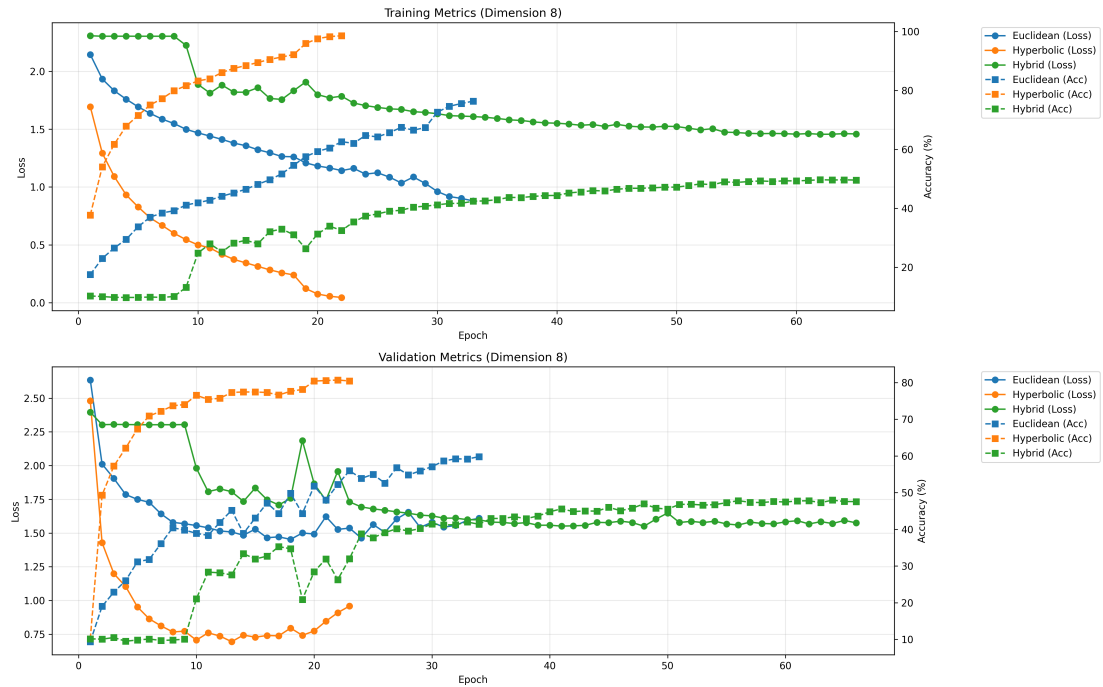


Figura 4.5: Resultados entrenamiento ResNet50 híbrido sobre CIFAR10.

Resulta notable que los modelos euclidiano e híbrido presentan dificultades en baja dimensionalidad, llegando al caso extremo en que el modelo euclidiano no logra aprender para $d = 4$, situación que no ocurre con el modelo hiperbólico.

Las Figuras 4.6 a 4.8 presentan una comparación detallada de los tres modelos para las dimensiones $d = 4, 8, 16$.

Figura 4.6: Comparación entrenamiento ResNet50 sobre CIFAR10 para $d = 4$.Figura 4.7: Comparación entrenamiento ResNet50 sobre CIFAR10 para $d = 8$.

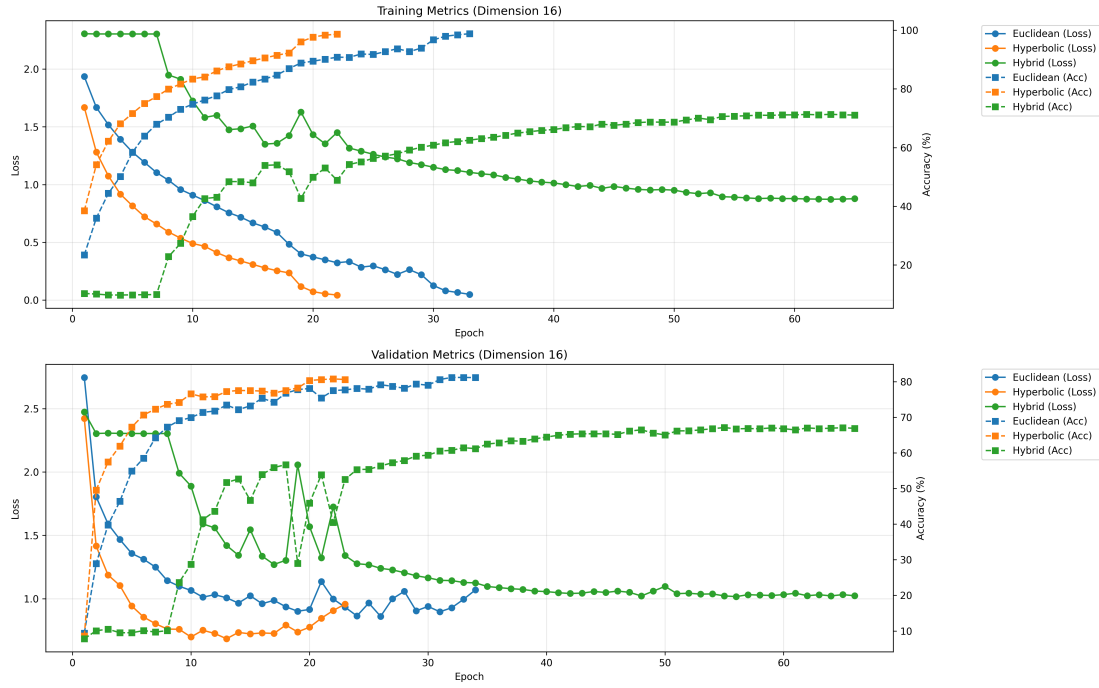


Figura 4.8: Comparación entrenamiento ResNet50 sobre CIFAR10 para $d = 16$.

El análisis por dimensión confirma que para bajas dimensiones ($d = 4, d = 8$), el modelo hiperbólico es claramente superior a los otros dos. A partir de $d = 16$, el modelo hiperbólico obtiene un rendimiento similar al euclidiano. El modelo híbrido no logra alcanzar un rendimiento comparable en ninguna dimensión.

4.4.2. CIFAR10 - EfficientNetV2

Las Figuras [4.9](#), [4.10](#) y [4.11](#) presentan los resultados de entrenamiento por dimensión para cada modelo.

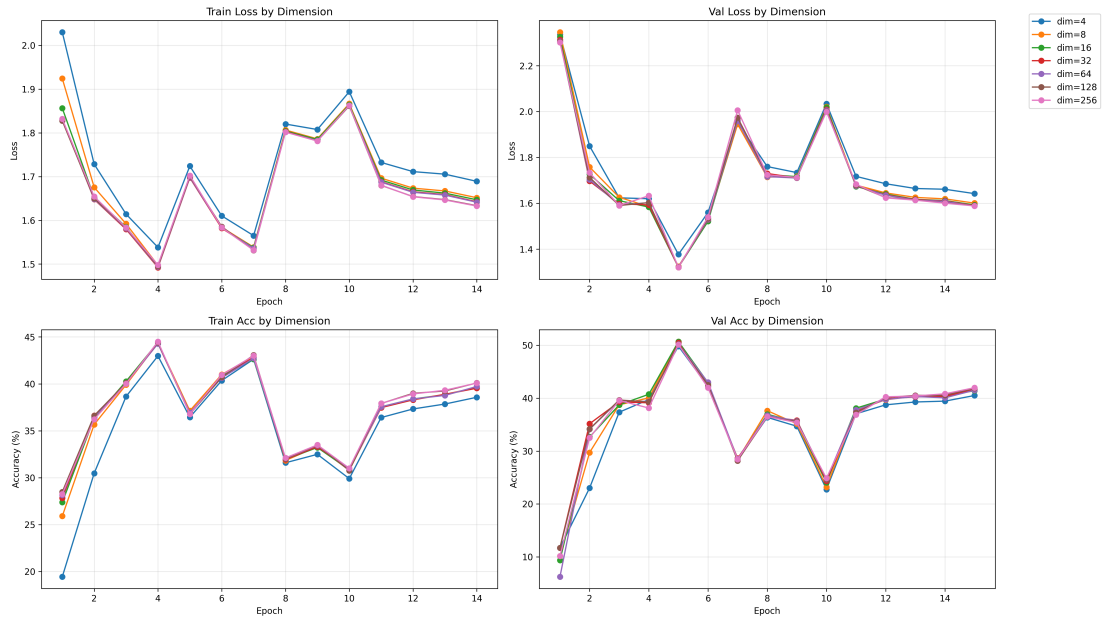


Figura 4.9: Resultados entrenamiento EfficientNetV2 euclidiano sobre CIFAR10.

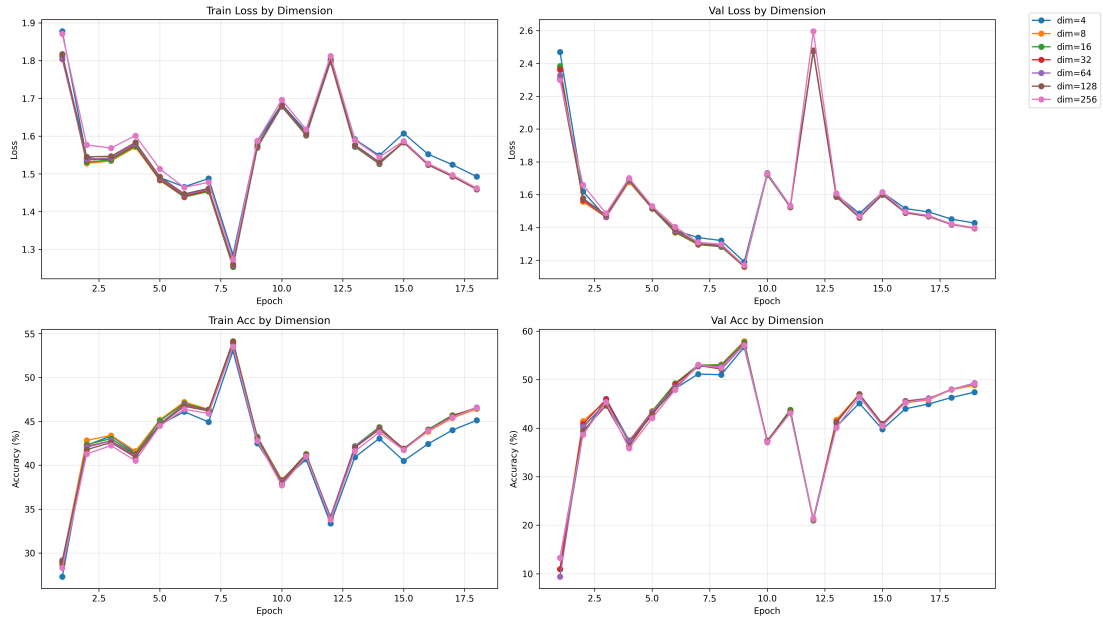


Figura 4.10: Resultados entrenamiento EfficientNetV2 hiperbólico sobre CIFAR10.

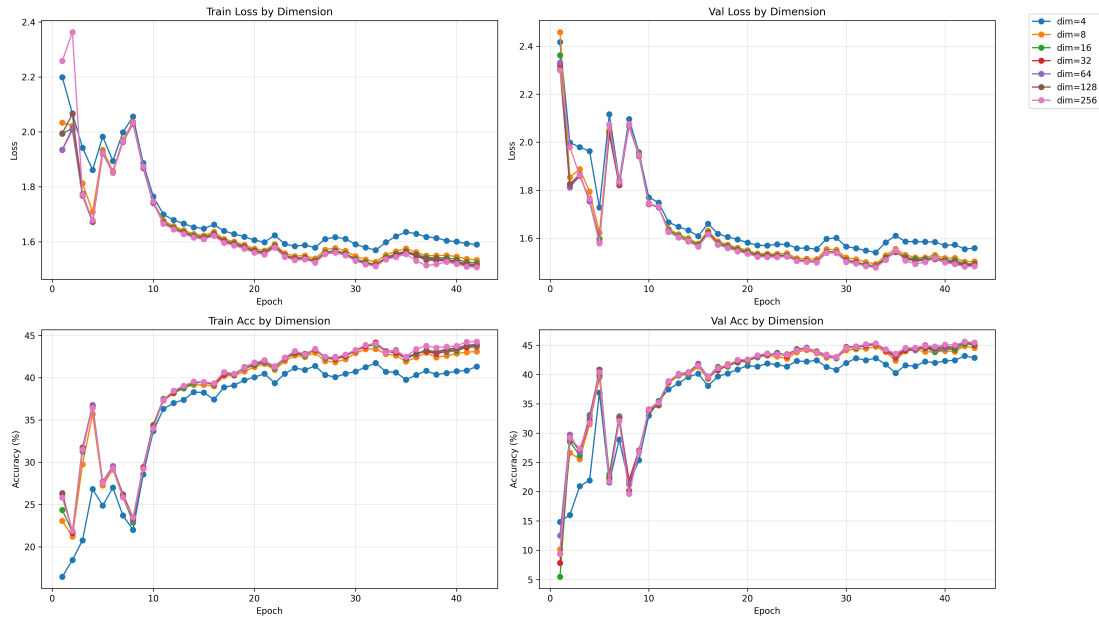
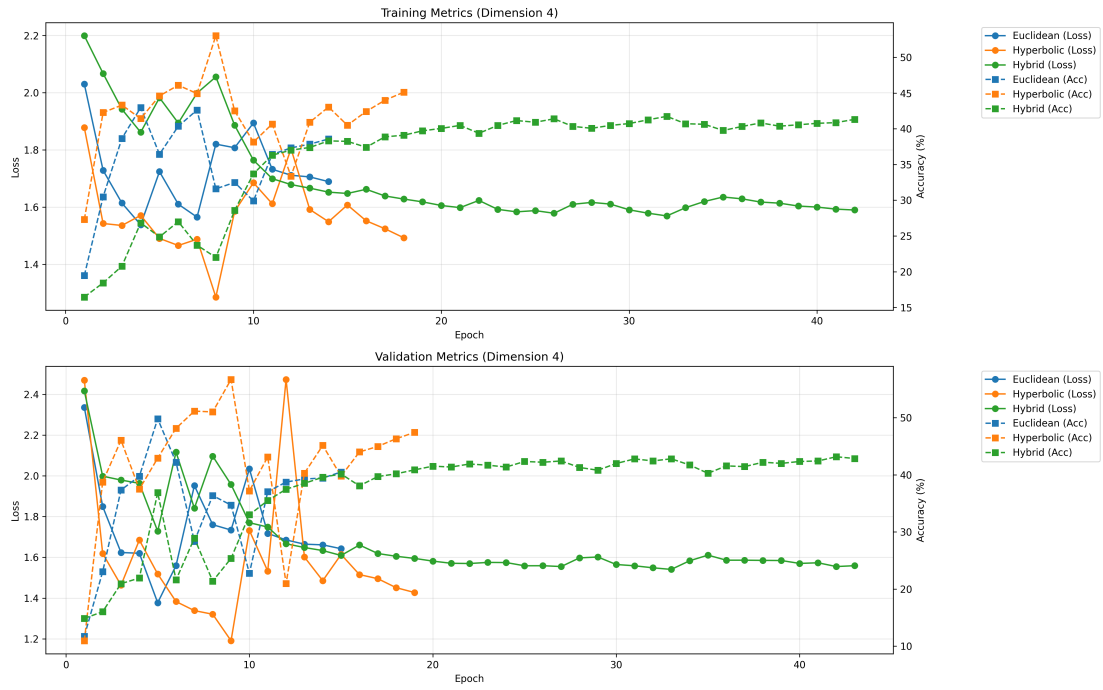
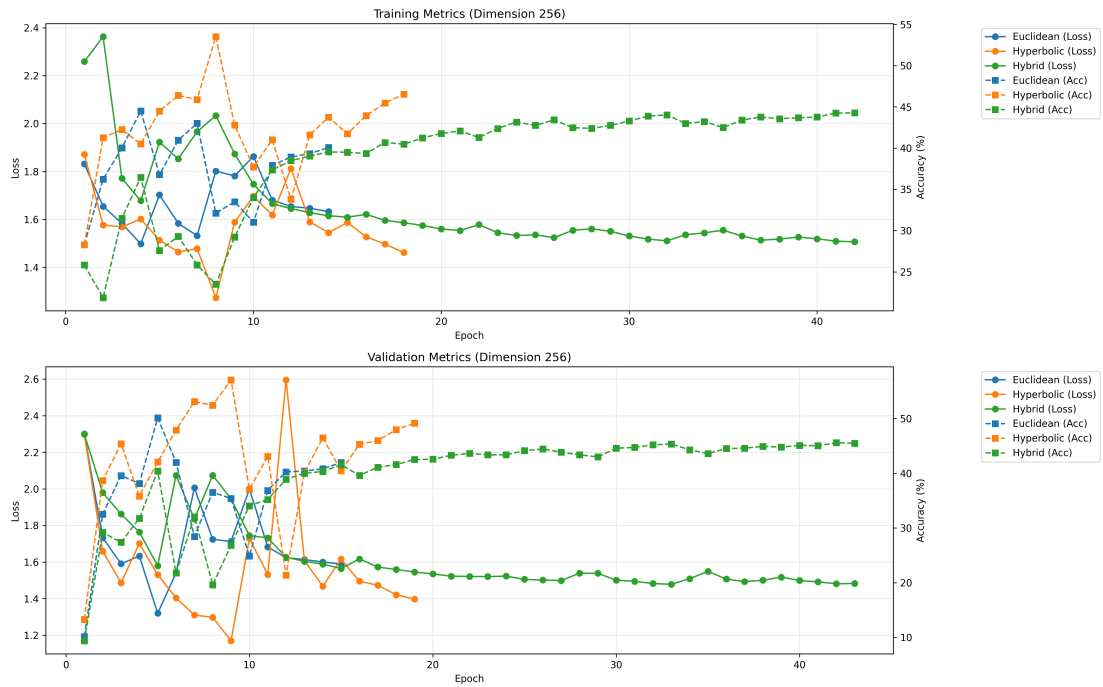


Figura 4.11: Resultados entrenamiento EfficientNetV2 híbrido sobre CIFAR10.

A diferencia de lo observado con ResNet50, con EfficientNetV2 los tres modelos logran aprender representaciones útiles en todas las dimensiones evaluadas. Esto resulta bastante interesante: incluso con un *embedding* de dimensión $d = 4$, los modelos alcanzan un rendimiento similar al obtenido en dimensiones mucho mayores como $d = 256$. Una posible explicación es que EfficientNetV2, al ser una arquitectura más expresiva y eficiente que ResNet50, es capaz de aprender la información relevante incluso en muy pocas dimensiones. A esto se suma que CIFAR10 es un dataset de baja complejidad, lo que reduce la cantidad de información discriminativa que el modelo necesita preservar, facilitando una representación compacta sin pérdida significativa de rendimiento.

Las Figuras [4.12](#) a [4.13](#) presentan la comparación detallada por dimensión.

Figura 4.12: Comparación entrenamiento EfficientNetV2 sobre CIFAR10 para $d = 4$.Figura 4.13: Comparación entrenamiento EfficientNetV2 sobre CIFAR10 para $d = 256$.

El análisis por dimensión revela que los modelos euclidiano e híbrido se comportan de manera si-

milar, alcanzando resultados comparables entre sí. El modelo hiperbólico mantiene un comportamiento consistente con lo observado en ResNet50, superando a los otros dos modelos en todas las dimensiones.

4.4.3. CIFAR100 - ResNet50

Las Figuras 4.14, 4.15 y 4.16 presentan los resultados de entrenamiento por dimensión para cada modelo.

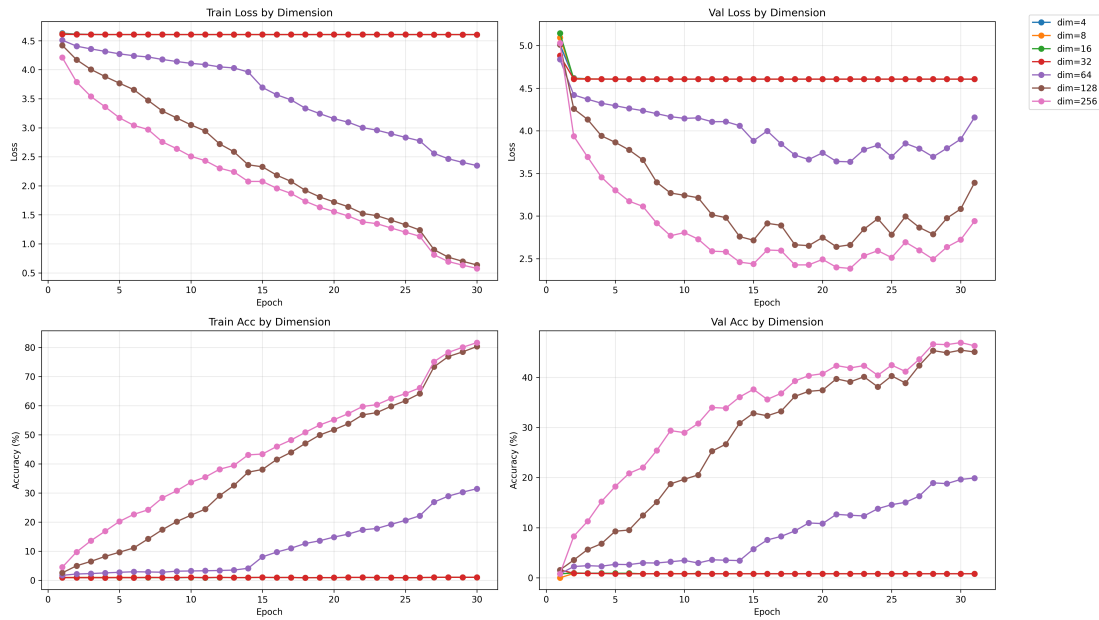


Figura 4.14: Resultados entrenamiento ResNet50 euclidiano sobre CIFAR100.

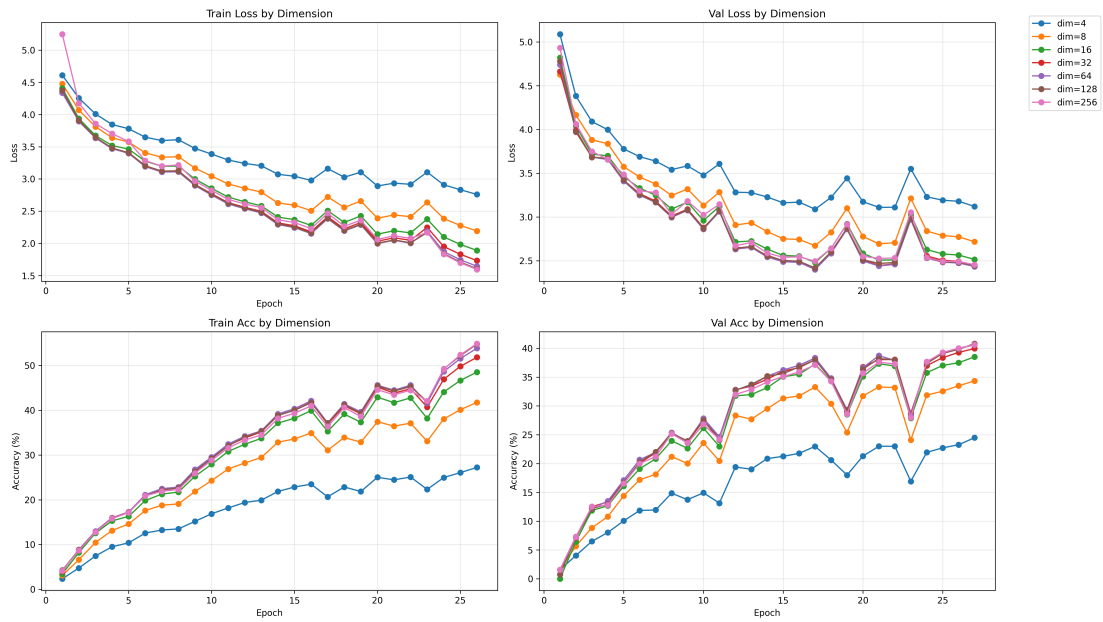


Figura 4.15: Resultados entrenamiento ResNet50 hiperbólico sobre CIFAR100.

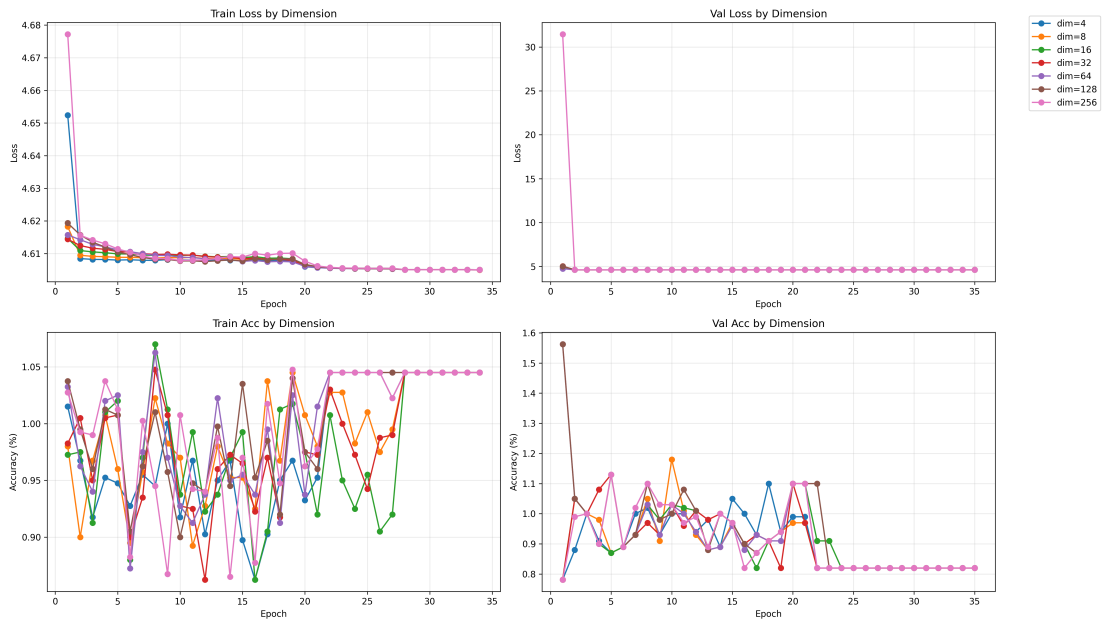


Figura 4.16: Resultados entrenamiento ResNet50 híbrido sobre CIFAR100.

El comportamiento del modelo euclidiano es similar al observado con CIFAR10: presenta dificultades para aprender en bajas dimensiones. El modelo hiperbólico, por su parte, aprende consistentemente en todas las dimensiones. El modelo híbrido no logra superar una *accuracy* del 1 %.

Las Figuras 4.17 y 4.18 comparan las pérdidas de entrenamiento y validación entre los tres modelos.

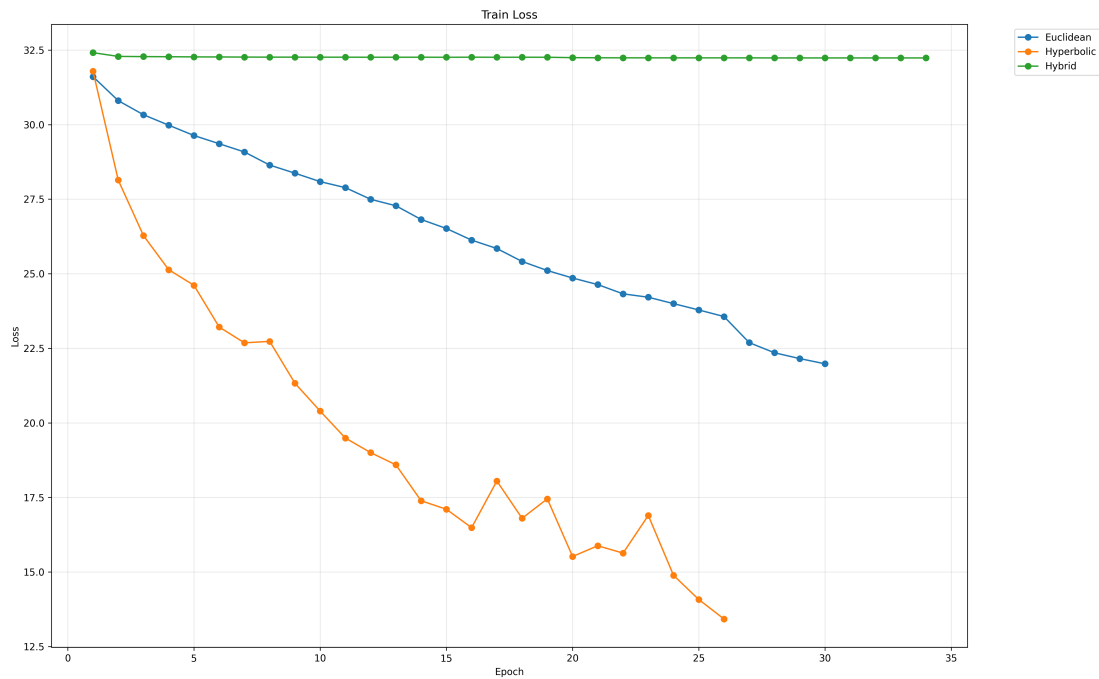


Figura 4.17: Comparación entrenamiento ResNet50 sobre CIFAR100.

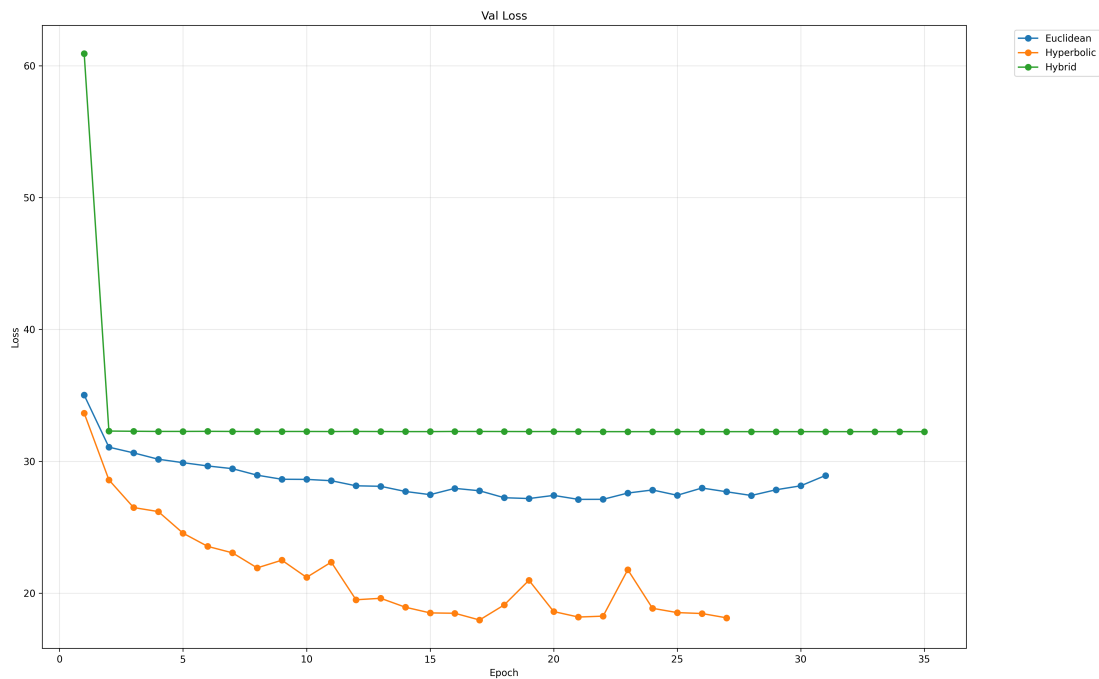


Figura 4.18: Comparación validación ResNet50 sobre CIFAR100.

En CIFAR100, la diferencia de rendimiento entre el modelo hiperbólico y los otros dos es considerablemente más pronunciada que en CIFAR10. Este resultado es consistente con la hipótesis de que los

embeddings hiperbólicos ofrecen mayores ventajas en datasets con mayor complejidad jerárquica. CIFAR100 presenta 100 clases organizadas en 20 superclases, lo que constituye una estructura taxonómica más profunda que las 10 clases de CIFAR10. Además, el mayor número de clases expone las limitaciones del espacio euclidiano en baja dimensión, donde el volumen disponible resulta insuficiente para separar adecuadamente tantas categorías.

Las Figuras 4.19 y 4.20 presentan la comparación detallada por dimensión.



Figura 4.19: Comparación entrenamiento ResNet50 sobre CIFAR100 para $d = 32$.

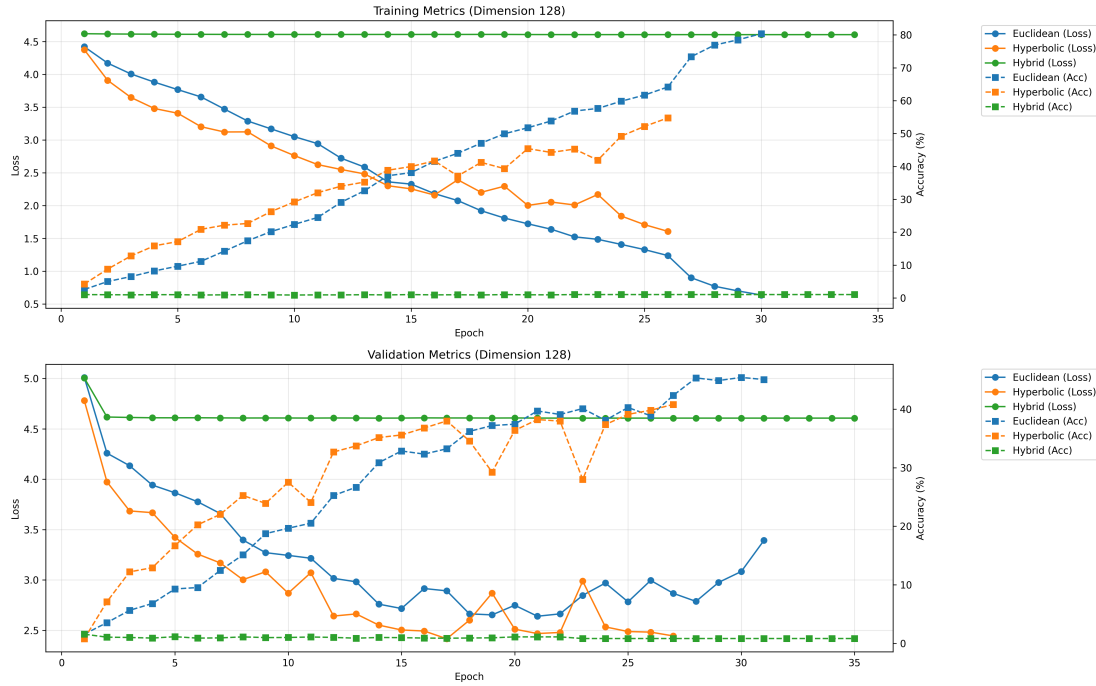


Figura 4.20: Comparación entrenamiento ResNet50 sobre CIFAR100 para $d = 128$.

El análisis por dimensión confirma lo anteriormente discutido: hasta $d = 32$, el modelo hiperbólico es el único capaz de aprender efectivamente sobre el dataset. A partir de $d = 64$, los modelos euclidiano e híbrido comienzan a alcanzar un rendimiento comparable, e incluso superior en algunas dimensiones. Este patrón sugiere que la ventaja del espacio hiperbólico es especialmente crítica cuando la capacidad representacional está severamente limitada.

4.4.4. CIFAR100 - EfficientNetV2

Las Figuras [4.21](#), [4.22](#) y [4.23](#) presentan los resultados de entrenamiento por dimensión para cada modelo.

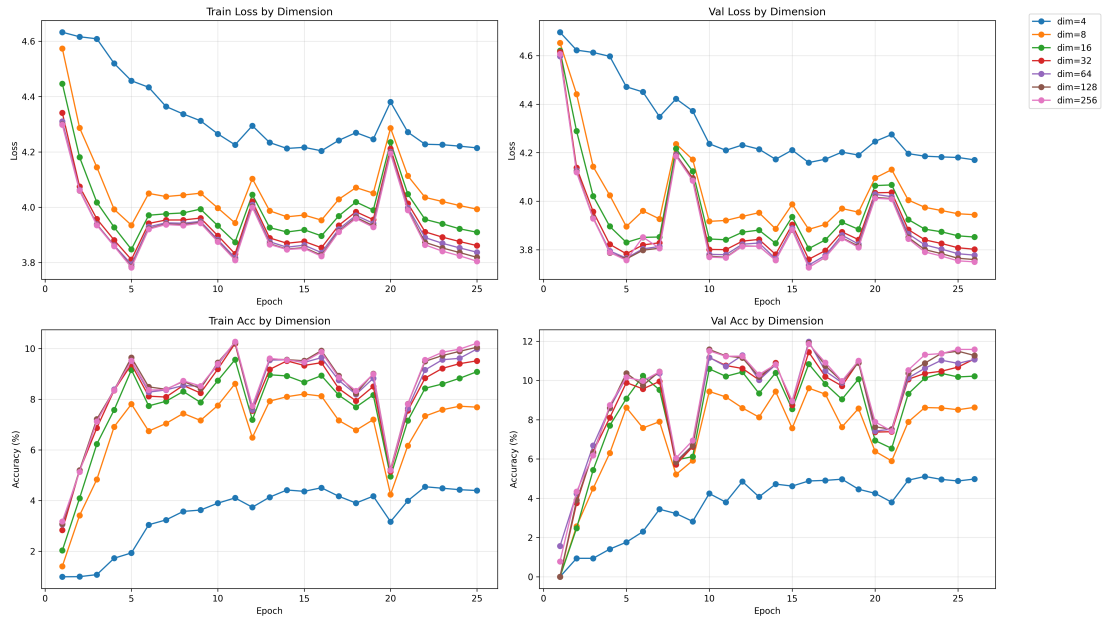


Figura 4.21: Resultados entrenamiento EfficientNetV2 euclidiano sobre CIFAR100.

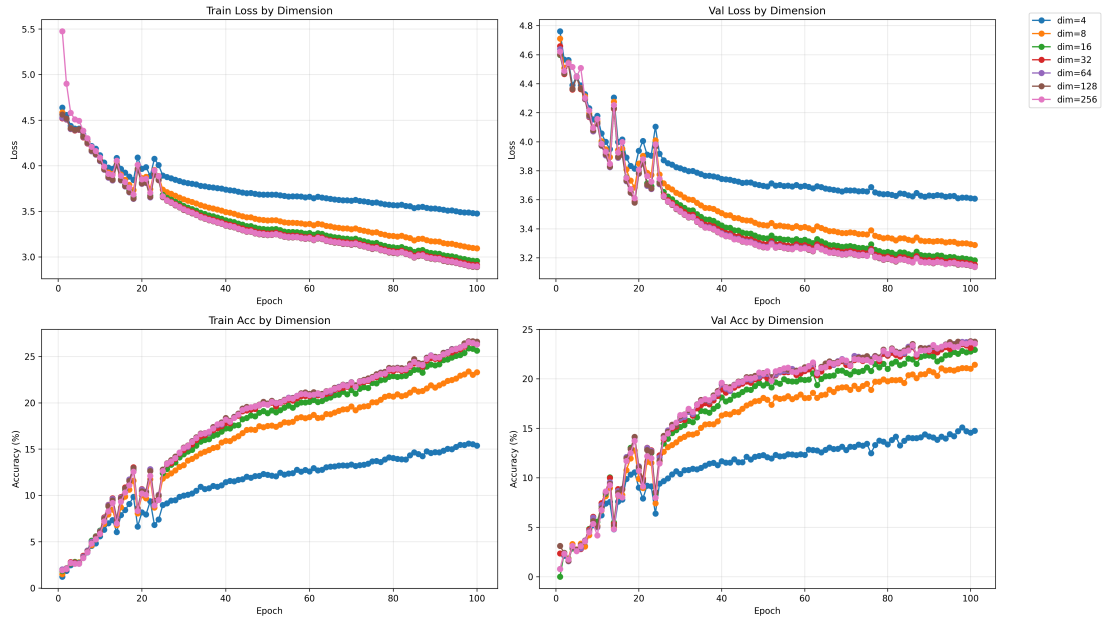


Figura 4.22: Resultados entrenamiento EfficientNetV2 hiperbólico sobre CIFAR100.

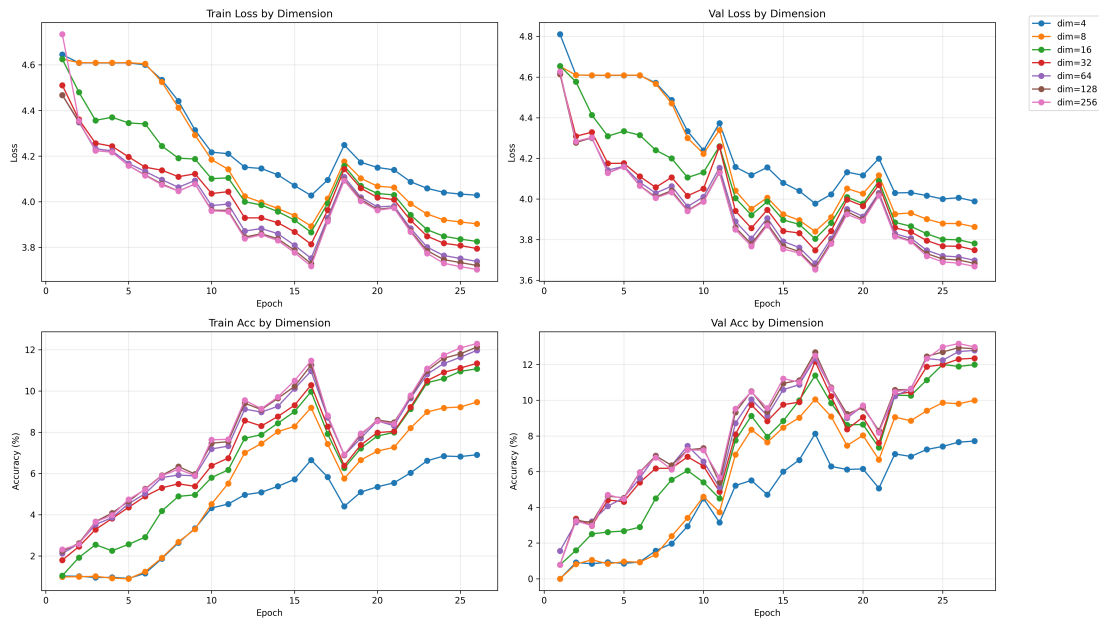


Figura 4.23: Resultados entrenamiento EfficientNetV2 híbrido sobre CIFAR100.

Al igual que en CIFAR10, con EfficientNetV2 los tres modelos logran aprender en todas las dimensiones, aunque todos presentan mayor dificultad en las dimensiones más bajas. A diferencia de lo ocurrido con CIFAR10, vemos que ya no se presenta el comportamiento "determinista.^{en} que todas las dimensiones tienen un rendimiento similar, lo que reafirma la hipótesis de que EfficientNetV2 es capaz de aprender la información relevante incluso en muy pocas dimensiones, pero que esta capacidad se ve comprometida a medida que la complejidad del dataset aumenta.

Las Figuras 4.24 y 4.25 comparan las pérdidas de entrenamiento y validación entre los tres modelos.

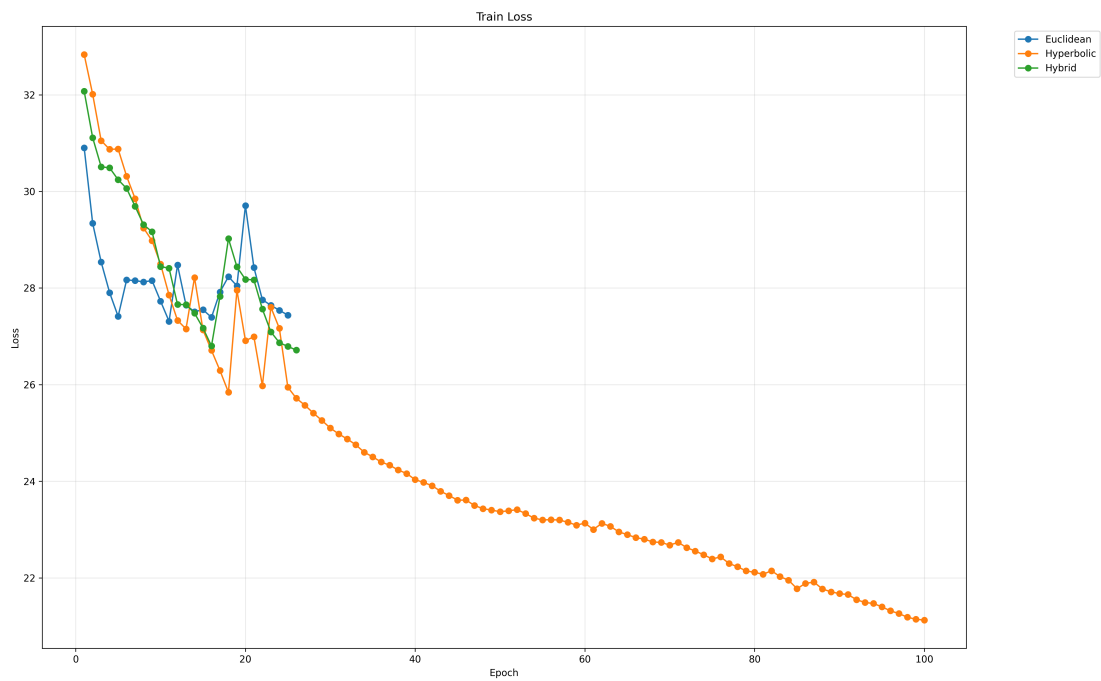


Figura 4.24: Comparación entrenamiento EfficientNetV2 sobre CIFAR100.

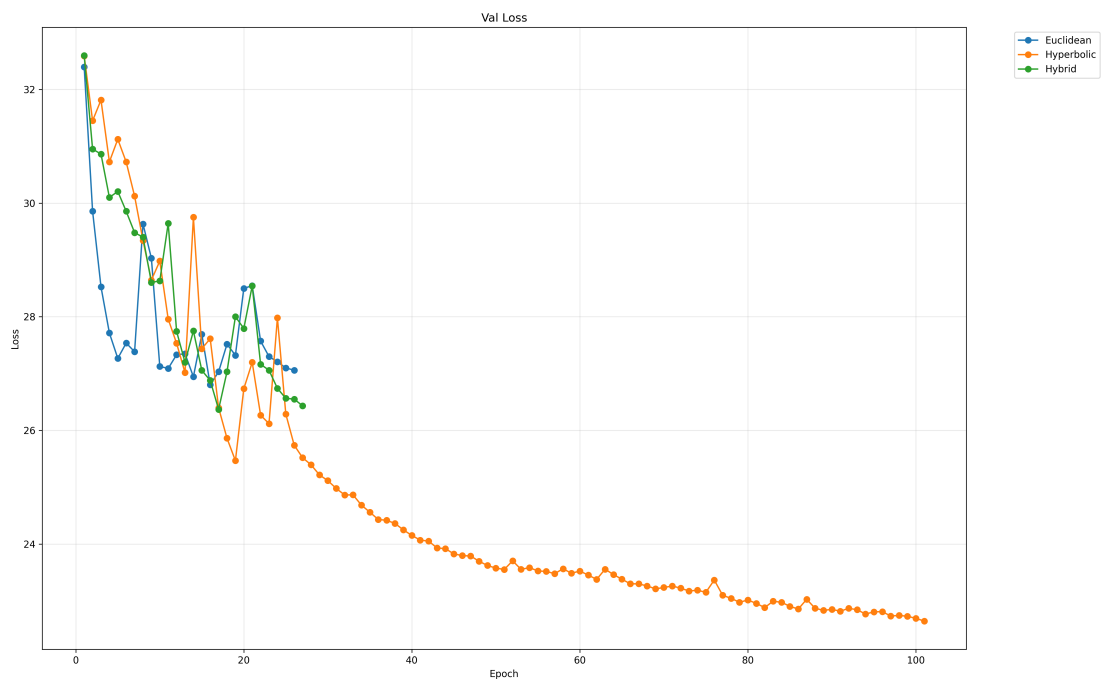


Figura 4.25: Comparación entrenamiento EfficientNetV2 sobre CIFAR100.

A diferencia de lo observado en CIFAR10, en este caso el modelo hiperbólico muestra una superioridad clara en ambas métricas.

4.4.5. Imagenette - ResNet50

Las Figuras 4.26, 4.27 y 4.28 presentan los resultados de entrenamiento por dimensión para cada modelo.

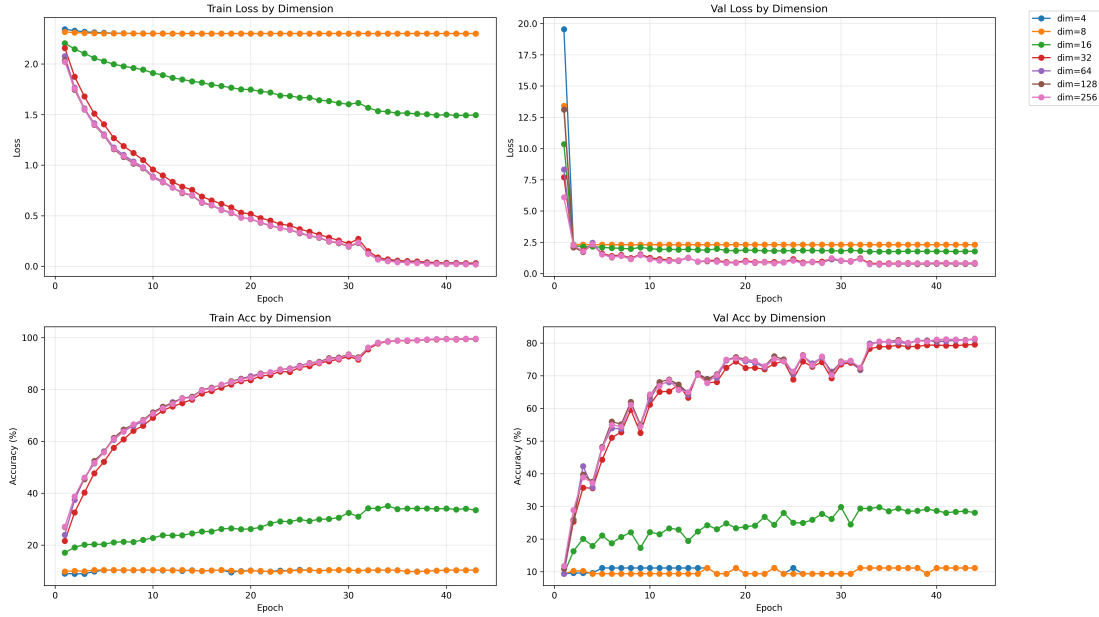


Figura 4.26: Resultados entrenamiento ResNet50 euclidiano sobre Imagenette.

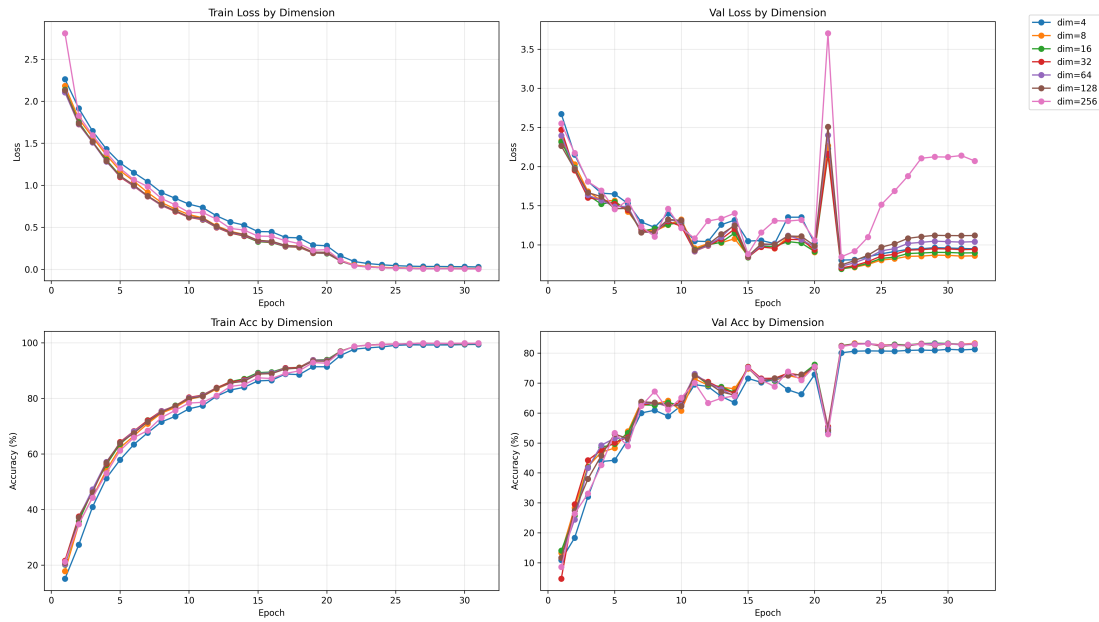


Figura 4.27: Resultados entrenamiento ResNet50 hiperbólico sobre Imagenette.

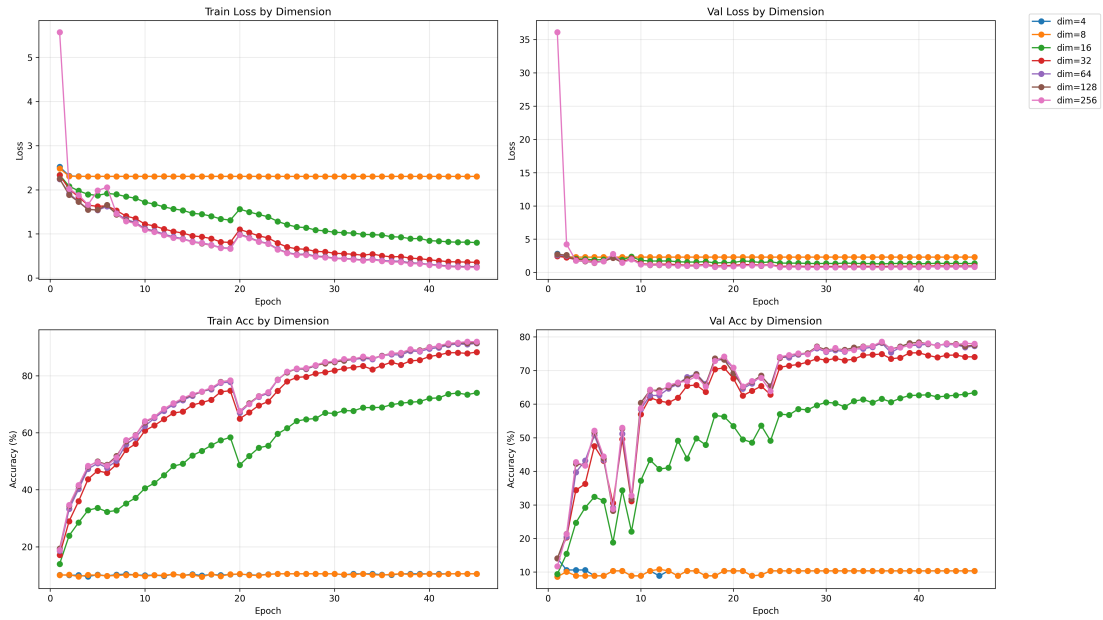


Figura 4.28: Resultados entrenamiento ResNet50 híbrido sobre Imagenette.

Nuevamente, los modelos euclidiano e híbrido presentan dificultades para aprender cuando el *embedding* es de baja dimensión, mientras que el modelo hiperbólico aprende consistentemente en todas las dimensiones.

Las Figuras 4.29 y 4.30 comparan las pérdidas de entrenamiento y validación entre los tres modelos.

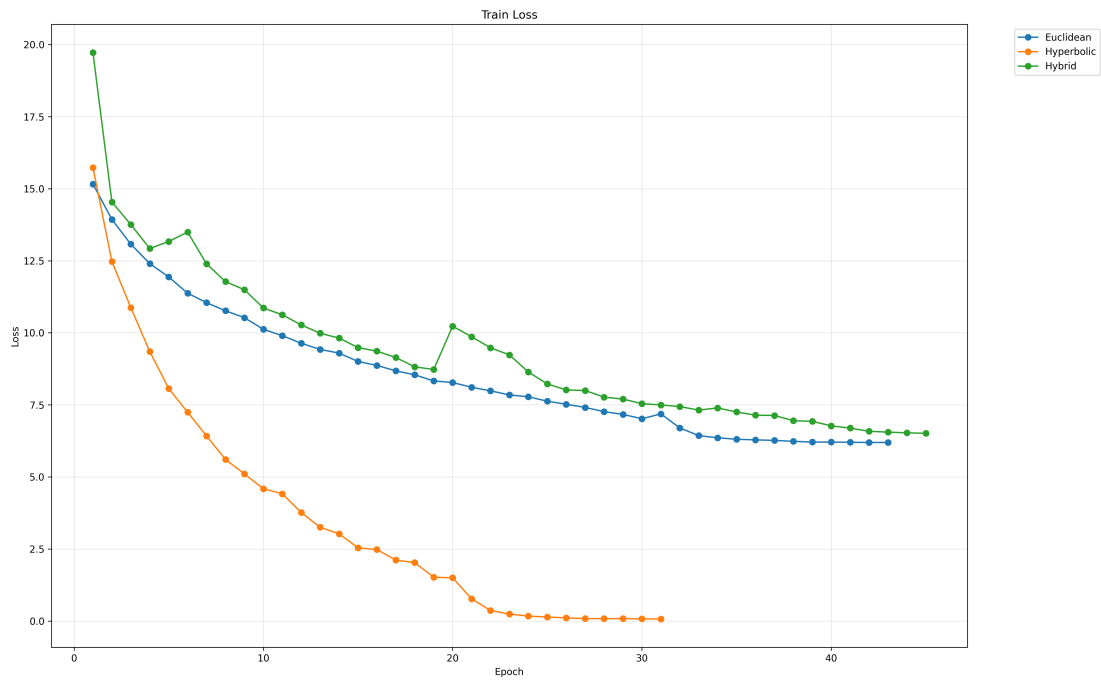


Figura 4.29: Comparación entrenamiento ResNet50 sobre Imagenette.

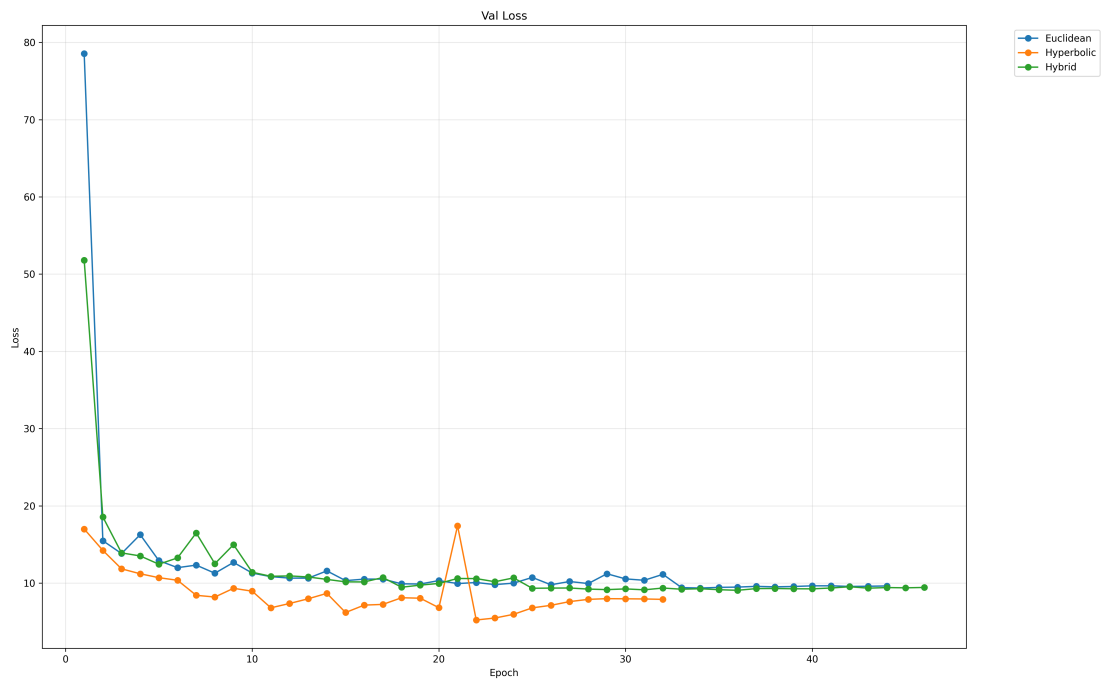


Figura 4.30: Comparación validación ResNet50 sobre Imagenette.

La comparación de pérdidas muestra que los tres modelos resultan competitivos en términos de pérdida de validación.

Las Figuras 4.31 a 4.32 presentan la comparación detallada por dimensión.

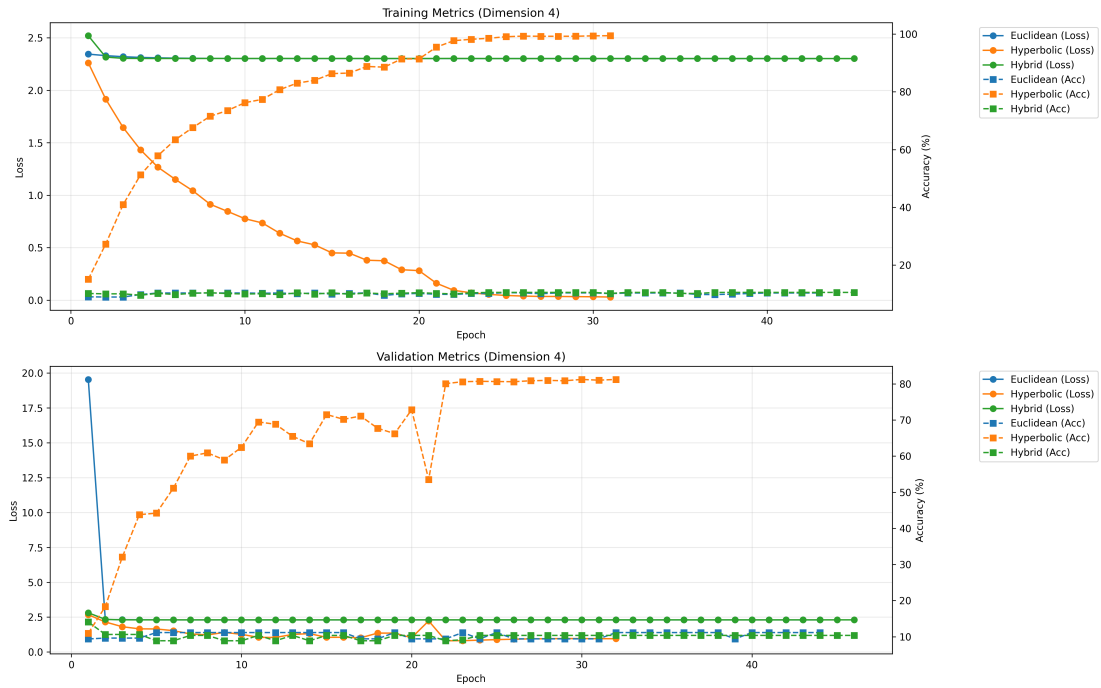


Figura 4.31: Comparación entrenamiento ResNet50 sobre Imagenette para $d = 4$.

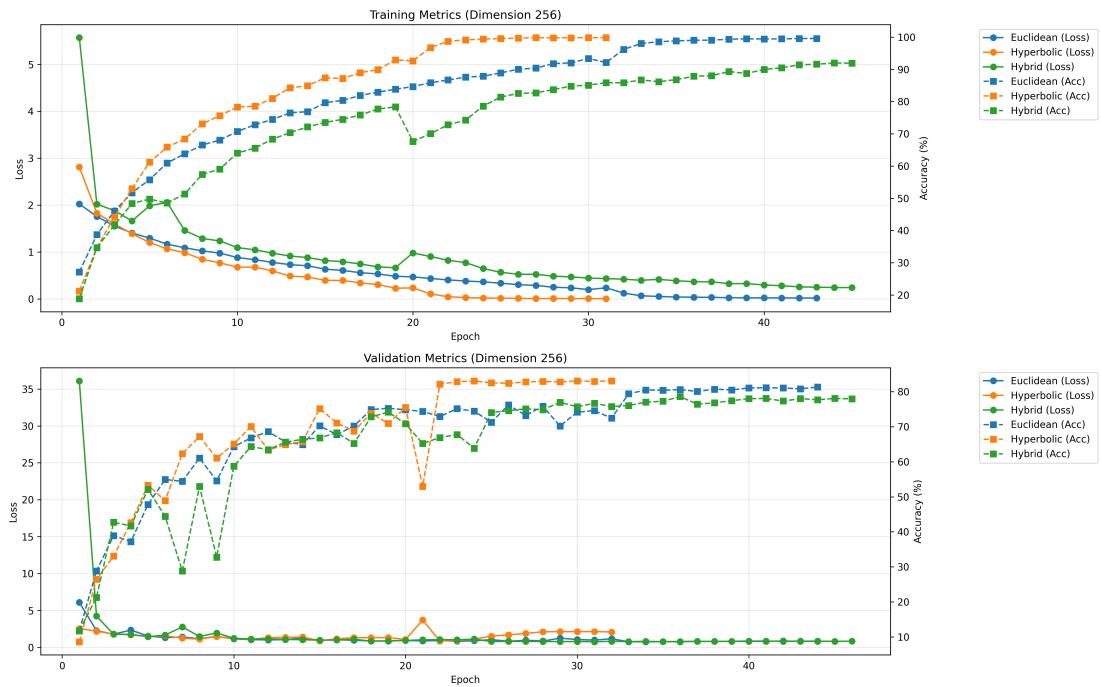


Figura 4.32: Comparación entrenamiento ResNet50 sobre Imagenette para $d = 256$.

El análisis por dimensión muestra un patrón consistente con los experimentos anteriores: el modelo hiperbólico domina en las dimensiones más bajas, mientras que en dimensiones mayores los modelos convergen a rendimientos similares. Cabe destacar que Imagenette, a diferencia de los datasets CIFAR, no posee una estructura jerárquica explícita. Sin embargo, el modelo hiperbólico sigue presentando ventajas en baja dimensión, lo que sugiere que los beneficios de la geometría hiperbólica no se limitan exclusivamente a datasets con jerarquías predefinidas, sino que también aplican cuando las restricciones de dimensionalidad son severas.

4.4.6. Imagenette - EfficientNetV2

Las Figuras 4.33, 4.34 y 4.35 presentan los resultados de entrenamiento por dimensión para cada modelo.

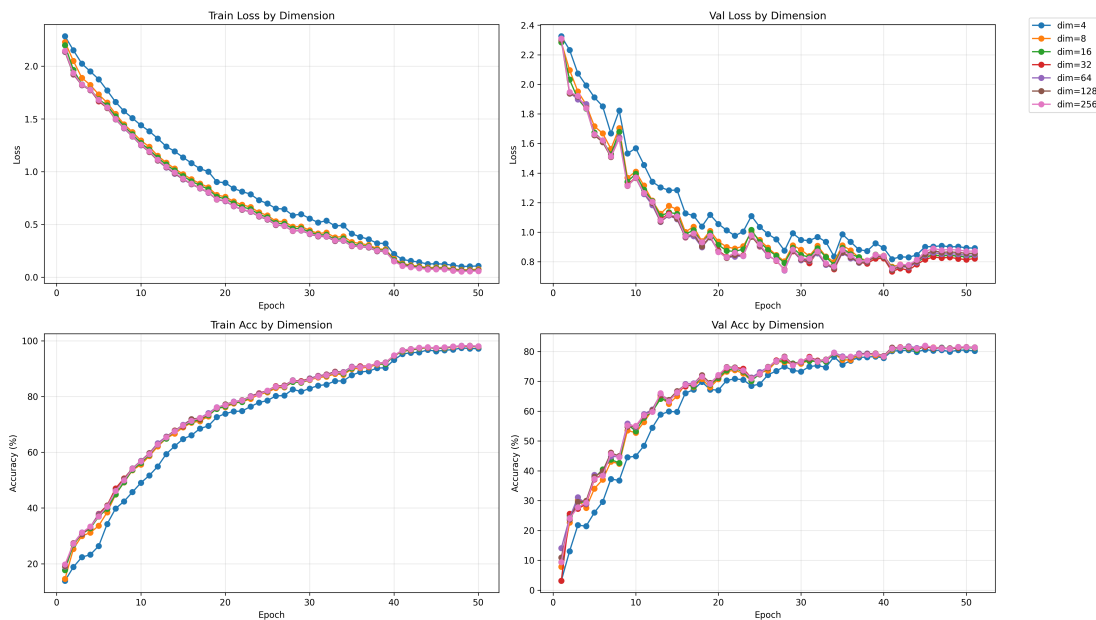


Figura 4.33: Resultados entrenamiento EfficientNetV2 euclidiano sobre Imagenette.

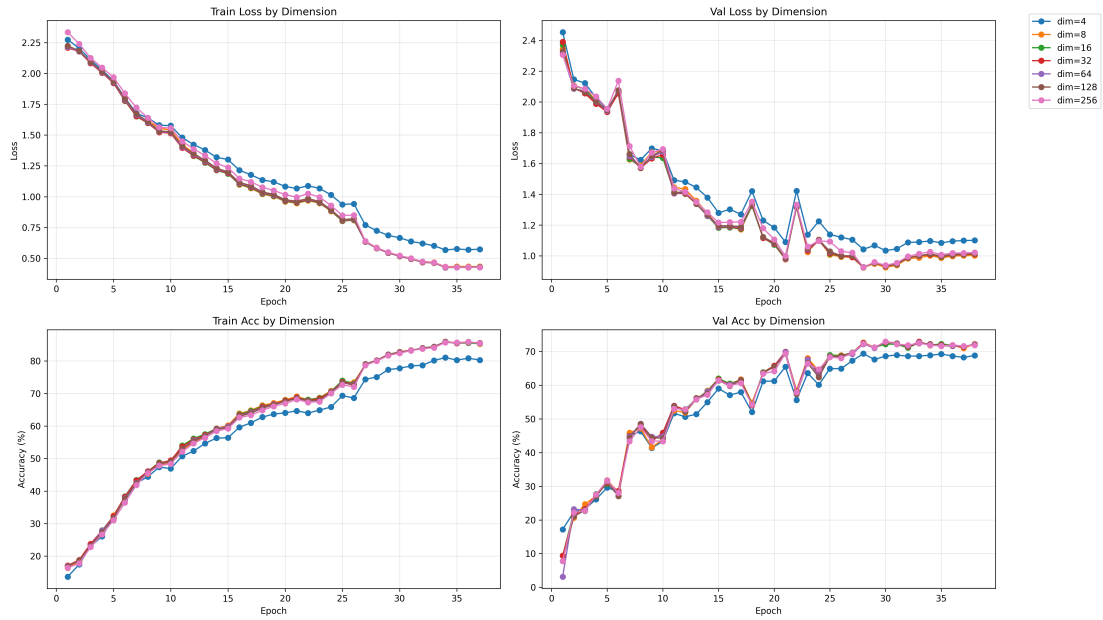


Figura 4.34: Resultados entrenamiento EfficientNetV2 hiperbólico sobre Imagenette.

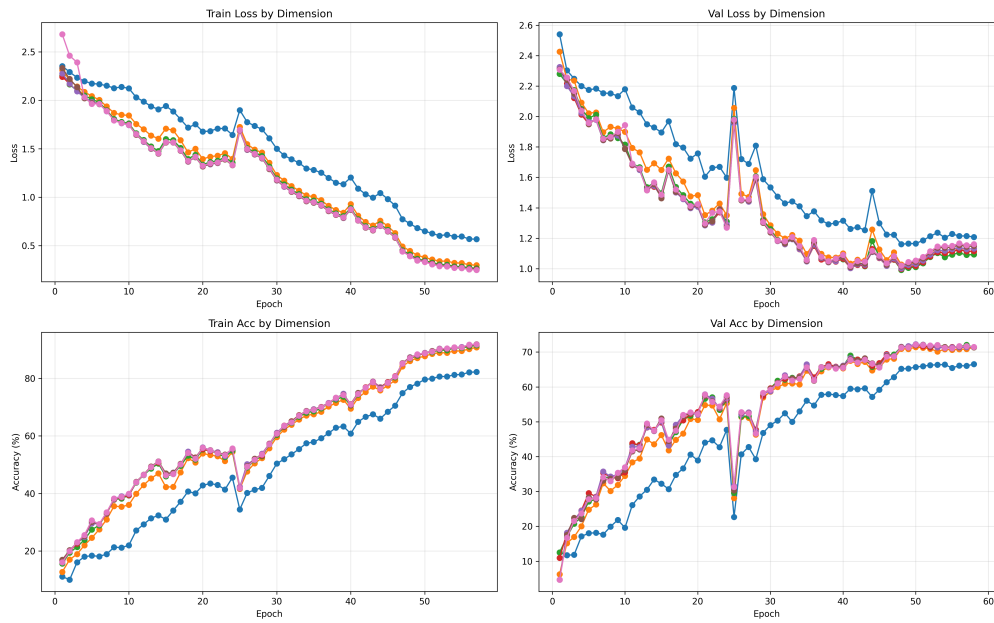


Figura 4.35: Resultados entrenamiento EfficientNetV2 híbrido sobre Imagenette.

De manera consistente con los experimentos anteriores de EfficientNetV2, los tres modelos logran aprender en todas las dimensiones evaluadas.

Las Figuras 4.36 y 4.37 comparan las pérdidas de entrenamiento y validación entre los tres modelos.

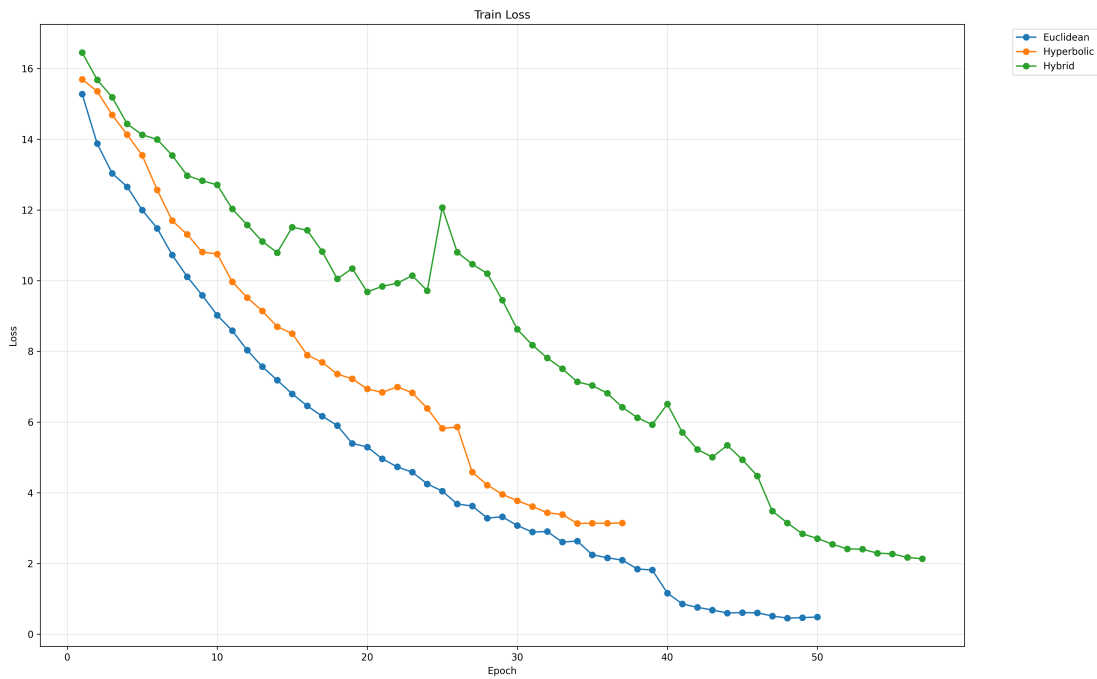


Figura 4.36: Comparación entrenamiento EfficientNetV2 sobre Imagenette.

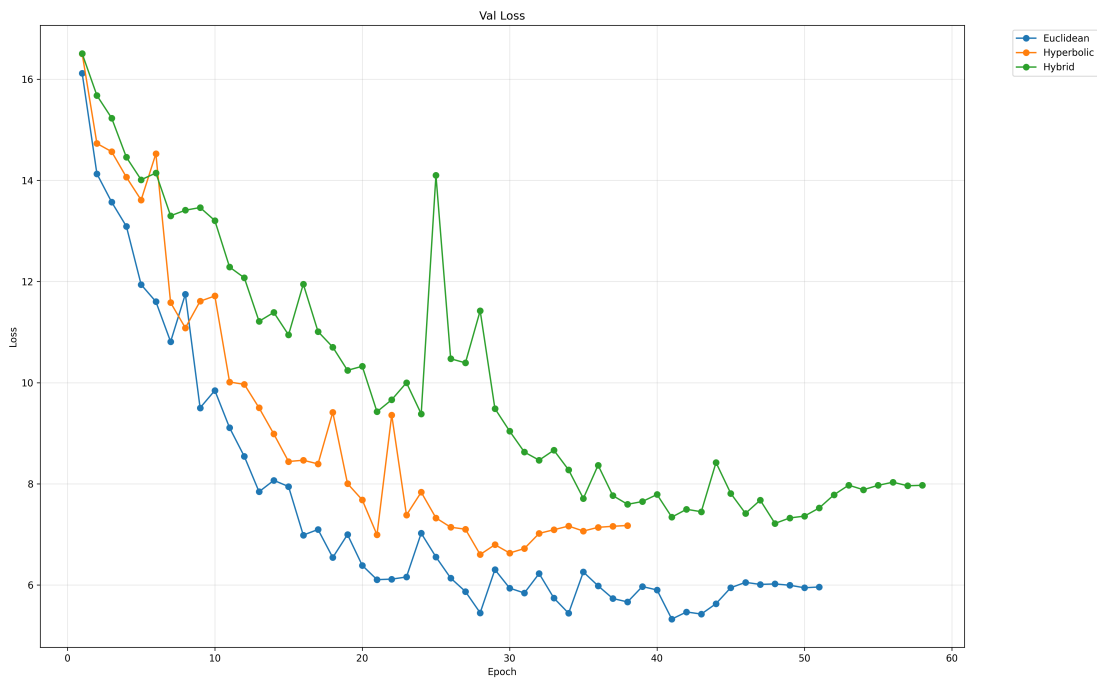


Figura 4.37: Comparación entrenamiento EfficientNetV2 sobre Imagenette.

En este caso, el modelo euclidiano logra un mejor rendimiento general en ambas métricas evaluadas. La exploración del rendimiento por dimensiones nos da una mejor explicación sobre este fenómeno.

Las Figuras 4.38 a 4.39 presentan la comparación detallada por dimensión.

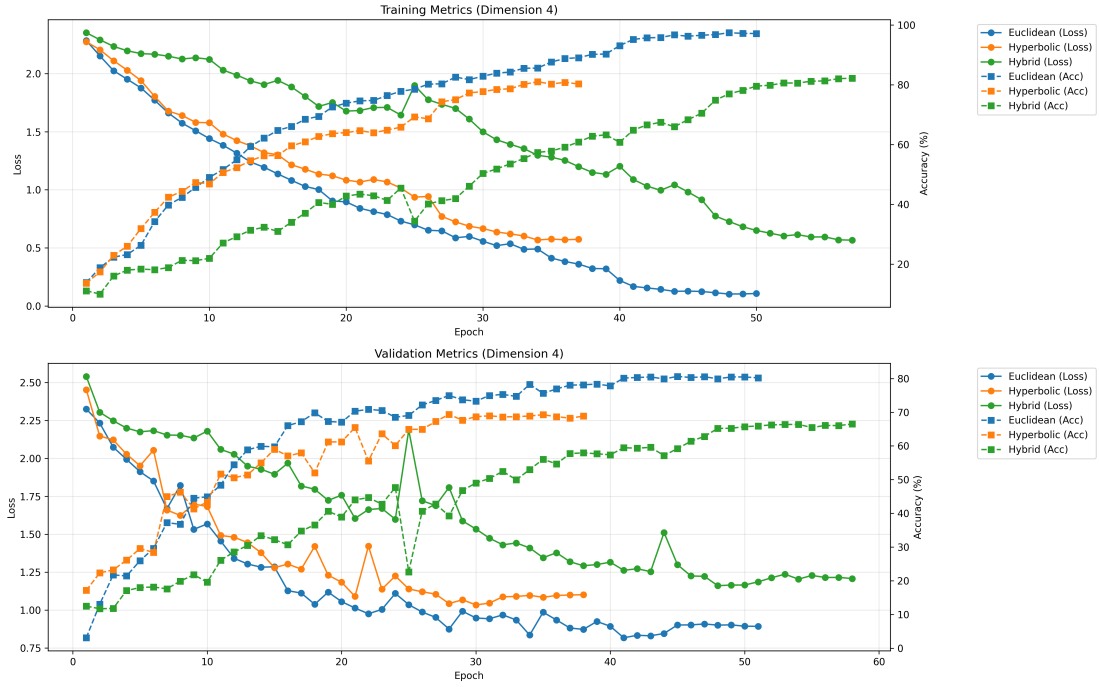


Figura 4.38: Comparación entrenamiento EfficientNetV2 sobre Imagenette para $d = 4$.

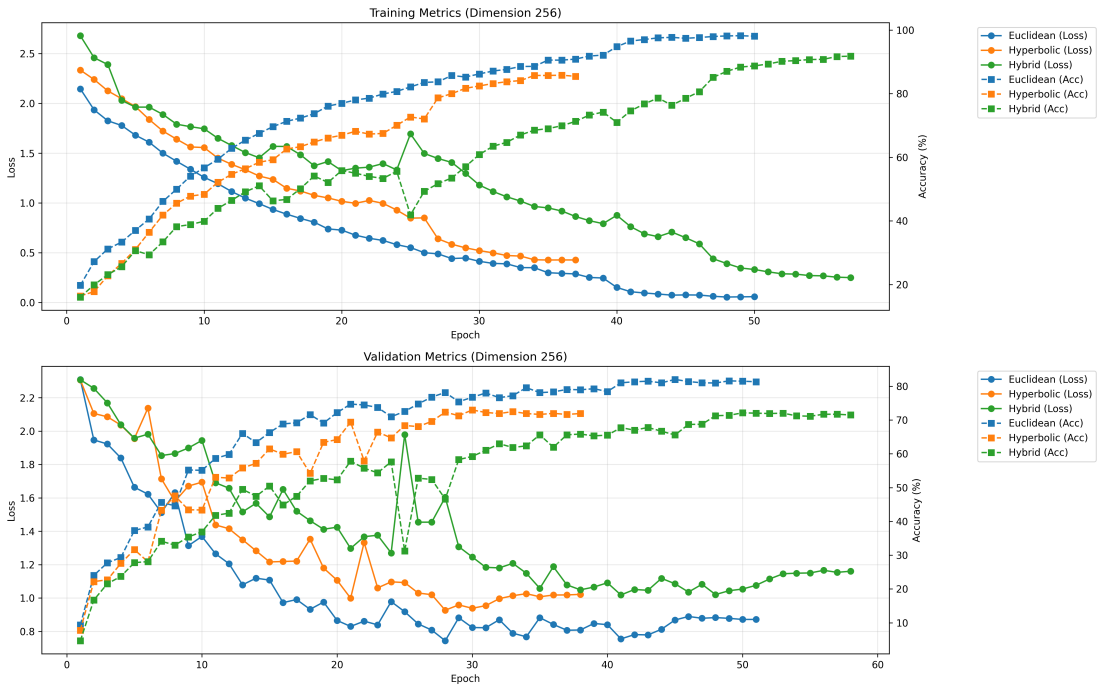


Figura 4.39: Comparación entrenamiento EfficientNetV2 sobre Imagenette para $d = 256$.

El análisis por dimensión revela que, para Imagenette con EfficientNetV2, el modelo euclidiano no solo logra aprender en todas las dimensiones evaluadas, sino que supera al hiperbólico en varias de ellas. Este comportamiento puede explicarse por la suma de dos factores. Por un lado, Imagenette no presenta una estructura jerárquica clara entre sus clases, lo que limita la ventaja intrínseca de la geometría hiperbólica. Por otro lado, EfficientNetV2 fue creada con el espacio euclidiano en mente, y por ende puede poseer bloques cuyas operaciones internas favorecen representaciones bien separables en espacio euclidiano. En dimensiones bajas, los beneficios de la geometría hiperbólica son suficientes para compensar estos factores: sin embargo, a medida que la dimensión aumenta, la geometría euclidiana logra explotar de mejor manera esta ventaja en la arquitectura de la red.

4.5. Resumen de Resultados

La Tabla 4.1 presenta un resumen de la exactitud obtenida sobre el conjunto de test para todas las configuraciones evaluadas.

Tabla 4.1: *Accuracy (%) sobre test para todas las configuraciones de Matrioshka Embeddings.*

Dataset	Modelo	$d = 4$	$d = 8$	$d = 16$	$d = 32$	$d = 64$	$d = 128$	$d = 256$
<i>ResNet50</i>								
CIFAR10	Euclidiano	10.00	54.87	78.72	79.35	79.87	79.94	79.89
	Hiperbólico	76.79	77.09	77.43	77.42	77.35	77.24	76.95
	Híbrido	47.82	48.06	66.68	68.82	69.36	69.25	69.24
CIFAR100	Euclidiano	1.00	1.00	1.00	1.00	12.80	40.77	42.35
	Hiperbólico	22.96	32.56	36.27	37.23	37.26	36.72	36.53
	Híbrido	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Imagenette	Euclidiano	9.10	9.10	30.85	79.49	80.05	80.48	80.48
	Hiperbólico	79.46	81.63	81.55	81.55	81.50	81.48	81.58
	Híbrido	9.10	9.10	62.39	74.78	77.12	77.66	77.91
<i>EfficientNetV2-S</i>								
CIFAR10	Euclidiano	49.88	50.77	50.86	50.70	50.22	50.11	50.13
	Hiperbólico	57.15	57.95	57.98	57.70	57.58	57.46	57.54
	Híbrido	43.21	44.98	45.08	45.21	45.35	45.14	45.33
CIFAR100	Euclidiano	4.99	9.54	10.83	11.29	11.57	11.23	11.57
	Hiperbólico	15.18	21.52	23.40	23.96	24.04	24.08	23.95
	Híbrido	7.48	10.30	11.67	12.08	12.97	12.95	13.08
Imagenette	Euclidiano	80.18	80.82	81.25	81.40	81.61	81.43	81.38
	Hiperbólico	69.22	72.54	72.43	72.41	72.36	72.25	72.48
	Híbrido	65.94	71.03	72.36	72.25	72.20	72.33	71.90

Los resultados confirman las hipótesis planteadas en este trabajo. En primer lugar, el modelo hi-

perbólico demuestra una ventaja consistente en las dimensiones más bajas ($d = 4, d = 8$), donde el espacio euclidiano presenta severas limitaciones de capacidad representacional. Esta ventaja es especialmente pronunciada en CIFAR100, el dataset con mayor complejidad jerárquica, donde el modelo euclidiano con ResNet50 no logra superar el 1 % de exactitud hasta $d = 64$, mientras que el hiperbólico alcanza un 22.96 % incluso con $d = 4$. En segundo lugar, se observa que a medida que la dimensión aumenta, la brecha entre los modelos tiende a reducirse, e incluso el modelo euclidiano llega a superar al hiperbólico en algunas configuraciones de alta dimensión (e.g., CIFAR10 con ResNet50 para $d \geq 128$). La única excepción a estos patrones es Imagenette con EfficientNetV2-S, donde el modelo euclidiano domina en todas las dimensiones. Este resultado puede atribuirse a la ausencia de estructura jerárquica explícita en Imagenette, combinada con la mayor capacidad de EfficientNetV2-S para extraer características discriminativas sin depender de la geometría del espacio de *embedding*. Finalmente, el modelo híbrido no logra igualar el rendimiento de los otros dos en ninguna configuración, lo que sugiere que la combinación de capas euclidianas con clasificadores hiperbólicos no ofrece beneficios claros.

4.6. Análisis de la Curvatura Aprendida

Dado que la curvatura c actúa como parámetro entrenable, el análisis de los valores aprendidos al término del entrenamiento permite evaluar la sensibilidad de los resultados respecto al valor inicial $c_0 = 0.05$. La Tabla 4.2 resume los valores de curvatura al final del entrenamiento para el modelo hiperbólico y el modelo híbrido en cada configuración.

Tabla 4.2: Valores de curvatura aprendida al final del entrenamiento (experimentos sin pesos preentrenados).

Dataset	Extractor	Hiperbólico	Híbrido
CIFAR10	ResNet50	0.022	< 0.001
	EfficientNetV2	0.008	< 0.001
CIFAR100	ResNet50	0.008	0.022
	EfficientNetV2	0.005	< 0.001
Imagenette	ResNet50	0.018	< 0.001
	EfficientNetV2	0.007	< 0.001

En todos los experimentos, el modelo hiperbólico converge a valores de curvatura en el rango $[0.005, 0.022]$, independientemente del valor de inicialización $c_0 = 0.05$. Este comportamiento indica que el optimizador ajusta c hacia el valor apropiado para cada tarea, lo que valida la elección de tratarla como parámetro entrenable: dado que el modelo encuentra su propio valor de equilibrio, la elección del punto de partida es menos crítica que en el caso de curvatura fija, y los resultados presentados son robustos respecto a variaciones moderadas de c_0 .

El modelo híbrido exhibe un comportamiento distinto: en cinco de las seis configuraciones, la curvatura colapsa a valores cercanos a 0 ($c < 0.001$) durante las primeras épocas y permanece en ese rango hasta el final del entrenamiento. La única excepción es CIFAR100 con ResNet50, donde el modelo híbrido estabiliza su curvatura en torno a 0.022. Esta degeneración implica que el clasificador hiperbólico del modelo híbrido opera en un espacio que se aproxima progresivamente a la geometría euclidiana, lo que ofrece una explicación geométrica coherente con el bajo rendimiento observado en este modelo: la combinación de capas euclidianas con un clasificador hiperbólico no logra sostener una curvatura efectiva durante el entrenamiento.

4.7. Análisis Exploratorio: Pesos Preentrenados

Los experimentos presentados anteriormente entrenan los extractores de características desde cero para garantizar una comparación justa entre las arquitecturas. Sin embargo, en la práctica es común utilizar pesos preentrenados de ImageNet como punto de partida, lo que generalmente mejora el rendimiento y acelera la convergencia. Este estudio de ablación evalúa el impacto de utilizar pesos preentrenados en el rendimiento relativo de los modelos euclidiano, hiperbólico e híbrido.

4.7.1. Configuración

Se replican los experimentos anteriores manteniendo la misma configuración de arquitectura, hiperparámetros y datasets (CIFAR10, CIFAR100 e Imagenette), con la única diferencia de inicializar los extractores de características (ResNet50 y EfficientNetV2-S) con pesos preentrenados de ImageNet en lugar de pesos aleatorios.

4.7.2. Resultados

La Tabla 4.3 presenta la *accuracy* sobre el conjunto de test para cada configuración utilizando pesos preentrenados de ImageNet.

Tabla 4.3: Accuracy (%) sobre test con pesos preentrenados de ImageNet.

Dataset	Modelo	$d = 4$	$d = 8$	$d = 16$	$d = 32$	$d = 64$	$d = 128$	$d = 256$
<i>ResNet50</i>								
CIFAR10	Euclidiano	87.54	87.75	87.72	87.84	87.80	87.89	87.84
	Hiperbólico	87.30	87.71	87.69	87.67	87.71	87.66	87.67
	Híbrido	87.89	88.13	88.00	88.08	87.94	88.05	87.96
CIFAR100	Euclidiano	53.81	60.88	62.54	62.94	63.05	63.25	63.18
	Hiperbólico	55.34	60.26	61.19	61.37	61.27	61.40	61.34
	Híbrido	54.85	60.55	61.79	62.23	62.31	62.26	62.42
Imagenette	Euclidiano	94.09	94.73	95.08	95.16	95.21	95.08	95.26
	Hiperbólico	94.93	96.61	96.59	97.27	97.20	97.12	96.31
	Híbrido	92.33	93.38	93.91	93.96	94.19	93.94	94.01
<i>EfficientNetV2-S</i>								
CIFAR10	Euclidiano	87.08	87.52	87.59	87.62	87.63	87.64	87.66
	Hiperbólico	85.48	85.87	85.75	85.73	85.66	85.77	85.76
	Híbrido	84.98	85.45	85.17	85.16	85.21	85.14	85.06
CIFAR100	Euclidiano	50.49	59.14	61.42	61.96	62.39	62.46	62.53
	Hiperbólico	15.89	24.59	27.75	28.98	28.34	28.11	27.12
	Híbrido	35.78	48.79	52.30	53.15	53.38	53.32	53.36
Imagenette	Euclidiano	94.29	95.03	95.31	95.18	95.13	95.13	95.18
	Hiperbólico	93.25	94.29	94.32	94.14	94.19	94.11	94.29
	Híbrido	94.47	95.13	95.24	95.24	95.29	95.41	95.34

Los resultados de este experimento tienen implicaciones directas sobre la hipótesis central de este trabajo. Con pesos preentrenados, la ventaja del modelo hiperbólico en bajas dimensiones se ve drásticamente reducida, lo que amerita una discusión más detallada sobre el extractor de características. Los pesos preentrenados en ImageNet generan *embeddings* que ya codifican la estructura semántica y visual de los datos de forma eficiente en el espacio euclidiano. Cuando el extractor opera en este régimen, la tarea de clasificación en bajas dimensiones se simplifica notablemente, reduciendo la necesidad de la eficiencia geométrica que caracteriza al espacio hiperbólico.

El caso más extremo de este fenómeno es EfficientNetV2 con CIFAR100, donde el modelo hiperbólico con pesos preentrenados obtiene apenas un 15.89 % en $d = 4$, prácticamente idéntico al resultado sin preentrenamiento (15.18 %), mientras que el euclidiano pasa de 4.99 % a 50.49 %. Esto indica que el modelo hiperbólico no logra aprovechar el conocimiento codificado en los pesos preentrenados. Una explicación posible es que las representaciones internas de EfficientNetV2 están fuertemente optimizadas para el espacio euclidiano; la proyección al disco de Poincaré mediante la capa ToPoincare introduce distorsiones geométricas que degradan la calidad del *embedding*, impidiendo que el modelo se beneficie del preentrenamiento.

4.7. ANÁLISIS EXPLORATORIO: RESPUESTA A ENTRENAMIENTO DE HIPERBÓLICAS Matrioshka EMBEDDINGS

La contraparte más notable es ResNet50 con Imagenette, donde el modelo hiperbólico no solo mantiene su ventaja, sino que alcanza el mejor resultado de todo el estudio (97.27% en $d = 32$). Este resultado muestra que, bajo ciertas condiciones, la geometría hiperbólica puede potenciar las representaciones preentrenadas en lugar de verse degradada por ellas. A diferencia de EfficientNetV2, ResNet50 es una arquitectura más simple cuyas representaciones internas parecen poseer mayor compatibilidad con el espacio hiperbólico, incluso tras el preentrenamiento.

Esto nos permite refinar nuestra hipótesis: la superioridad de los *embeddings* hiperbólicos en bajas dimensiones dentro del marco Matrioshka se sostiene de forma más robusta cuando el extractor entrena desde cero, siendo máxima bajo las restricciones más severas de dimensionalidad. Con pesos preentrenados, el beneficio se vuelve situacional, condicionado por la arquitectura del extractor. Esto sugiere que el diseño de Matrioshka *embeddings* hiperbólicos debe considerar la compatibilidad entre la arquitectura del extractor y la geometría del espacio de *embedding*.

Capítulo 5

Extensiones a Recuperación de Imágenes

Este capítulo presenta dos trabajos realizados en el marco de los *embeddings* hiperbólicos, con el objetivo de explorar la geometría hiperbólica más allá del contexto de los *Matrioshka Embeddings*. El primero corresponde a un trabajo publicado [25] en el que se investiga la capacidad de los *embeddings* de Poincaré para recuperar relaciones jerárquicas latentes en conjuntos de datos de imágenes sin supervisión directa, con aplicaciones en recuperación de imágenes basada en contenido. El segundo explora la incorporación de geometría hiperbólica en el marco de *DeepCluster*, un método de aprendizaje auto-supervisado basado en clustering, cuyo resultado fue negativo y cuyo análisis permite extraer conclusiones relevantes sobre las condiciones necesarias para el uso exitoso de la geometría hiperbólica.

5.1. Recuperación de Imágenes con Embeddings Hiperbólicos

Los trabajos [1,18] sugieren que los clasificadores hiperbólicos son capaces de aprender la jerarquía latente del conjunto de datos incluso si son entrenados únicamente con las clases hoja de la jerarquía. Esto quiere decir que, en un escenario donde las relaciones jerárquicas entre objetos no se especifican explícitamente durante el entrenamiento, se espera que los *embeddings* hiperbólicos superen a los euclidianos en el descubrimiento de jerarquías, especialmente cuando la dimensionalidad del espacio de *embeddings* se ve reducida. El trabajo [25] se centró en verificar esta hipótesis sobre datos de imágenes.

Para esto, se entrenaron y evaluaron redes convolucionales tanto euclidianas como hiperbólicas, utilizando VGG16 [28] preentrenado en ImageNet como arquitectura base. La variante hiperbólica es de naturaleza híbrida: las capas anteriores a las dos últimas operan en el espacio euclidiano, mientras que la penúltima capa proyecta las representaciones al disco de Poincaré mediante la capa *ToPoincaré*, y la

capa final es una Regresión Logística Multinomial Hiperbólica (HMLR). La variante euclidiana utiliza la misma arquitectura, reemplazando estas dos últimas capas por su equivalente euclidiano estándar con *softmax*. Se emplearon los conjuntos de datos CIFAR10 y CIFAR100, cuyas clases siguen una estructura jerárquica verificable [12, 32].

5.1.1. Métrica de Evaluación: HmAP

La evaluación del desempeño en recuperación de imágenes se realizó mediante una extensión jerárquica del *mean Average Precision* (mAP), denominada HmAP (*Hierarchy-Aware mean Average Precision*). La motivación detrás de esta métrica es que, en conjuntos de datos con estructura jerárquica, los errores de recuperación no son equivalentes: confundir una imagen de un “camión” con una de un “auto” es un error menos severo que confundirla con una de un “perro”, pues los primeros comparten un ancestro común en la jerarquía.

Formalmente, dado un conjunto de N consultas, la métrica $\text{mAP}@k$ estándar se define como el promedio de las precisiones medias individuales:

$$\text{mAP}@k = \frac{1}{N} \sum_{i=1}^N \text{AP}_i, \quad \text{donde} \quad \text{AP}@k = \frac{1}{\min(m_q, k)} \sum_{i=1}^k \text{P}@i \cdot R(q, d_i). \quad (5.1)$$

Aquí, $\text{P}@i$ denota la precisión en la posición i , m_q es el número total de imágenes relevantes para la consulta q , y $R(q, d_i)$ es una función de relevancia, normalmente binaria.

Para incorporar el componente jerárquico, se extiende la función de relevancia $R(q, d_i)$ considerando los distintos niveles $\ell = 1, 2, \dots, L$ de la jerarquía de clases. Para un nivel ℓ dado, se define $R(q, d_i) = 1$ si y solo si la clase de d_i y la clase de q comparten un ancestro común en el nivel ℓ de la jerarquía; en caso contrario, $R(q, d_i) = 0$. De este modo, la métrica $\text{HmAP}@k$ se evalúa en múltiples niveles de granularidad, donde niveles más altos corresponden a conceptos más específicos (clases hoja) y niveles más bajos a conceptos más abstractos (superclases).

Por ejemplo, en CIFAR10, a una granularidad de nivel 3 (el más específico) solo se considera relevante una imagen si comparte exactamente la misma clase que la consulta, equivalente al mAP tradicional. A granularidad de nivel 2, una imagen de clase “automóvil” se considera relevante para una consulta de clase “camión”, pues ambas pertenecen a la misma superclase de vehículos terrestres.

5.1.2. Experimentos

Los modelos se entrenaron en una tarea de clasificación estándar, sin proporcionar explícitamente las relaciones jerárquicas al modelo. Los datos se dividieron en conjuntos de entrenamiento, validación y prueba; la partición de validación se obtuvo aplicando una división 80:20 sobre los datos de entrena-

miento, y para la evaluación se utilizó la partición de prueba estándar de cada conjunto de datos. Como aumentación de datos se aplicaron rotaciones horizontales y verticales aleatorias, y todas las particiones se normalizaron según la media y desviación estándar del conjunto de datos correspondiente.

Para evaluar el efecto de la dimensionalidad en la capacidad del modelo de capturar jerarquías latentes, se varió el tamaño del *embedding* modificando el número de neuronas de la penúltima capa, tomando valores de 2 a 64 en potencias de dos. Todos los modelos se entrenaron durante 200 épocas con el optimizador Adam [19] y una tasa de aprendizaje inicial de 0.001, reduciéndose en un factor de 10 si la pérdida de validación no mejoraba en 10 épocas. Se implementó *early stopping* con una paciencia de 15 épocas. La curvatura hiperbólica c se fijó en 0.0005, valor determinado experimentalmente mediante búsqueda logarítmica.

Una vez entrenados los modelos, se evaluó la recuperación de imágenes mediante HmAP@10. La partición de prueba se dividió en una proporción 90:10, donde el 10 % actuó como conjunto de consultas y el 90 % restante como galería de búsqueda. La similitud entre *embeddings* se calculó mediante la distancia euclidiana para los modelos euclidianos, y mediante la distancia geodésica del disco de Poincaré para los modelos hiperbólicos. Se midió HmAP@10 en todos los niveles de la jerarquía: desde granularidad 3 (conceptos específicos) hasta granularidad 1 (conceptos amplios) en CIFAR10, y desde granularidad 5 hasta granularidad 1 en CIFAR100.

5.1.3. Resultados

Los resultados se resumen en las Tablas 5.1 y 5.2 para CIFAR10 y CIFAR100, respectivamente.

Tabla 5.1: HmAP@10 para modelos euclidiano (E) e hiperbólico (H) entrenados en CIFAR10, a diferentes niveles de granularidad.

Dim. Embedding	Granularidad 3		Granularidad 2		Granularidad 1	
	H	E	H	E	H	E
2	0.7681	0.7350	0.8016	0.8335	0.8682	0.8570
4	0.8016	0.7828	0.8653	0.8526	0.8871	0.8724
8	0.8018	0.7759	0.8707	0.8559	0.8878	0.8768
16	0.7765	0.7992	0.8565	0.8704	0.8822	0.8897
32	0.8049	0.8014	0.8734	0.8682	0.8950	0.8935
64	0.8023	0.7976	0.8689	0.8620	0.8864	0.8794

Para CIFAR10, los *embeddings* hiperbólicos superan a los euclidianos en la mayoría de las configuraciones y niveles de jerarquía, aunque la diferencia no es concluyente. Esto puede atribuirse a que el número de clases en CIFAR10 no es suficiente para estresar la capacidad del espacio euclidiano, por lo que la ventaja del espacio hiperbólico no se manifiesta con claridad.

En CIFAR100, en cambio, los resultados son marcadamente más favorables para la geometría hiperbólica. Los *embeddings* hiperbólicos superan a los euclidianos por un margen considerable en los

Tabla 5.2: HmAP@10 para modelos euclidiano (E) e hiperbólico (H) entrenados en CIFAR100, a diferentes niveles de granularidad.

	Granularidad 5		Granularidad 4		Granularidad 3		Granularidad 2		Granularidad 1	
Dim. Embedding	H	E	H	E	H	E	H	E	H	E
2	0.1338	0.0829	0.3130	0.2502	0.4934	0.5005	0.7508	0.7515	0.8235	0.8344
4	0.2804	0.1810	0.4766	0.3800	0.6548	0.5791	0.8315	0.7804	0.8898	0.8596
8	0.3389	0.2514	0.5023	0.4469	0.6720	0.6366	0.8430	0.8198	0.8860	0.8721
16	0.3504	0.3366	0.5251	0.5037	0.6795	0.6640	0.8395	0.8302	0.8784	0.8850
32	0.4155	0.3860	0.5759	0.5472	0.7194	0.6923	0.8677	0.8506	0.9006	0.8921
64	0.4029	0.3407	0.5634	0.5101	0.7106	0.6645	0.8587	0.8381	0.8998	0.8858

niveles más específicos de la jerarquía, y esta ventaja se mantiene consistentemente a través de todas las dimensiones de *embedding* evaluadas. Adicionalmente, las diferencias tienden a reducirse a medida que aumenta la dimensionalidad del *embedding*, lo cual es coherente con la teoría: al disponer de mayor capacidad representacional, el espacio euclidiano deja de estar tan limitado y puede aproximar mejor la estructura jerárquica subyacente.

Estos resultados confirman que la geometría hiperbólica favorece la organización de las representaciones en concordancia con la estructura jerárquica inherente a los datos, y que este efecto se vuelve especialmente pronunciado cuando el número de clases es alto y la dimensionalidad del espacio de representación es reducida, condiciones en las que el espacio euclidiano enfrenta mayores dificultades para acomodar estructuras de tipo árbol.

5.2. Hyperbolic DeepCluster

Dentro de esta sección se presentan los resultados de un segundo trabajo realizado en el marco de los *embeddings* hiperbólicos, cuyo objetivo central fue evaluar la viabilidad y el beneficio de incorporar geometría hiperbólica en el marco de *DeepCluster*.

En su formulación original, DeepCluster [3] alterna iterativamente entre dos pasos: la asignación de pseudo-etiquetas mediante clustering sobre los *embeddings* producidos por la red, y la actualización de los parámetros del modelo utilizando dichas pseudo-etiquetas como señal de supervisión. La propuesta explorada en este trabajo consiste en reemplazar el espacio euclidiano del *embedding* por el disco de Poincaré, evaluando diferentes estrategias para el paso de clustering.

La motivación central de esta propuesta radica en la naturaleza jerárquica implícita que emerge del proceso de clustering. A medida que el entrenamiento progresa, los clusters descubiertos no son categorías arbitrarias, sino agrupaciones que reflejan regularidades semánticas y visuales presentes en los datos. En datasets con estructura taxonómica explícita, como CIFAR100, donde las clases se organizan en superclases semánticamente, es razonable esperar que esta estructura jerárquica latente se manifieste en la geometría del espacio de representación. El espacio hiperbólico, que ya sabemos permite embe-

ber jerarquías con distorsión mínima, constituye en este contexto una geometría potencialmente más adecuada que el espacio euclidiano para dichas representaciones.

A esto se suma el argumento de eficiencia en baja dimensión. Como se demostró en el capítulo anterior, el espacio de Poincaré ofrece una capacidad representacional significativamente mayor por dimensión que su contraparte euclidiana, lo que sugiere que *embeddings* hiperbólicos de dimensión moderada podrían capturar con mayor fidelidad la estructura semántica de los datos, redundando en pseudo-etiquetas de mayor calidad y, en consecuencia, en una señal de entrenamiento más informativa.

La hipótesis que guía estos experimentos es que la geometría hiperbólica favorece la organización de las representaciones en concordancia con la estructura jerárquica inherente a los datos, y que este efecto se traduce en una mejora del rendimiento del método en las métricas estándar de evaluación.

5.2.1. Arquitectura

Para los experimentos se definen dos variantes del modelo: euclidiana e hiperbólica. Ambas comparten la misma arquitectura base, con la diferencia de que la variante hiperbólica incorpora una capa ToPoincaré al final del extractor de características.

Como extractores de características se utilizan ResNet50 y EfficientNetV2-S, los mismos empleados en los experimentos de *Matrioshka Embeddings*. El flujo del modelo es el siguiente: la imagen se procesa a través del extractor de características, obteniendo un *embedding* que se utiliza tanto para el clustering como para el entrenamiento supervisado con las pseudo-etiquetas generadas.

Para el modelo euclidiano, el clustering se realiza mediante k -means estándar implementado con FAISS [17]. Para el modelo hiperbólico, se exploraron dos estrategias de clustering:

- **Fréchet k -means:** una extensión del algoritmo k -means al espacio hiperbólico que utiliza la media de Fréchet como centroide.
- **Proyección + k -means euclidiano:** los *embeddings* hiperbólicos se proyectan de vuelta al espacio euclidiano mediante la capa FromPoincaré (mapa logarítmico) y luego se aplica k -means estándar con FAISS.

Siguiendo la propuesta original de DeepCluster, el número de clusters se define como $K \times 10$, donde K es el número de clases del dataset. Adicionalmente, se evaluó la configuración con K clusters (igual al número de clases).

5.2.2. Entrenamiento

Cada modelo se entrena por un máximo de 100 épocas con un batch size de 64, utilizando el optimizador AdamW con un *learning rate* inicial de 0.001 y weight decay de 0.0001. Adicionalmente, se

implementa *early stopping* con una paciencia de 10 épocas.

Como única técnica de aumentación de datos se utiliza `RandomHorizontalFlip`. Las pseudo-etiquetas se recalculan al final de cada época mediante el algoritmo de clustering correspondiente. En los modelos hiperbólicos, se evaluaron curvaturas iniciales de $c = 0.05$ y $c = 1.0$, tanto con curvatura fija como optimizable durante el entrenamiento.

5.2.3. Evaluación

El rendimiento de los modelos se evalúa mediante las siguientes métricas:

- **NMI (Normalized Mutual Information)**: mide la concordancia entre las asignaciones de clusters y las etiquetas reales, normalizada entre 0 y 1.
- **NMI Successive**: mide la estabilidad de las asignaciones entre épocas consecutivas.
- **Reassignment Rate**: proporción de muestras que cambian de cluster entre épocas consecutivas.
- **Largest Cluster Ratio**: proporción de muestras asignadas al cluster más grande, útil para detectar degeneración.

La selección del mejor modelo se realiza según el NMI sobre el conjunto de validación.

5.2.4. Resultados

A diferencia de los resultados positivos obtenidos en los experimentos de recuperación de imágenes, el modelo hiperbólico presentó severos problemas de estabilidad que impidieron su entrenamiento exitoso. En todas las configuraciones evaluadas, los *embeddings* degeneraron rápidamente hacia un único cluster durante las primeras épocas de entrenamiento, colapsando el proceso de aprendizaje.

La Tabla 5.3 resume las variaciones de hiperparámetros exploradas en el intento de estabilizar el modelo hiperbólico.

Tabla 5.3: Variaciones de hiperparámetros evaluadas para el modelo hiperbólico.

Hiperparámetro	Valores probados
Arquitectura	ResNet50, EfficientNetV2-S
Dimensión embedding	64, 256
Número de clusters	K (igual a clases), $10K$
Método de clustering	Fréchet k -means, FromPoincaré + k -means
Curvatura	Fija, optimizable
Recorte de gradiente	True, False

Ninguna de las configuraciones evaluadas logró evitar la degeneración a un único cluster.

Un ejemplo de los resultados de entrenamiento puede encontrarse en la Figura 5.1, donde se observa que el *Largest Cluster Ratio* alcanza rápidamente el 100 %, a la vez que *Active Clusters* cae drásticamente a 0, mostrando la degeneración de la solución para el caso hiperbólico. La totalidad de las imágenes de entrenamiento puede encontrarse en el Anexo B.

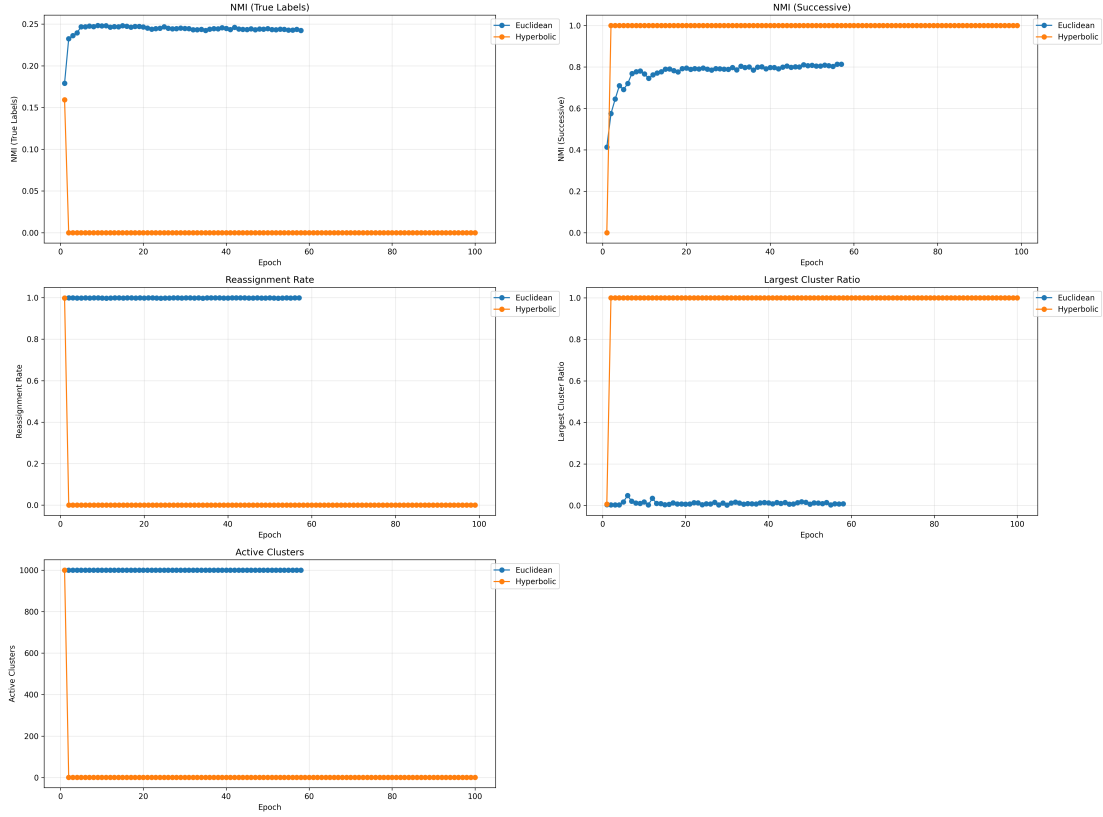


Figura 5.1: Resultados entrenamiento EfficientNetV2 sobre CIFAR100.

Análisis del Fallo

La degeneración observada en los modelos hiperbólicos puede atribuirse a varios factores:

- **Inestabilidad del clustering hiperbólico:** el algoritmo de Fréchet k -means es sensible a la distribución inicial de los puntos en el espacio hiperbólico. Si los *embeddings* comienzan concentrados en una región del disco de Poincaré, el algoritmo puede converger a una solución trivial con un único cluster.
- **Interacción entre geometría y gradientes:** en DeepCluster, las pseudo-etiquetas generadas por el clustering definen la señal de supervisión. Cuando el clustering falla, las pseudo-etiquetas resultantes no proporcionan información discriminativa, lo que a su vez produce *embeddings* poco diferenciados, creando un ciclo de retroalimentación negativa que colapsa el modelo.

- **Ausencia de regularización:** a diferencia del trabajo de recuperación de imágenes, donde la función de pérdida proporciona supervisión directa sobre clases conocidas, DeepCluster depende enteramente de la calidad de las pseudo-etiquetas. Esta dependencia hace al método más vulnerable a la degeneración cuando la geometría del espacio de *embedding* no está adecuadamente regularizada.

Estos resultados sugieren que la incorporación de geometría hiperbólica en métodos de aprendizaje auto-supervisado basados en clustering requiere mecanismos adicionales de estabilización que no fueron explorados en este trabajo, tales como esquemas de *warm-up*, regularización explícita de la distribución de *embeddings*, o algoritmos de clustering hiperbólico más robustos.

Una dirección particularmente interesante es utilizar el modelo de Lorentz (o del hiperboloide) en lugar del disco de Poincaré. Aunque ambos son representaciones isométricas del espacio hiperbólico, el disco de Poincaré presenta inestabilidades numéricas cuando los *embeddings* se aproximan a su frontera ($\|x\| \rightarrow 1$), ya que la métrica diverge en esa región, provocando gradientes de Riemannian de gran magnitud. El modelo de Lorentz, que opera sobre el hiperboloide del espacio de Minkowski, no posee este problema, haciendo que sus cálculos de distancias geodésicas y gradientes sean numéricamente estables en todo el dominio. Esto es especialmente relevante para el paso de clustering: la media de Fréchet en el modelo de Lorentz admite una solución de forma cerrada más simple y numéricamente robusta que su contraparte en el disco de Poincaré, lo que reduciría directamente la sensibilidad del algoritmo a la distribución inicial de los *embeddings*, identificada como factor clave del colapso observado.

Capítulo 6

Conclusiones

6.1. Conclusiones Generales

Este trabajo investigó la incorporación de geometría hiperbólica en el aprendizaje de representaciones visuales, específicamente en las tareas de *Matrioshka Embeddings* y *Deep Clustering*. Los resultados obtenidos demuestran que los *embeddings* hiperbólicos ofrecen ventajas significativas sobre sus contrapartes euclidianas en contextos donde la dimensionalidad del espacio de representación es limitada, confirmando la hipótesis central de que el espacio hiperbólico constituye una alternativa viable y beneficiosa para representaciones de imágenes.

La principal contribución de este trabajo es la demostración empírica de que los *embeddings* hiperbólicos superan consistentemente a los euclidianos en *Matrioshka Embeddings* para dimensiones bajas ($d \leq 16$), alcanzando diferencias de hasta 70 puntos porcentuales en *accuracy* (CIFAR100 con ResNet50, $d = 4$). Esta ventaja se mantiene a través de diferentes arquitecturas y datasets, y es especialmente pronunciada en conjuntos de datos con estructura jerárquica compleja. Adicionalmente, se desarrolló HypTorch, una biblioteca de capas neuronales hiperbólicas para PyTorch que facilita la experimentación con geometrías no euclidianas en el contexto de visión por computador.

Por otro lado, los experimentos en Deep Clustering revelaron que la incorporación directa de geometría hiperbólica en métodos de aprendizaje auto-supervisado basados en clustering presenta desafíos significativos de estabilidad que no fueron resueltos en este trabajo. Este resultado negativo constituye en sí mismo una contribución, al evidenciar las limitaciones actuales de las técnicas de clustering hiperbólico y la necesidad de desarrollar métodos más robustos.

6.2. Conclusiones Específicas

A continuación se presenta el contraste de las hipótesis planteadas en este trabajo:

1. **Hipótesis 1** *Las representaciones hiperbólicas preservan una mayor cantidad de información útil para tareas downstream al reducir su dimensionalidad, en comparación con sus equivalentes euclidianas.*

Confirmada. Los resultados experimentales demuestran que los modelos hiperbólicos superan consistentemente a los euclidianos en las dimensiones más bajas del *embedding* ($d = 4, d = 8$). En casos extremos, como CIFAR100 con ResNet50, el modelo euclidiano no logra superar el 1 % de exactitud hasta $d = 64$, mientras que el hiperbólico alcanza un 22.96 % con $d = 4$. Esta ventaja se observa de manera consistente en ambas arquitecturas evaluadas (ResNet50 y EfficientNetV2-S) y en los tres datasets (CIFAR10, CIFAR100 e Imagenette).

2. **Hipótesis 2** *Los métodos de Deep Clustering que operan en el espacio hiperbólico producirán agrupaciones que respetan mejor las jerarquías semánticas naturales de los datos visuales.*

Rechazada. No fue posible evaluar esta hipótesis debido a los problemas de estabilidad encontrados durante el entrenamiento. En todas las configuraciones evaluadas, los *embeddings* hiperbólicos degeneraron hacia un único cluster durante las primeras épocas, impidiendo el proceso de aprendizaje. Este resultado indica que la integración de geometría hiperbólica en *Deep Clustering* requiere mecanismos de estabilización adicionales que no fueron explorados en este trabajo.

3. **Hipótesis 3** *La ventaja relativa de los embeddings hiperbólicos sobre los euclidianos será más pronunciada en datasets con mayor estructura jerárquica y cuando la dimensión del embedding es especialmente limitada.*

Confirmada. Los experimentos muestran una correlación clara entre la complejidad jerárquica del dataset y la magnitud de la ventaja hiperbólica. En CIFAR100 (100 clases, 20 superclases), la diferencia de rendimiento entre modelos hiperbólicos y euclidianos es considerablemente mayor que en CIFAR10 (10 clases, 2 superclases). Asimismo, la ventaja del espacio hiperbólico disminuye progresivamente a medida que aumenta la dimensión del *embedding*, llegando incluso a invertirse en algunas configuraciones de alta dimensión. Este patrón es consistente con la teoría de que el espacio hiperbólico ofrece mayor eficiencia representacional precisamente donde el espacio euclidiano presenta mayores limitaciones.

6.3. Trabajo Futuro

Los resultados y limitaciones de este trabajo sugieren las siguientes direcciones de investigación futura:

- **Estabilización de Deep Clustering hiperbólico:** Desarrollar mecanismos que permitan entrenar modelos de *Deep Clustering* en el espacio hiperbólico de manera estable. Esto podría incluir esquemas de *warm-up* donde el modelo se preentrena brevemente en el espacio euclidiano antes de transicionar al hiperbólico, regularización de la distribución de *embeddings* para evitar su concentración, o el desarrollo de algoritmos de *k*-means hiperbólico más robustos que sean menos sensibles a la inicialización.
- **Evaluación en datasets de mayor escala:** Validar los resultados obtenidos en conjuntos de datos más grandes y complejos, como ImageNet completo, para verificar si las ventajas observadas se mantienen a mayor escala y con estructuras jerárquicas más profundas.
- **Exploración de otras arquitecturas:** Extender los experimentos a arquitecturas más recientes como Vision Transformers (ViT), que han demostrado resultados superiores en diversas tareas de visión y podrían beneficiarse de manera diferente de la geometría hiperbólica.
- **Aplicación a otros dominios:** Explorar la transferencia de las metodologías desarrolladas a otros dominios donde las estructuras jerárquicas son relevantes, como el procesamiento de lenguaje natural, el aprendizaje multimodal (imagen-texto), o el análisis de grafos y redes sociales.

Apéndice A

Figuras de Entrenamiento

A continuación, se presentan todas las figuras obtenidas durante el entrenamiento de los modelos.

A.1. CIFAR10 - ResNet50

Las figuras a continuación presentan los resultados de entrenamiento por dimensión para cada modelo.

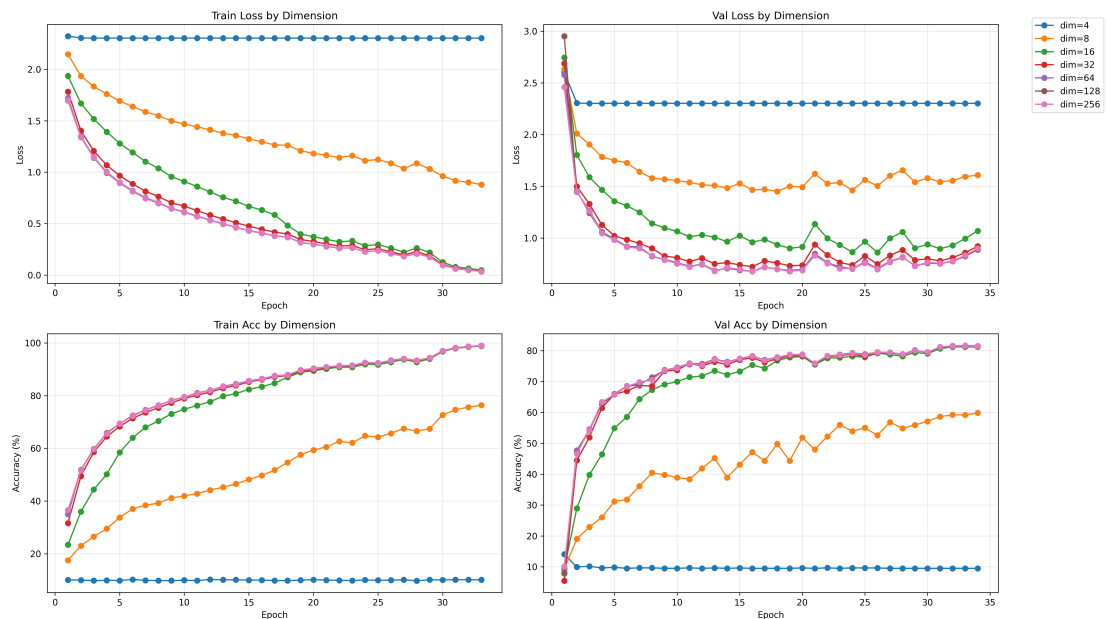


Figura A.1: Resultados entrenamiento ResNet50 euclidiano sobre CIFAR10.

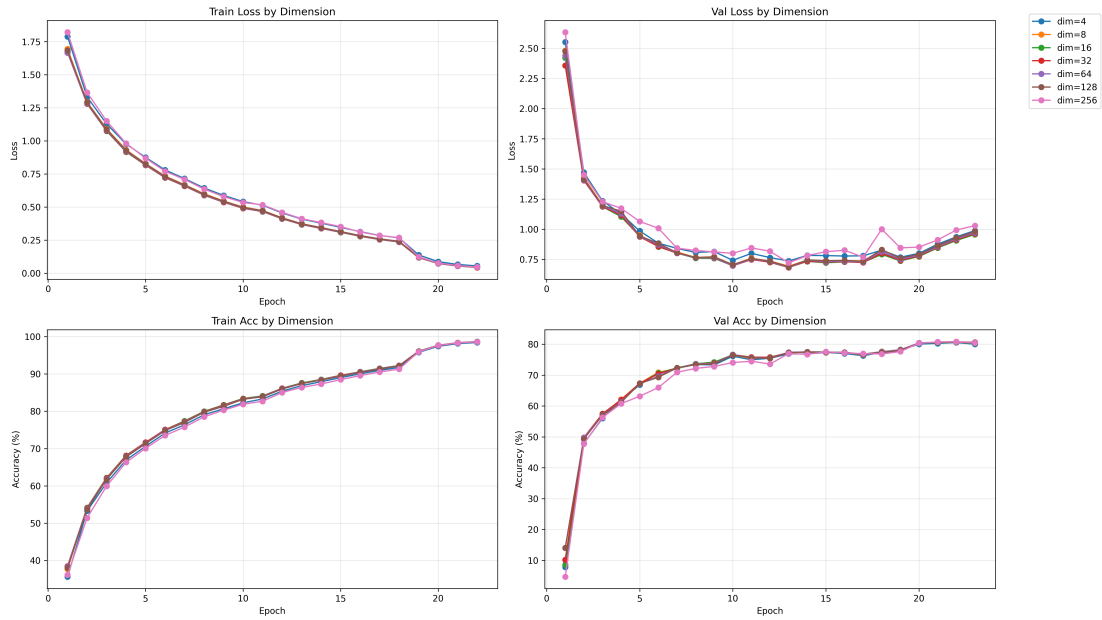


Figura A.2: Resultados entrenamiento ResNet50 hiperbólico sobre CIFAR10.

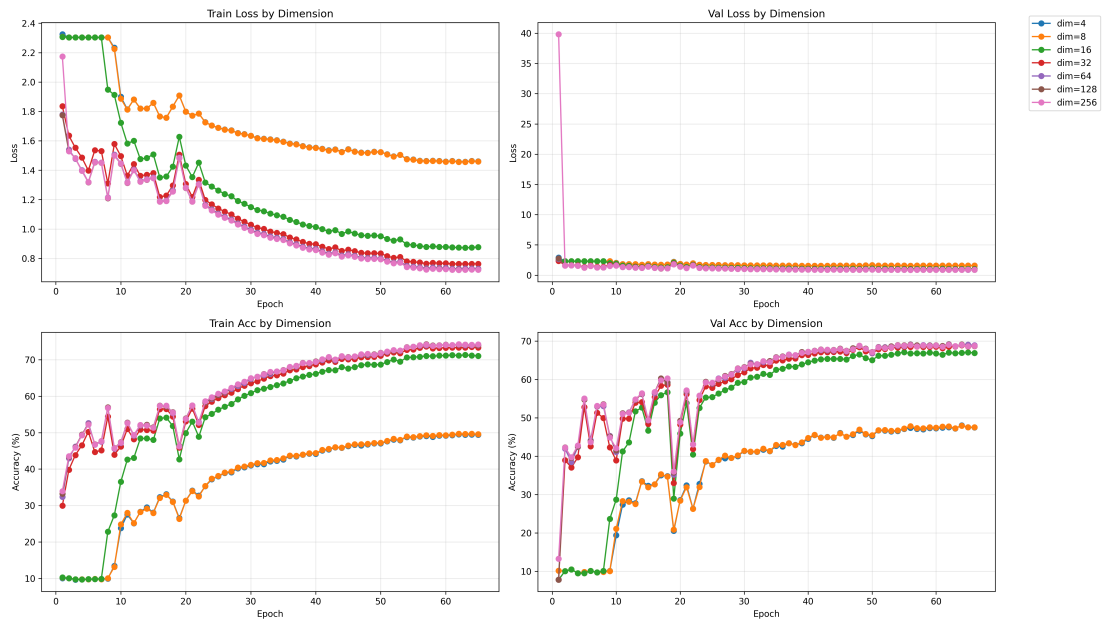


Figura A.3: Resultados entrenamiento ResNet50 híbrido sobre CIFAR10.

Las figuras a continuación comparan las pérdidas de entrenamiento y validación entre los tres modelos.

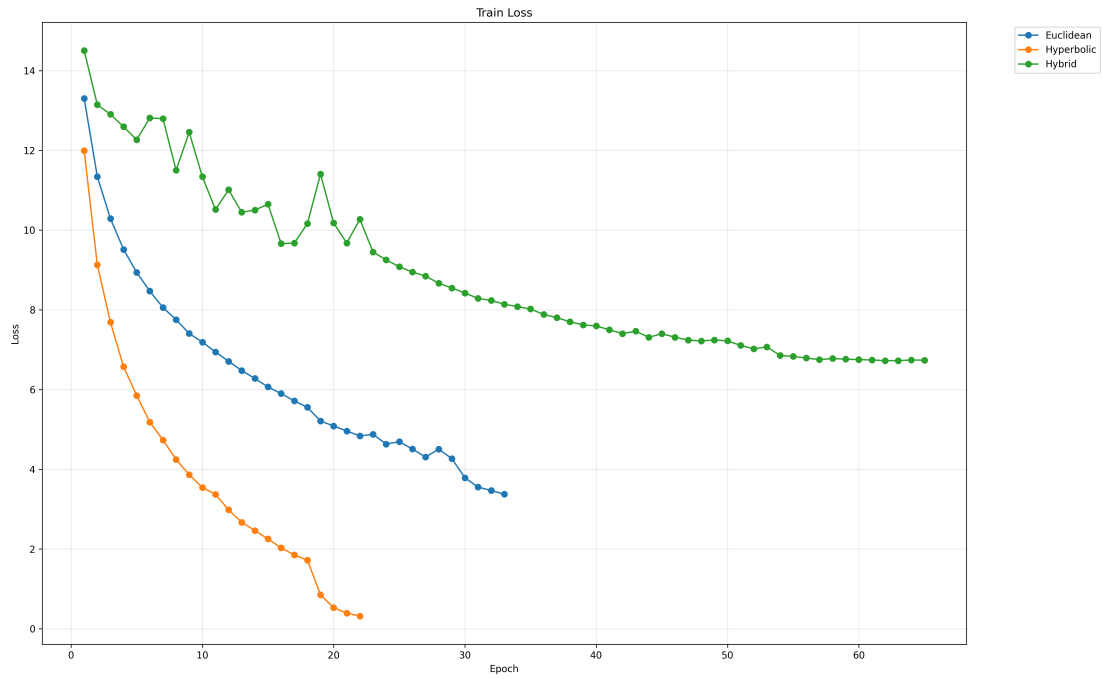


Figura A.4: Comparación entrenamiento ResNet50 sobre CIFAR10.

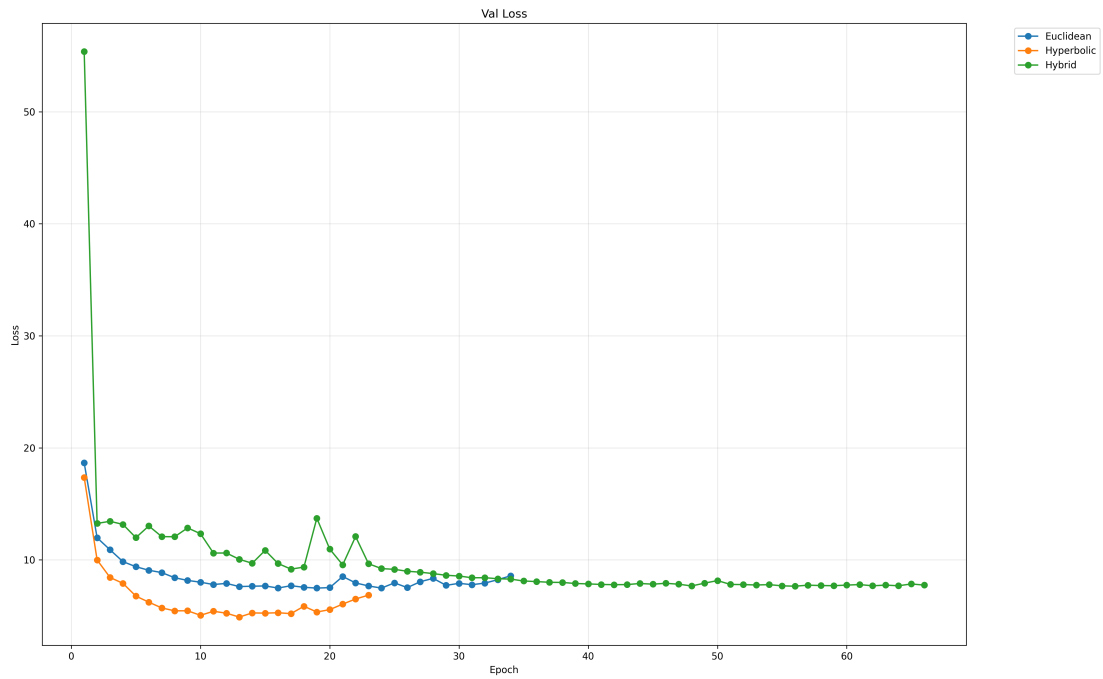
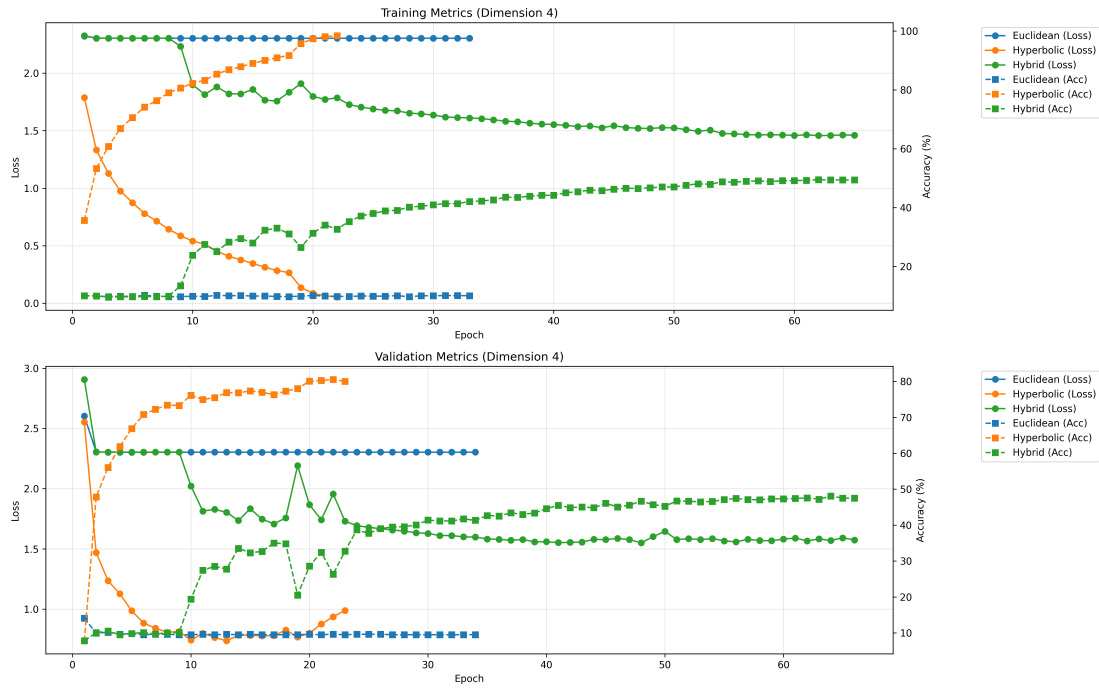
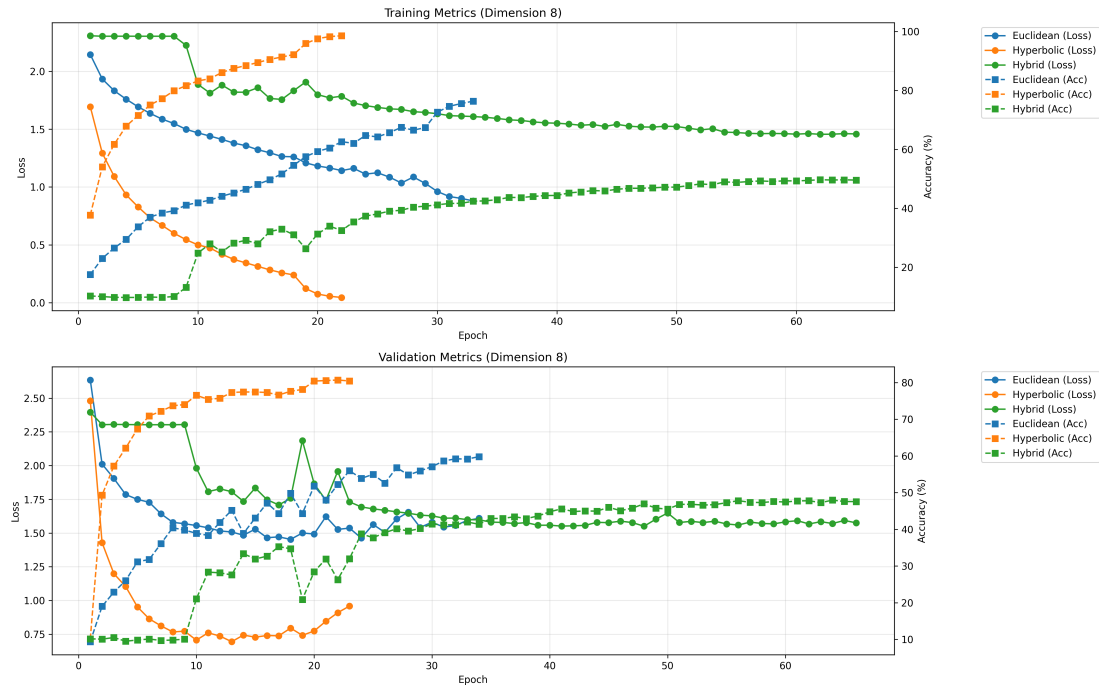
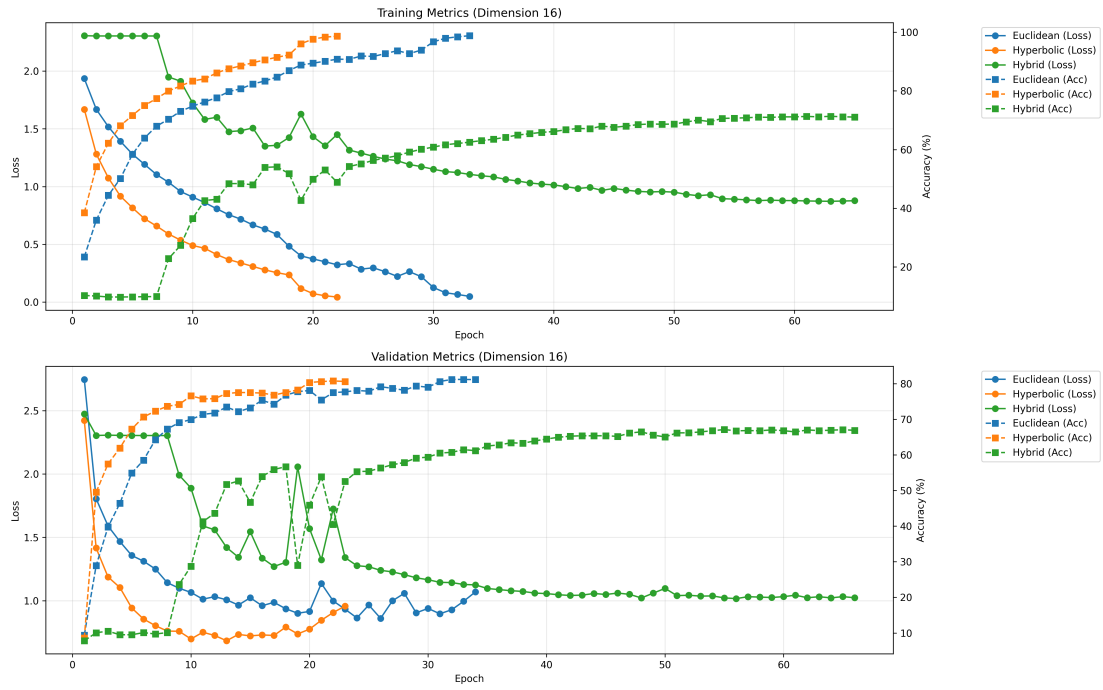
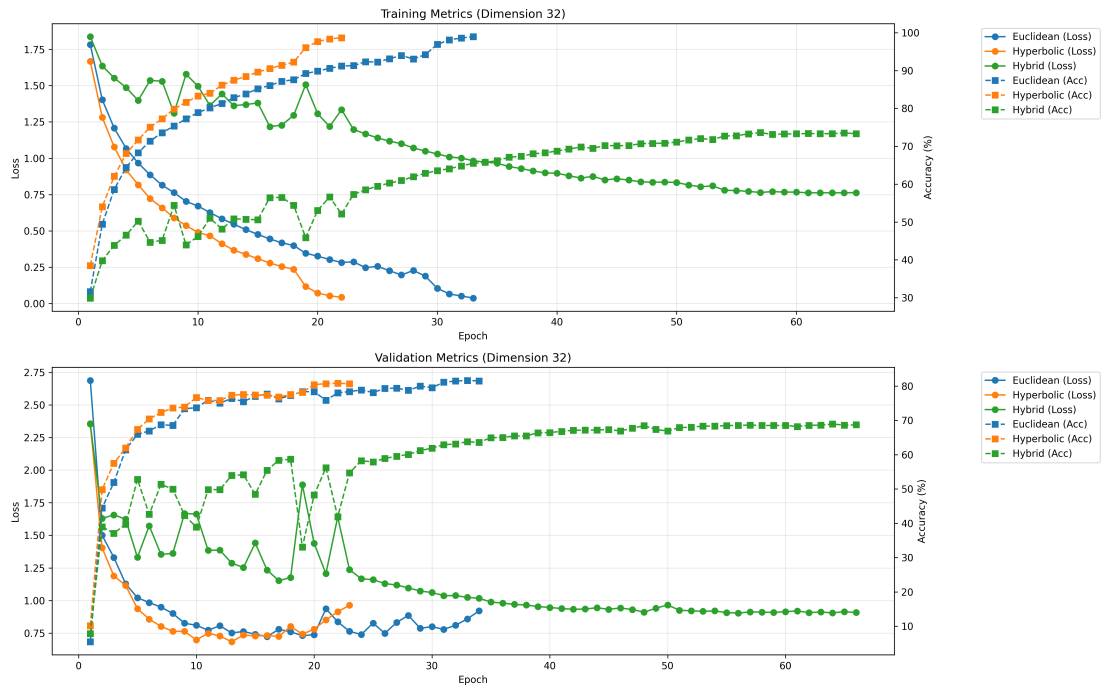
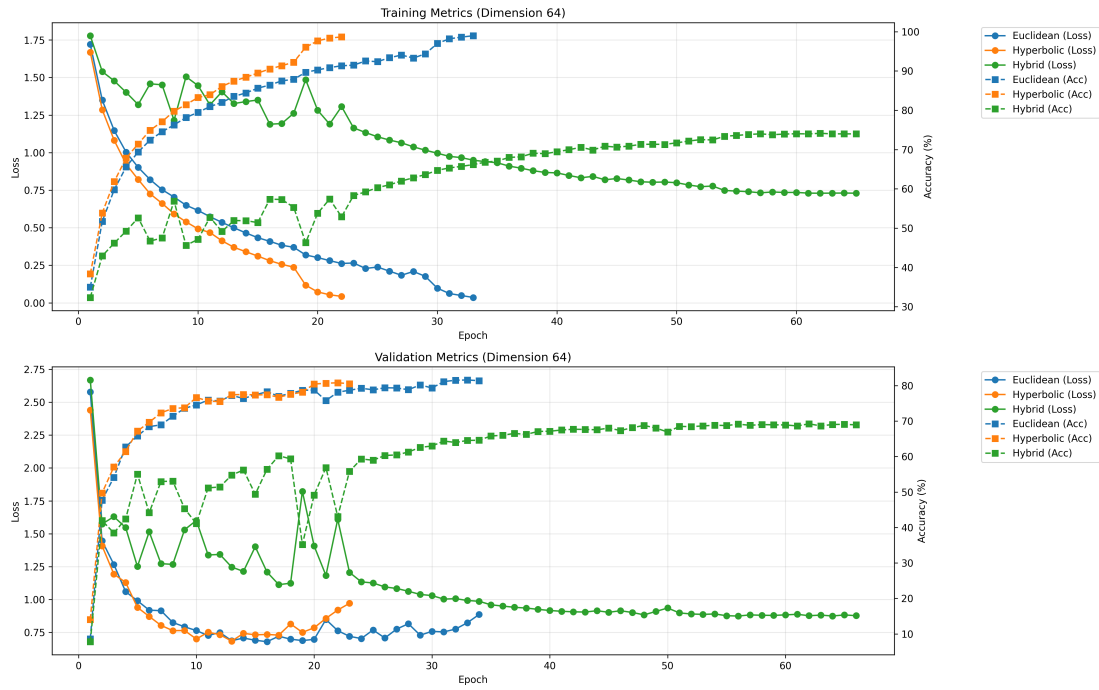
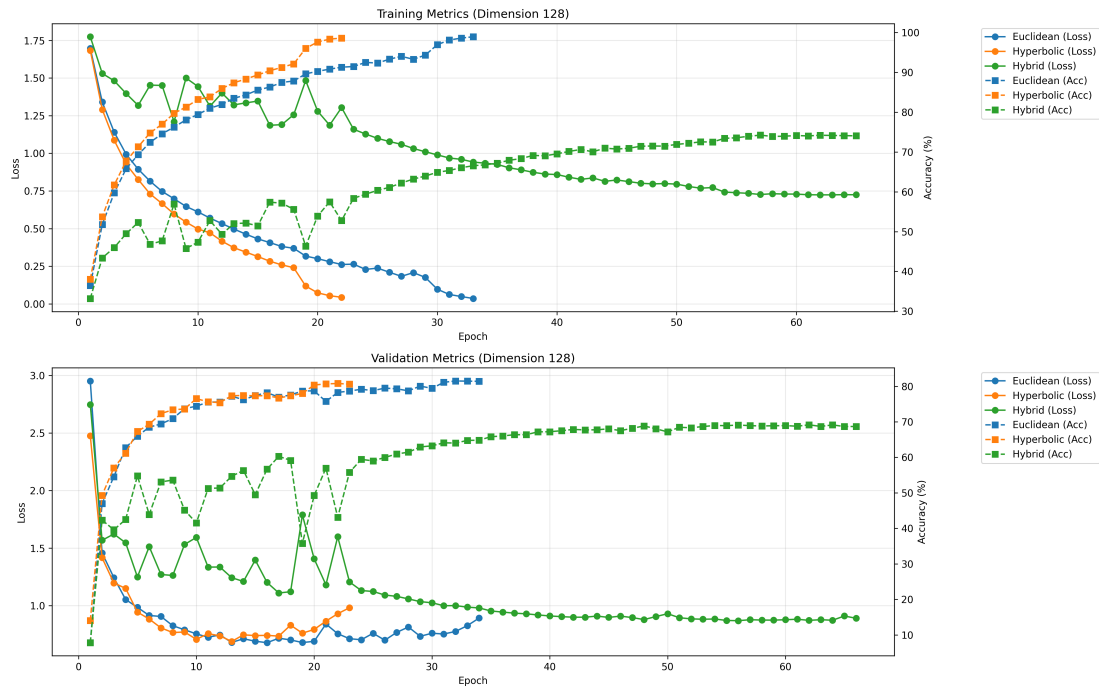


Figura A.5: Comparación validación ResNet50 sobre CIFAR10.

Las figuras a continuación presentan una comparación detallada de los tres modelos para cada dimensión.

Figura A.6: Comparación entrenamiento ResNet50 sobre CIFAR10 para $d = 4$.Figura A.7: Comparación entrenamiento ResNet50 sobre CIFAR10 para $d = 8$.

Figura A.8: Comparación entrenamiento ResNet50 sobre CIFAR10 para $d = 16$.Figura A.9: Comparación entrenamiento ResNet50 sobre CIFAR10 para $d = 32$.

Figura A.10: Comparación entrenamiento ResNet50 sobre CIFAR10 para $d = 64$.Figura A.11: Comparación entrenamiento ResNet50 sobre CIFAR10 para $d = 128$.

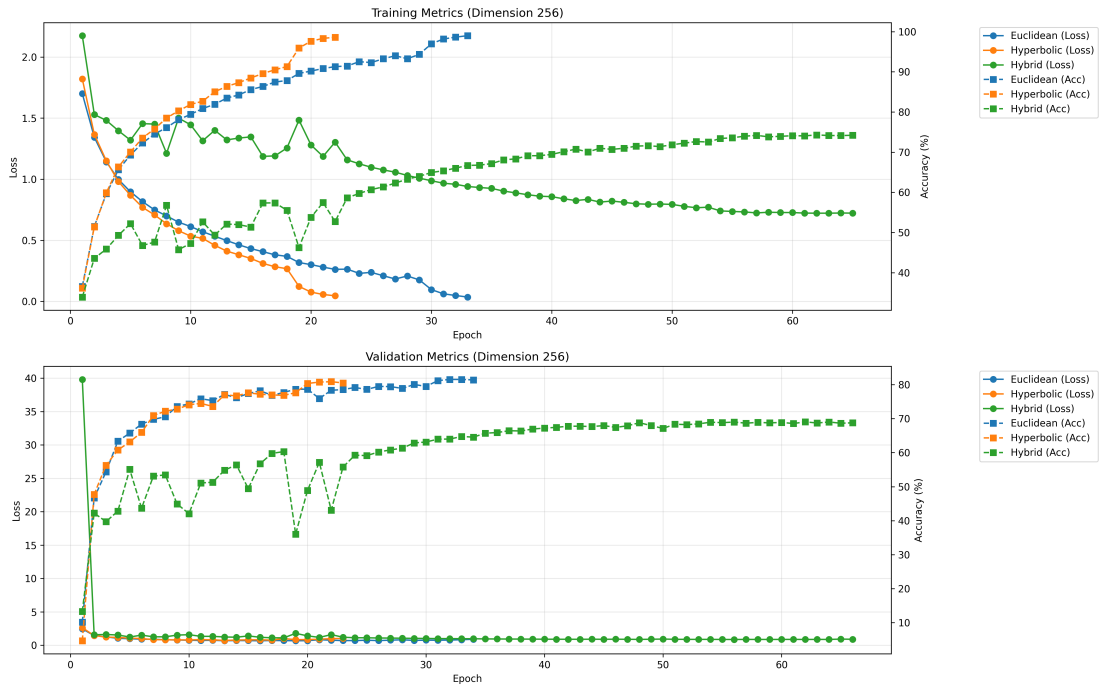


Figura A.12: Comparación entrenamiento ResNet50 sobre CIFAR10 para $d = 256$.

A.1.1. CIFAR10 - EfficientNetV2

Las figuras a continuación presentan los resultados de entrenamiento por dimensión para cada modelo.

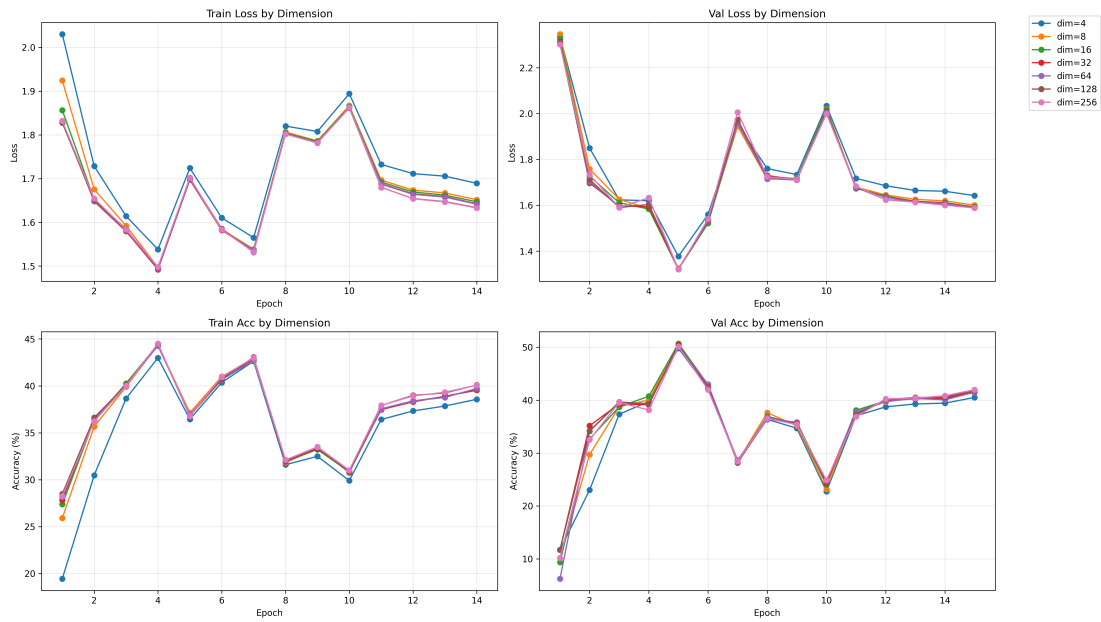


Figura A.13: Resultados entrenamiento EfficientNetV2 euclidiano sobre CIFAR10.

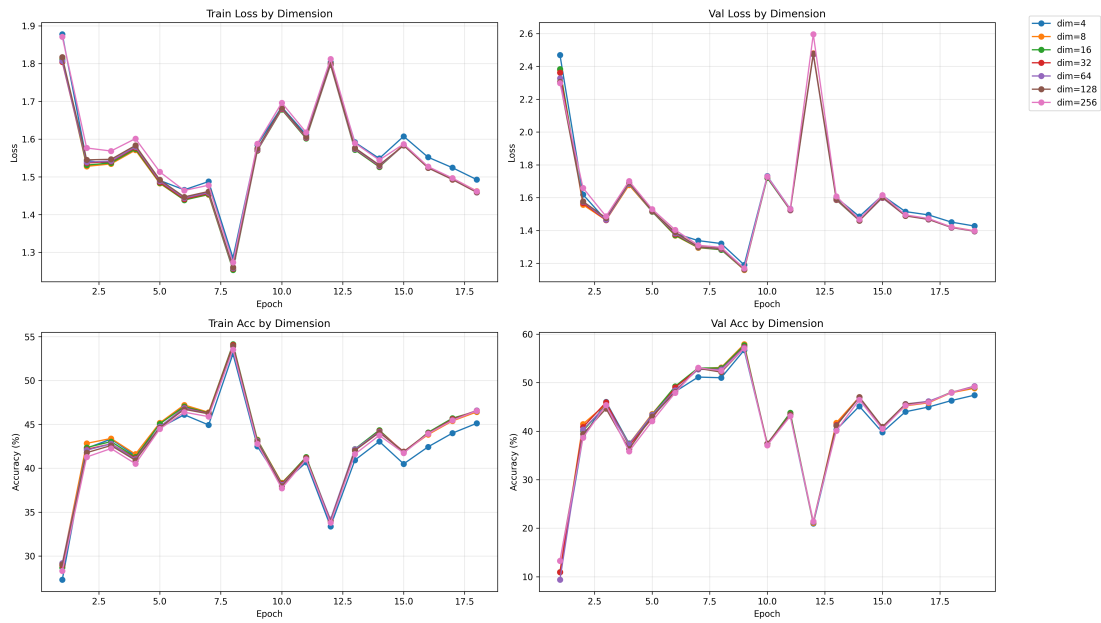


Figura A.14: Resultados entrenamiento EfficientNetV2 hiperbólico sobre CIFAR10.

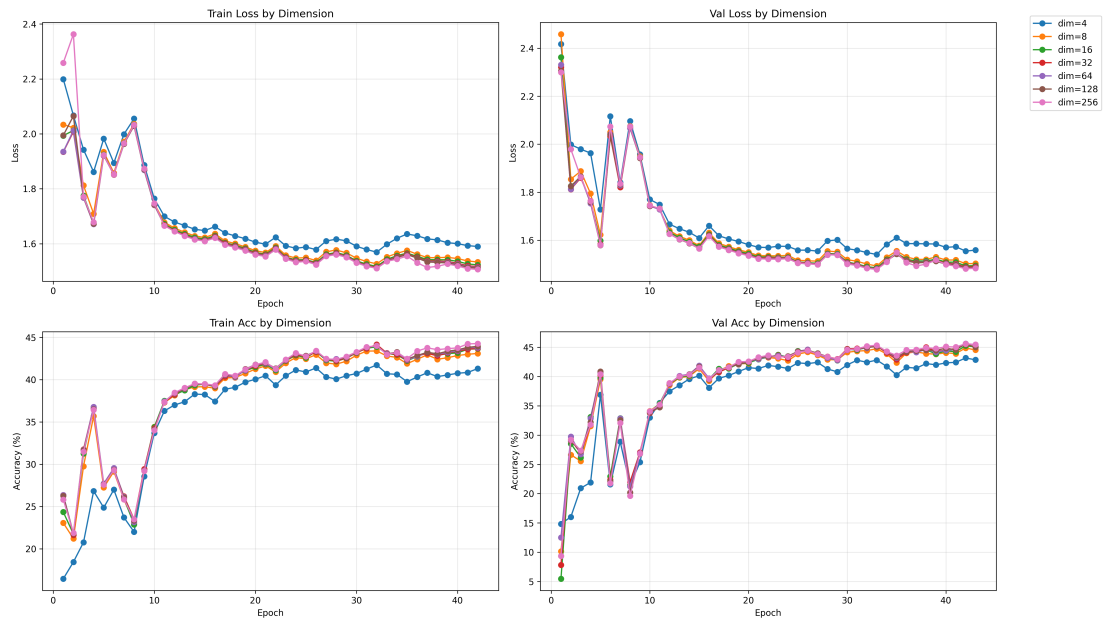


Figura A.15: Resultados entrenamiento EfficientNetV2 híbrido sobre CIFAR10.

Las figuras a continuación comparan las pérdidas de entrenamiento y validación entre los tres modelos.

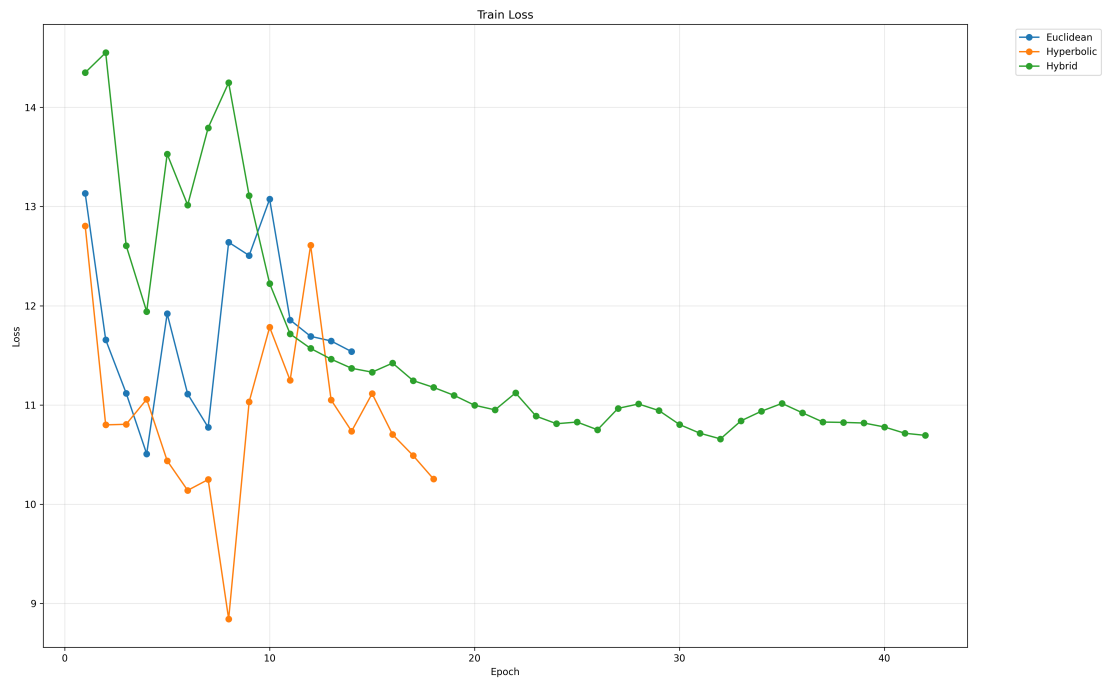


Figura A.16: Comparación entrenamiento EfficientNetV2 sobre CIFAR10.

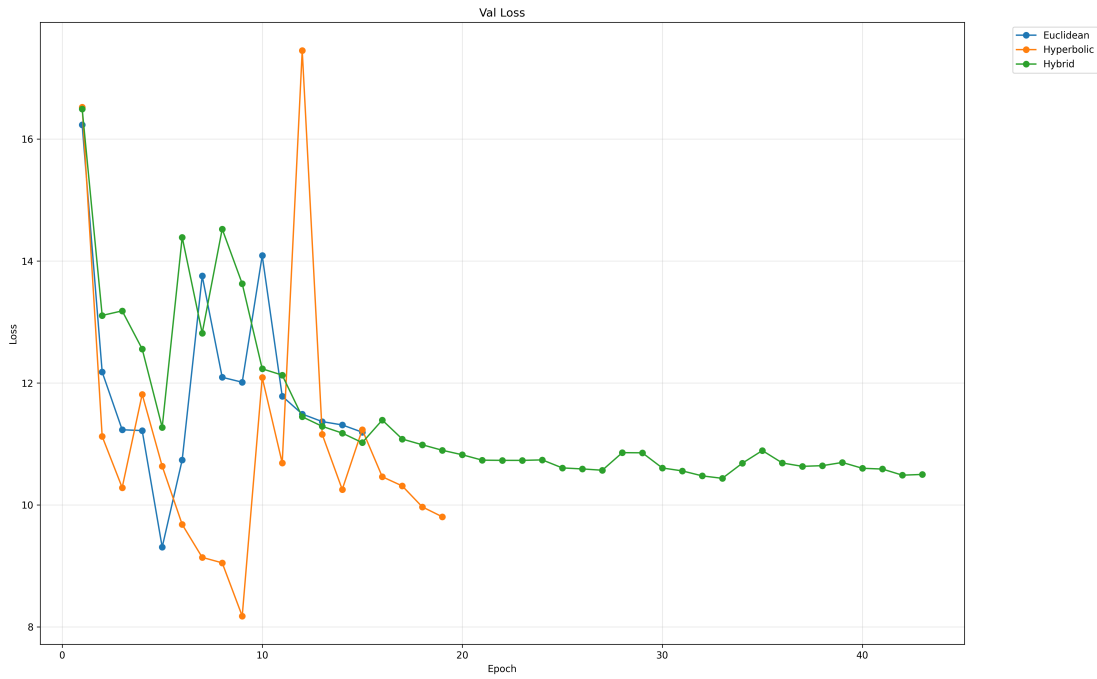
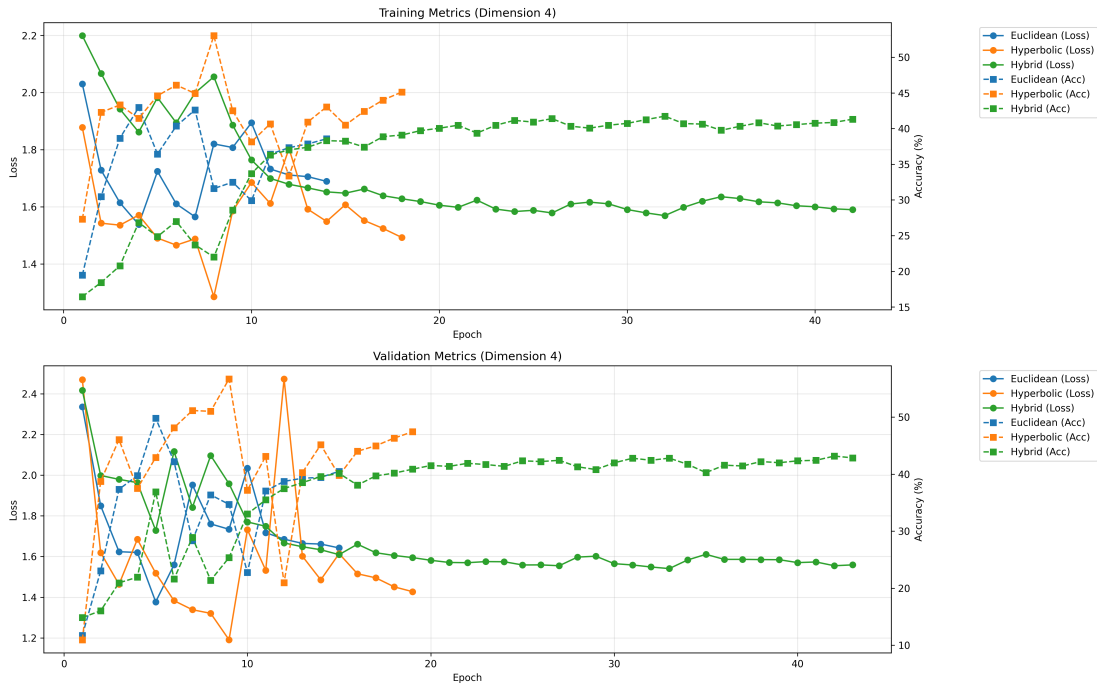
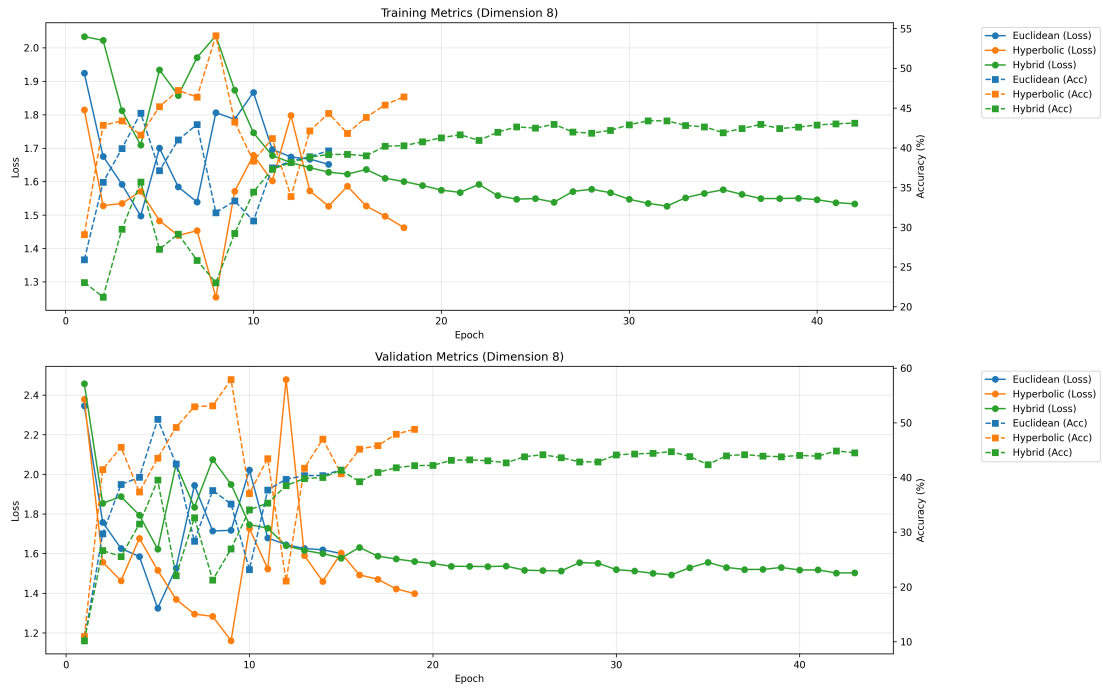
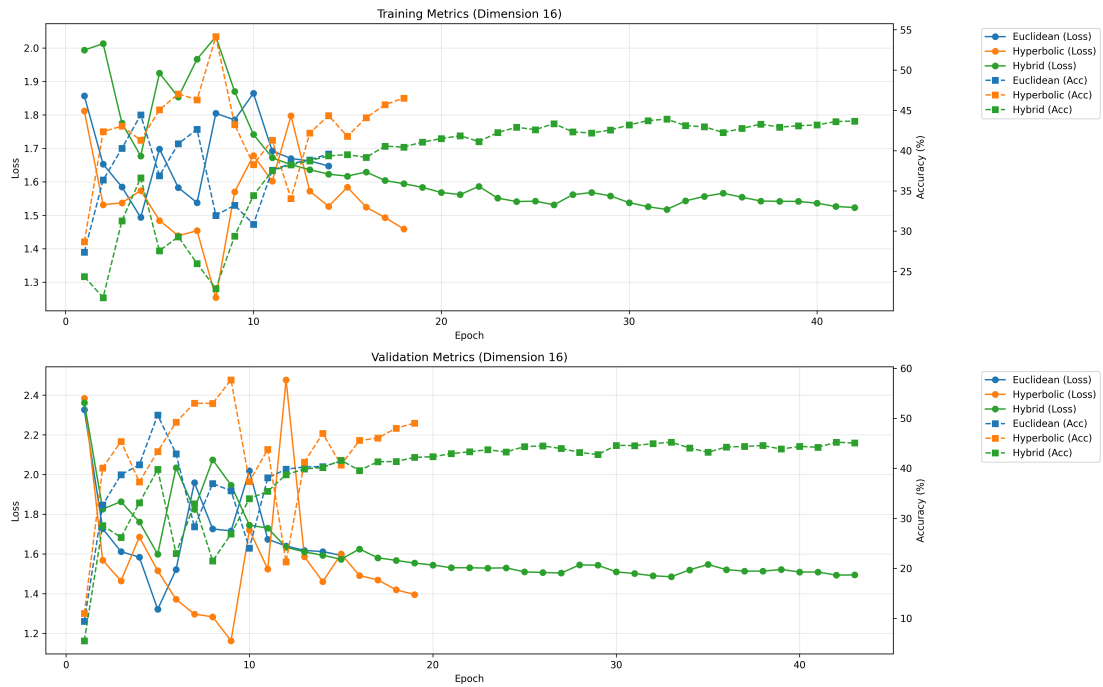


Figura A.17: Comparación entrenamiento EfficientNetV2 sobre CIFAR10.

Las figuras a continuación presentan la comparación detallada por dimensión.

Figura A.18: Comparación entrenamiento EfficientNetV2 sobre CIFAR10 para $d = 4$.

Figura A.19: Comparación entrenamiento EfficientNetV2 sobre CIFAR10 para $d = 8$.Figura A.20: Comparación entrenamiento EfficientNetV2 sobre CIFAR10 para $d = 16$.

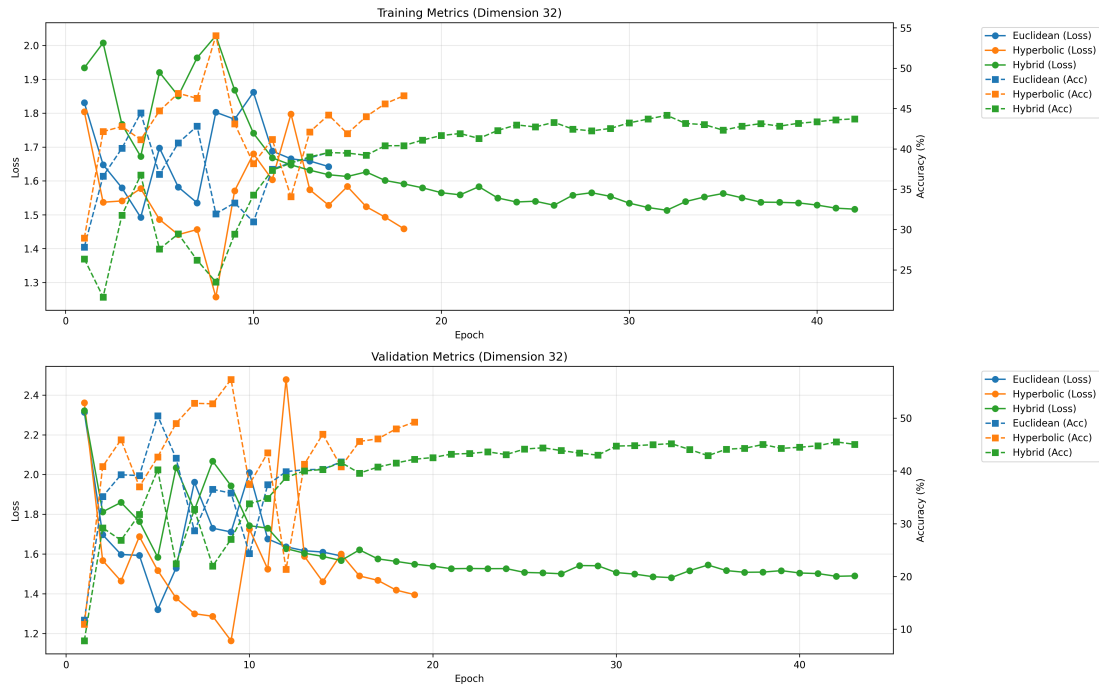


Figura A.21: Comparación entrenamiento EfficientNetV2 sobre CIFAR10 para $d = 32$.

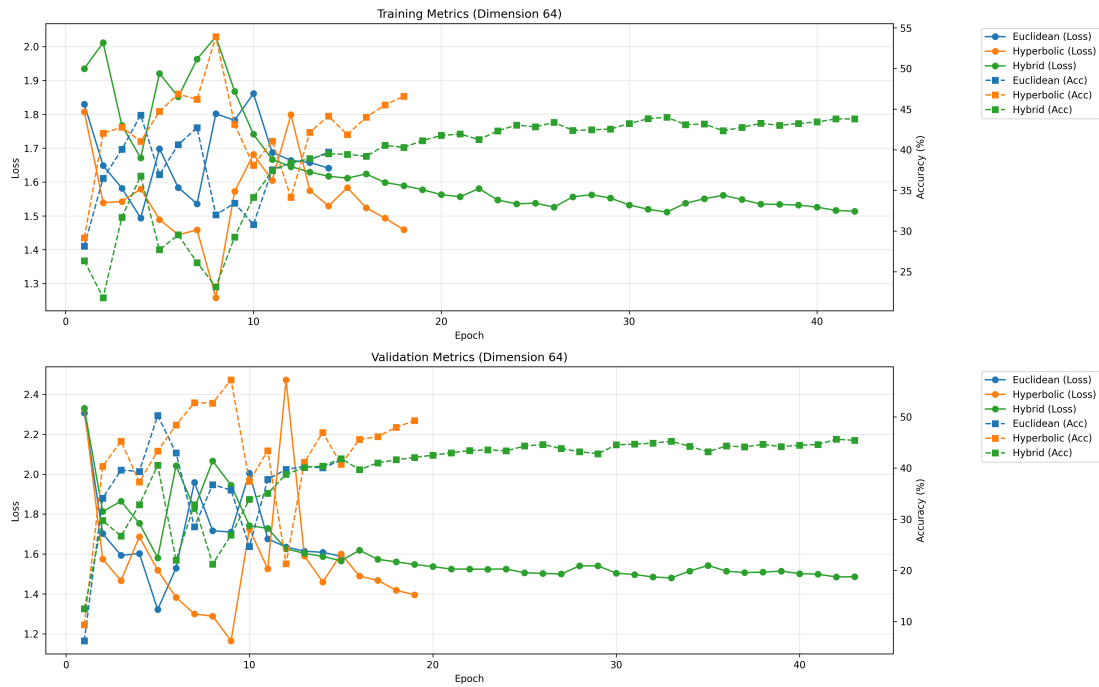


Figura A.22: Comparación entrenamiento EfficientNetV2 sobre CIFAR10 para $d = 64$.

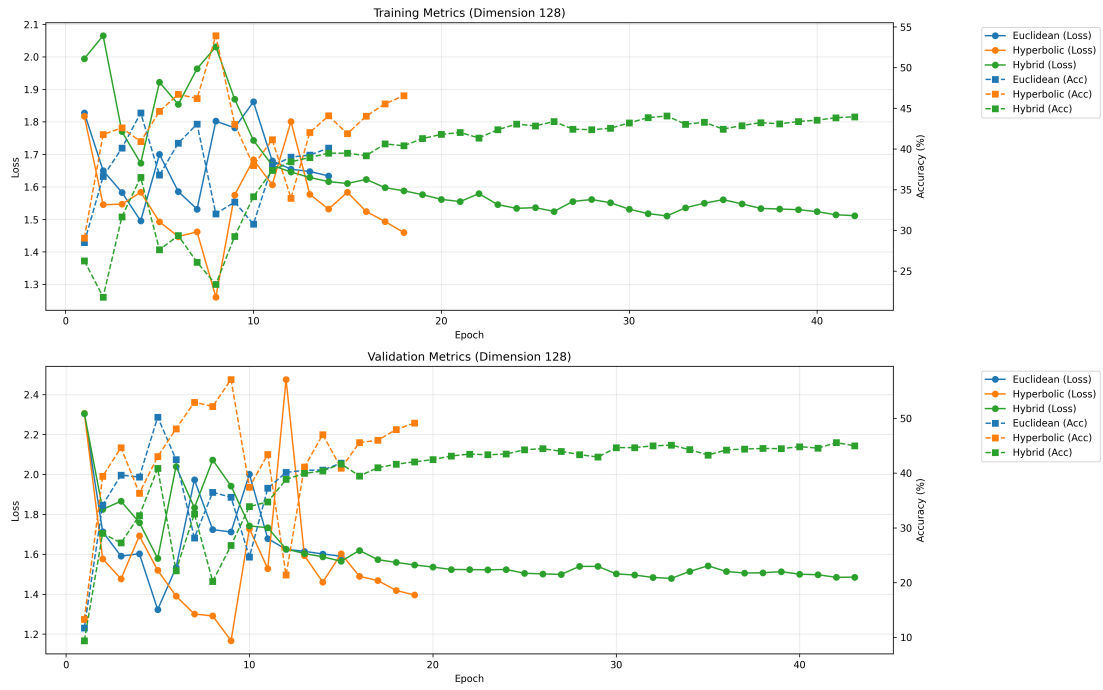


Figura A.23: Comparación entrenamiento EfficientNetV2 sobre CIFAR10 para $d = 128$.

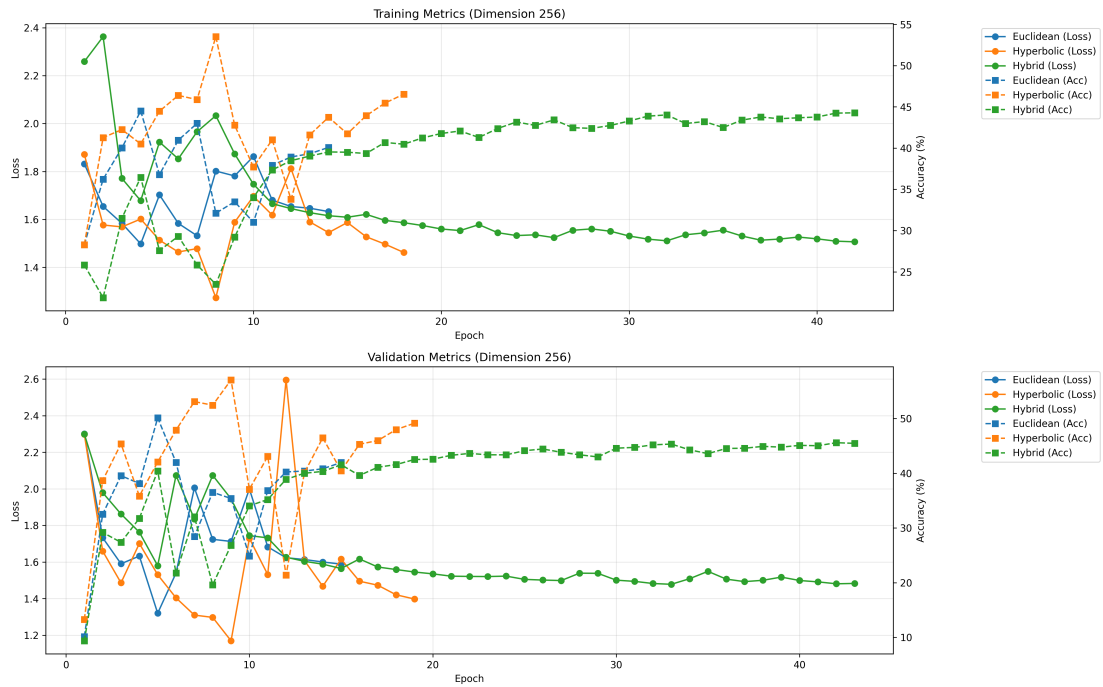


Figura A.24: Comparación entrenamiento EfficientNetV2 sobre CIFAR10 para $d = 256$.

A.1.2. CIFAR100 - ResNet50

Las figuras a continuación presentan los resultados de entrenamiento por dimensión para cada modelo.

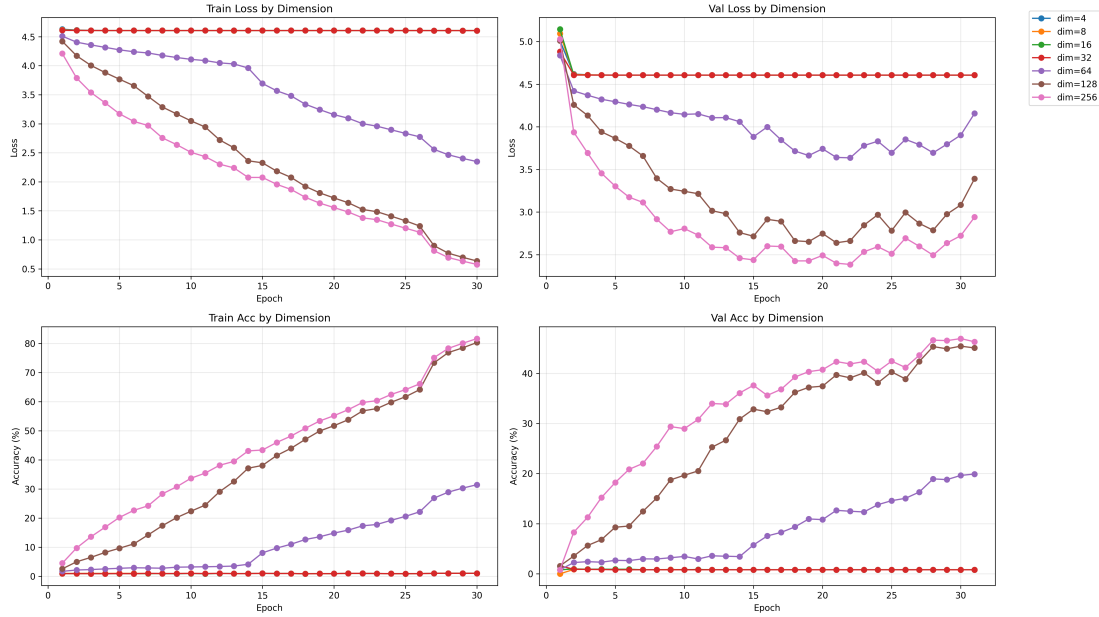


Figura A.25: Resultados entrenamiento ResNet50 euclidiano sobre CIFAR100.

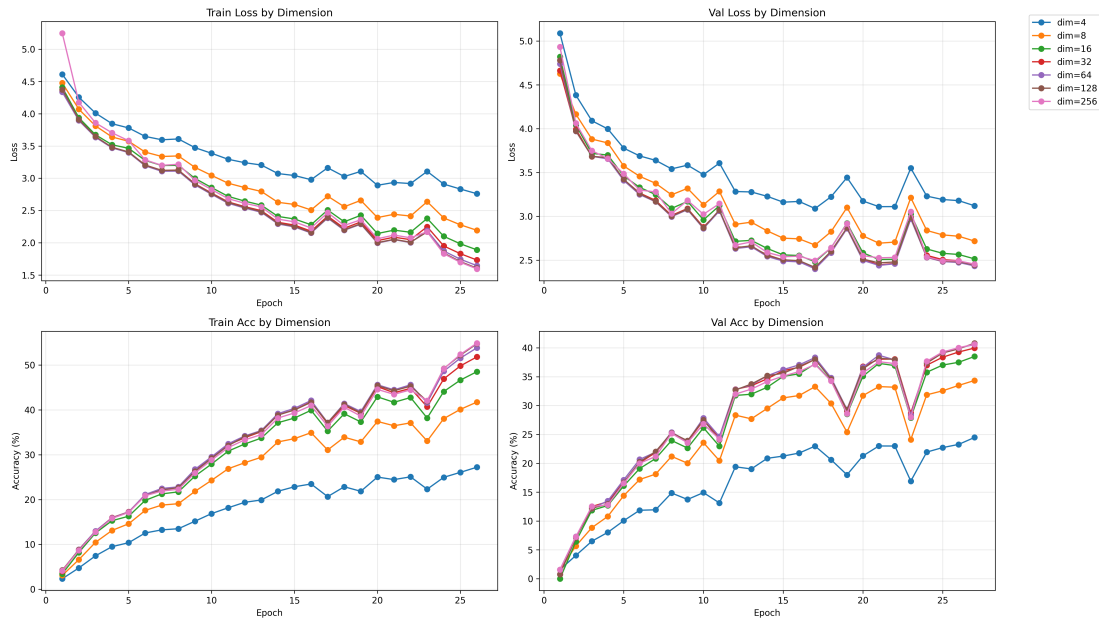


Figura A.26: Resultados entrenamiento ResNet50 hiperbólico sobre CIFAR100.

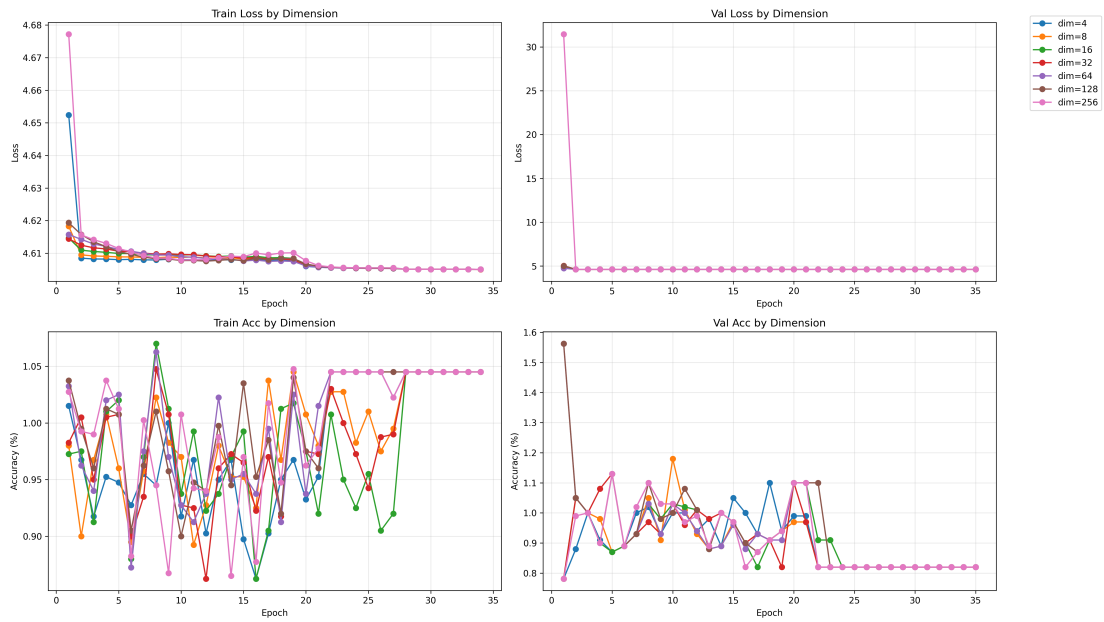


Figura A.27: Resultados entrenamiento ResNet50 híbrido sobre CIFAR100.

Las figuras a continuación comparan las pérdidas de entrenamiento y validación entre los tres modelos.

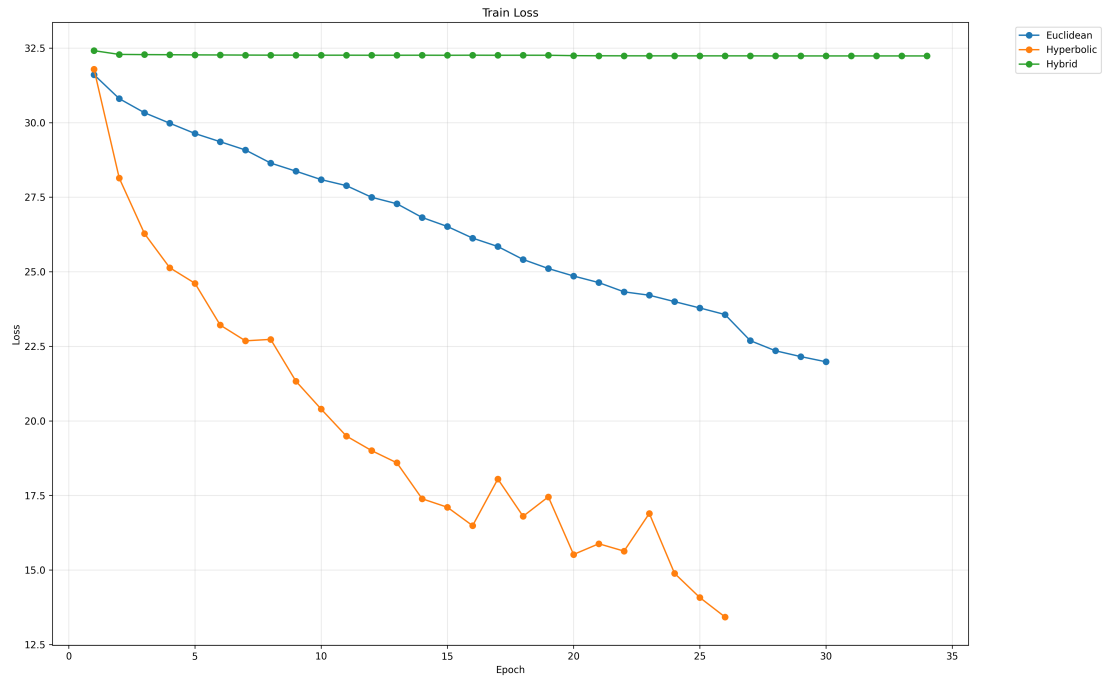


Figura A.28: Comparación entrenamiento ResNet50 sobre CIFAR100.

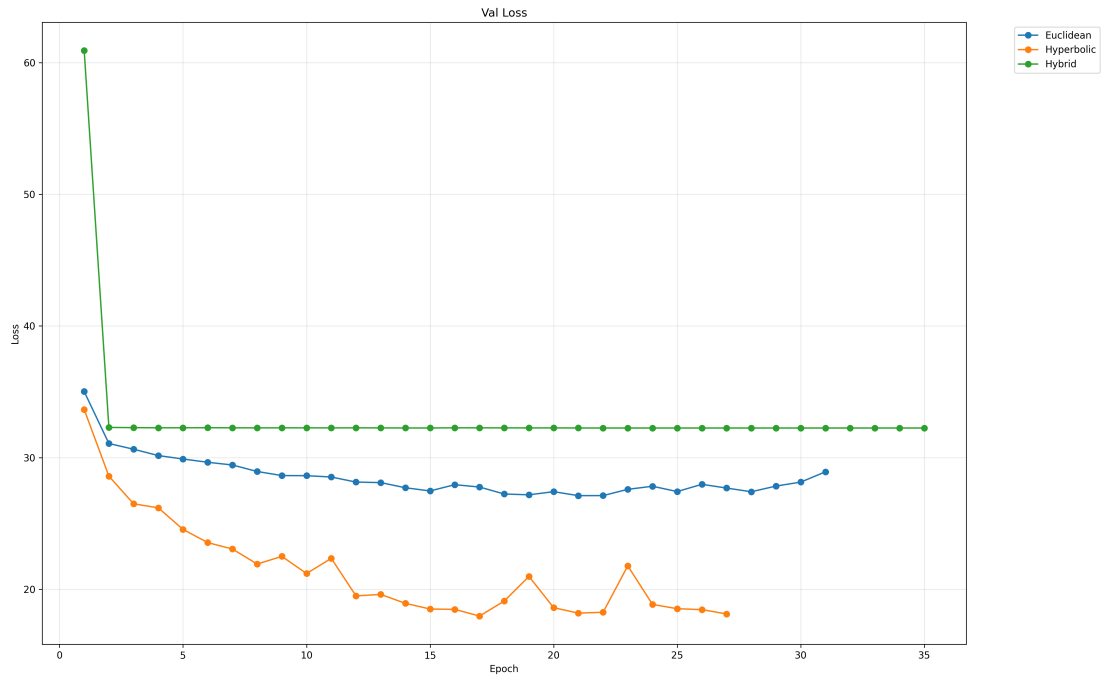
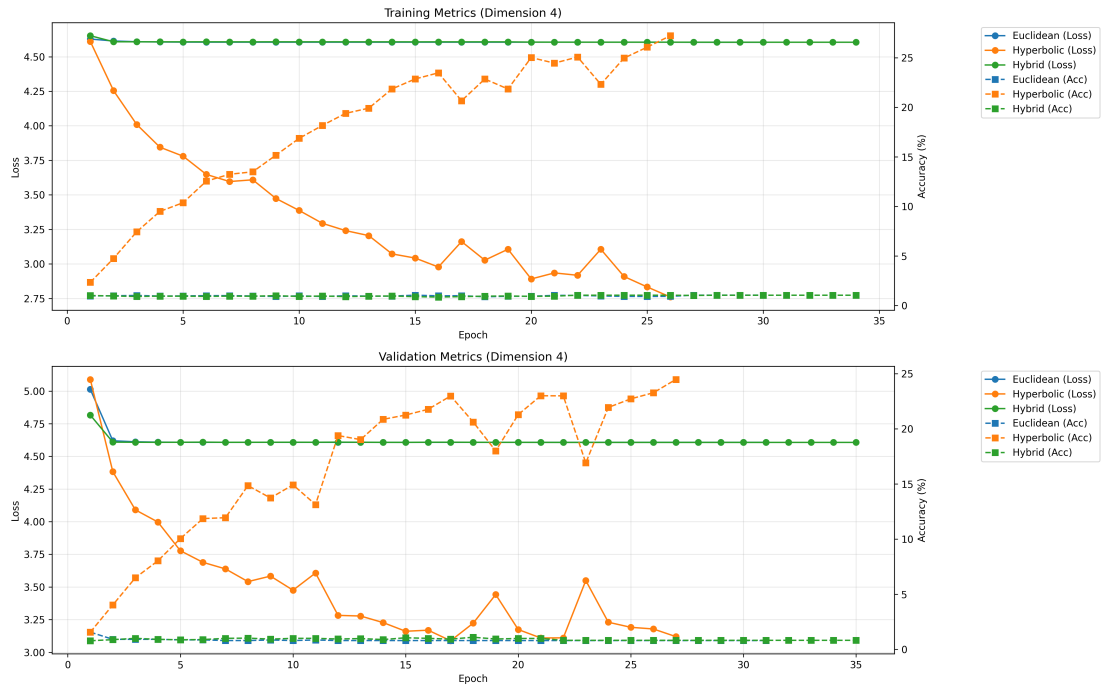
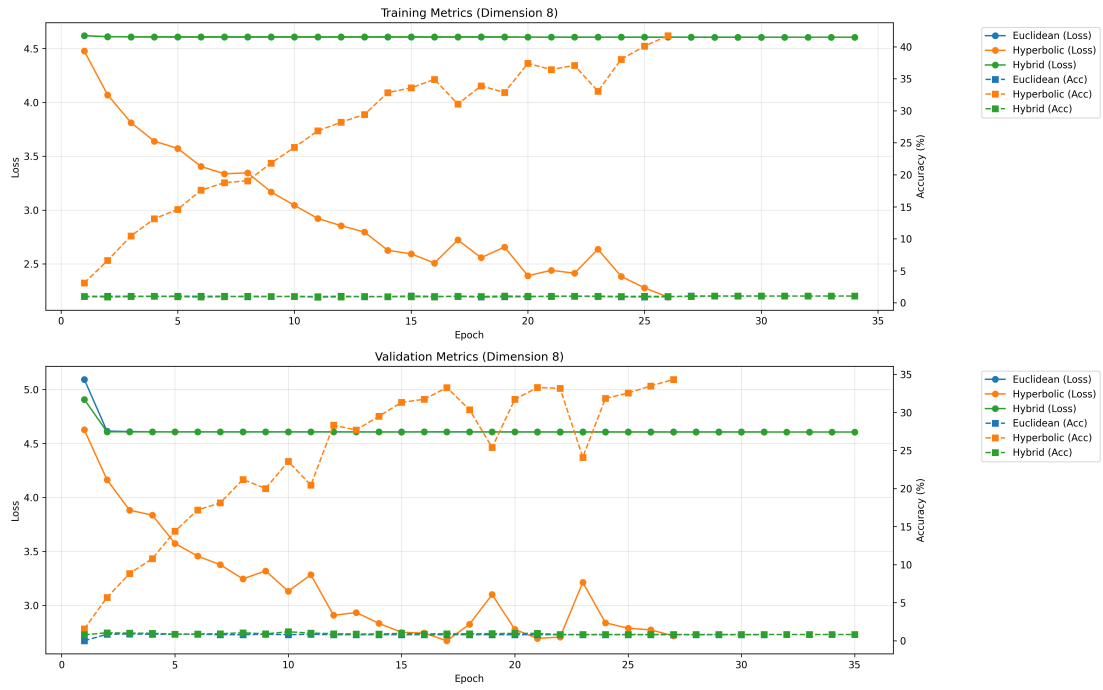
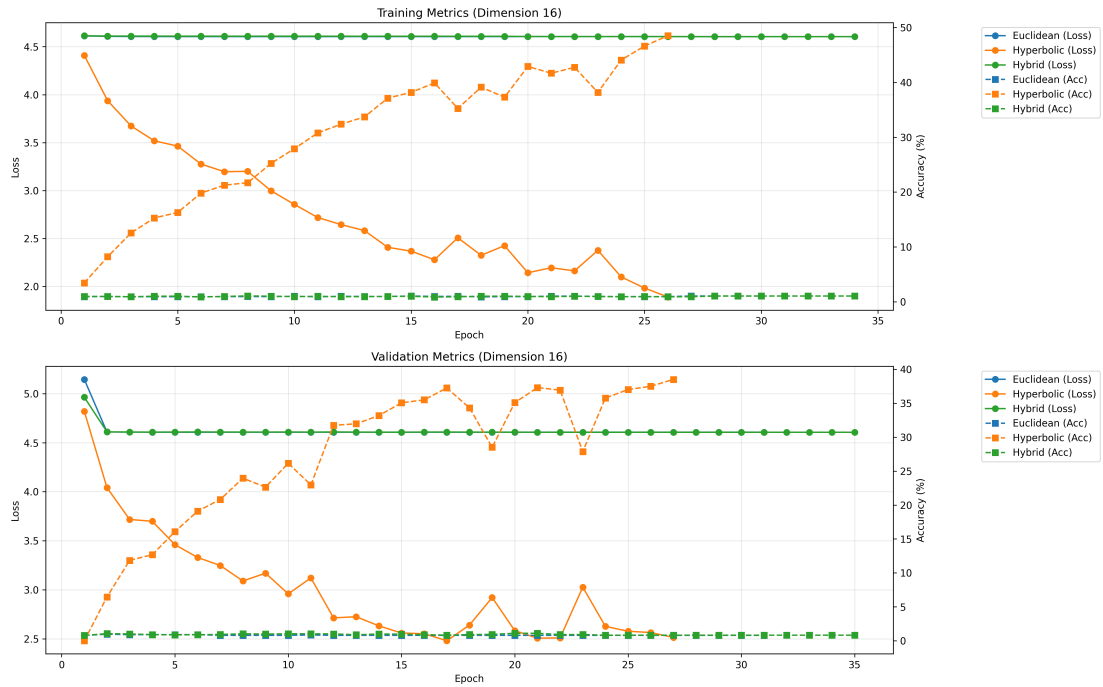
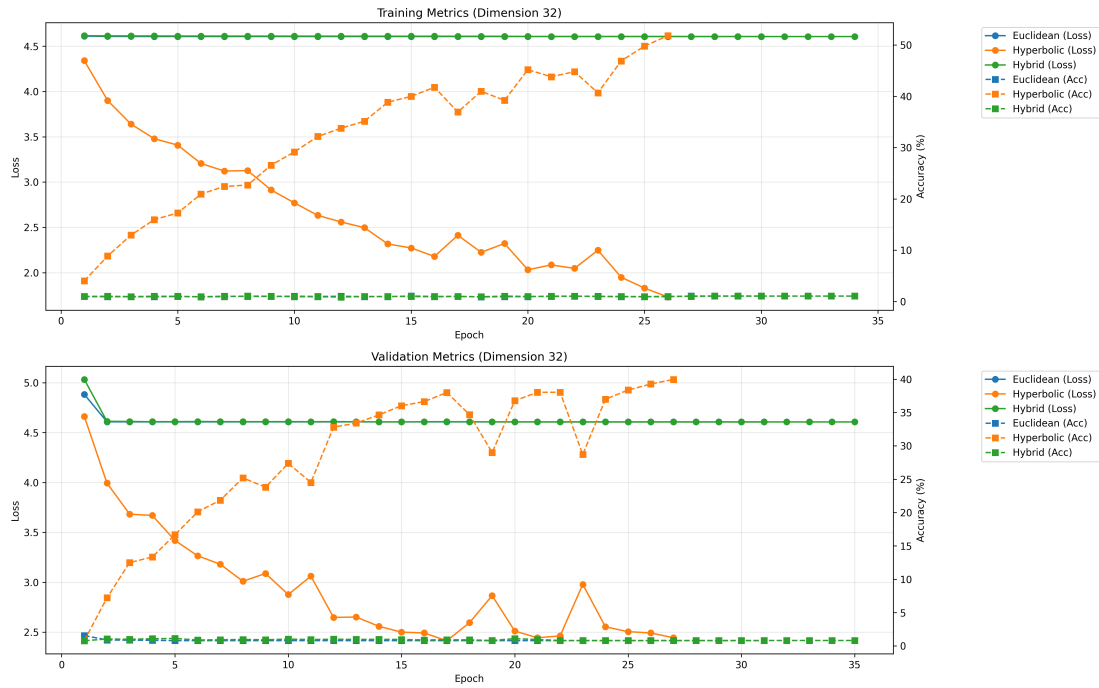
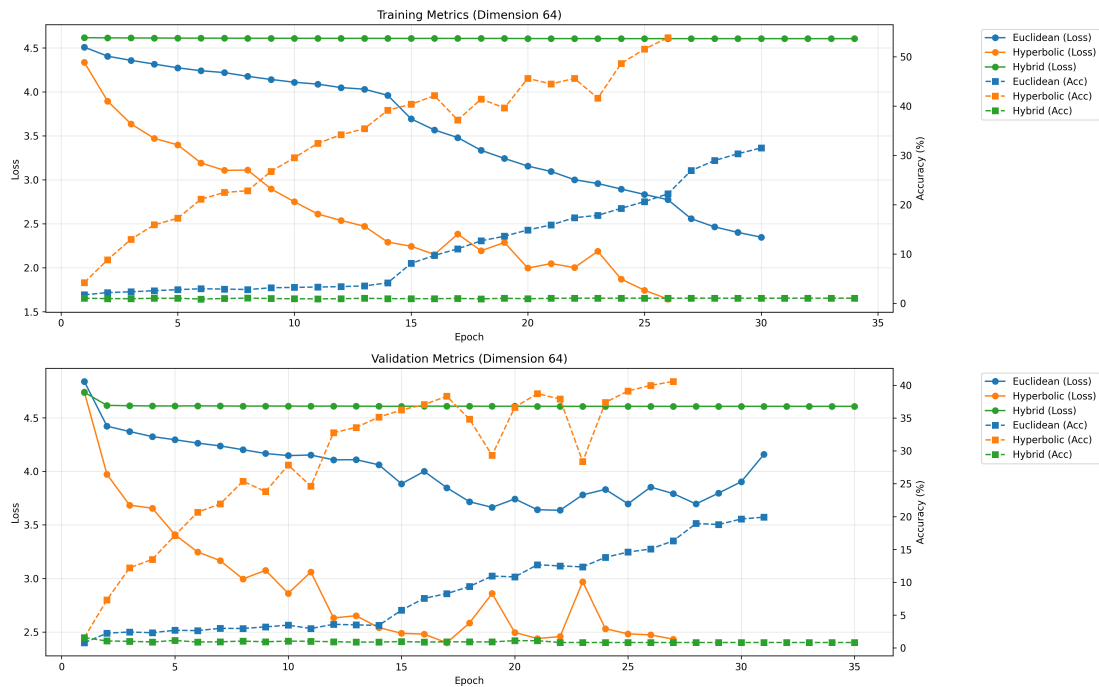


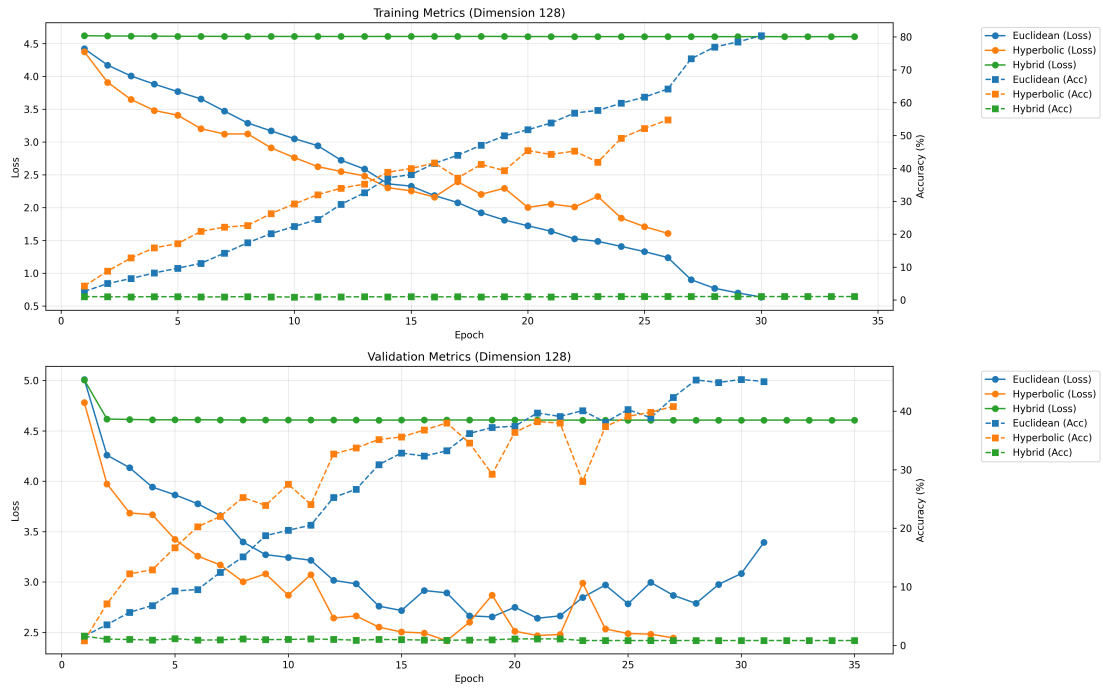
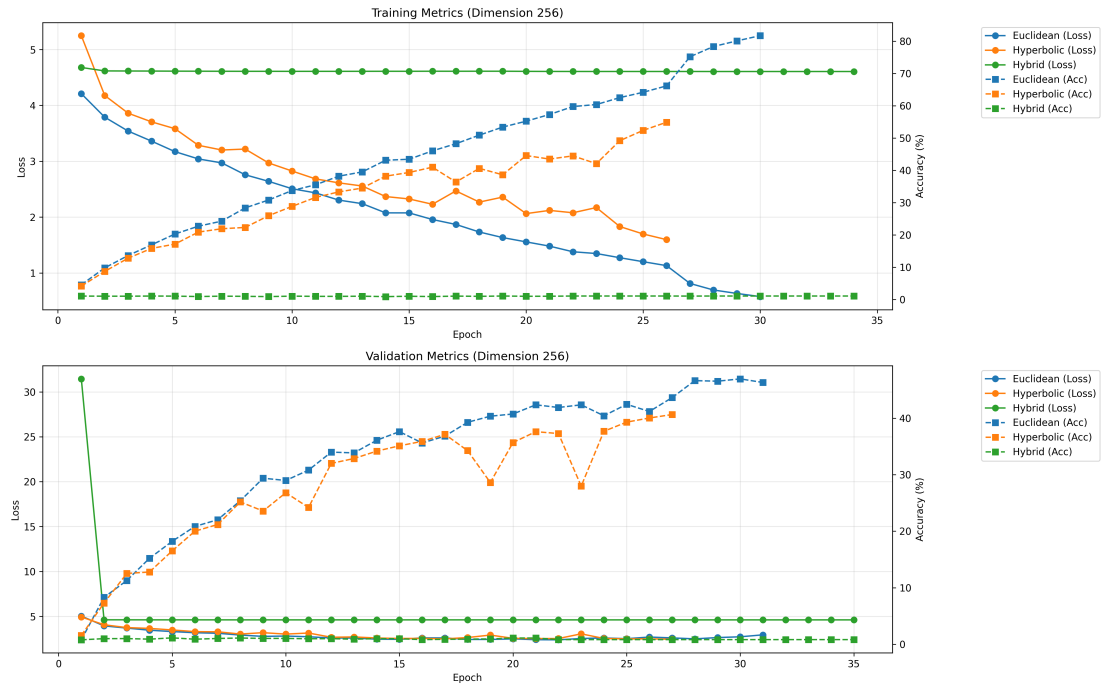
Figura A.29: Comparación validación ResNet50 sobre CIFAR100.

Las figuras a continuación presentan la comparación detallada por dimensión.

Figura A.30: Comparación entrenamiento ResNet50 sobre CIFAR100 para $d = 4$.

Figura A.31: Comparación entrenamiento ResNet50 sobre CIFAR100 para $d = 8$.Figura A.32: Comparación entrenamiento ResNet50 sobre CIFAR100 para $d = 16$.

Figura A.33: Comparación entrenamiento ResNet50 sobre CIFAR100 para $d = 32$.Figura A.34: Comparación entrenamiento ResNet50 sobre CIFAR100 para $d = 64$.

Figura A.35: Comparación entrenamiento ResNet50 sobre CIFAR100 para $d = 128$.Figura A.36: Comparación entrenamiento ResNet50 sobre CIFAR100 para $d = 256$.

A.1.3. CIFAR100 - EfficientNetV2

Las figuras a continuación presentan los resultados de entrenamiento por dimensión para cada modelo.

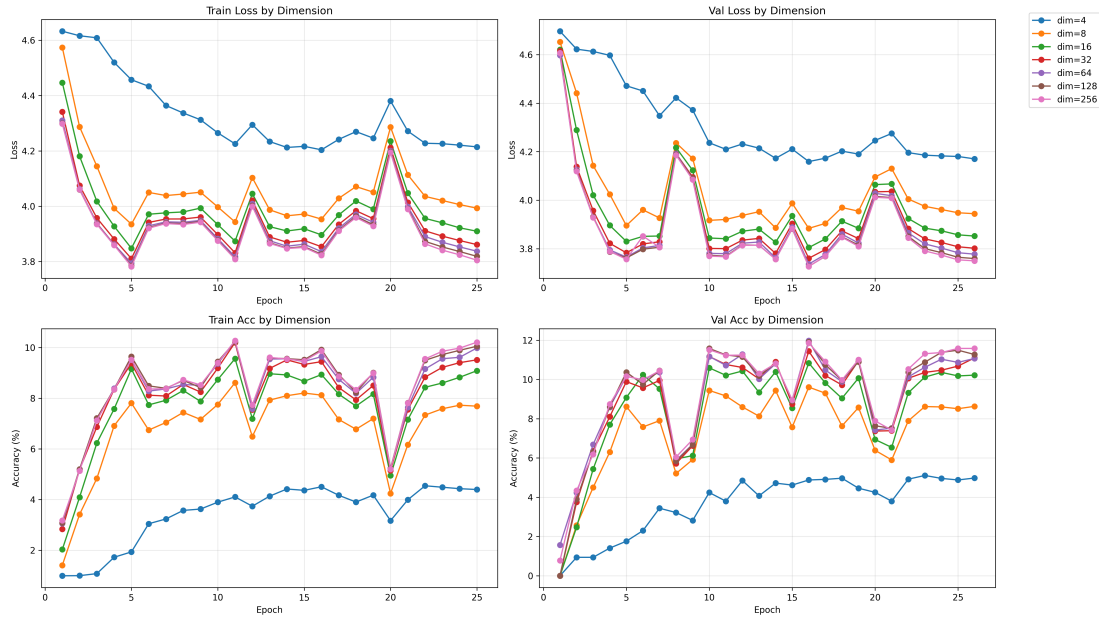


Figura A.37: Resultados entrenamiento EfficientNetV2 euclidiano sobre CIFAR100.

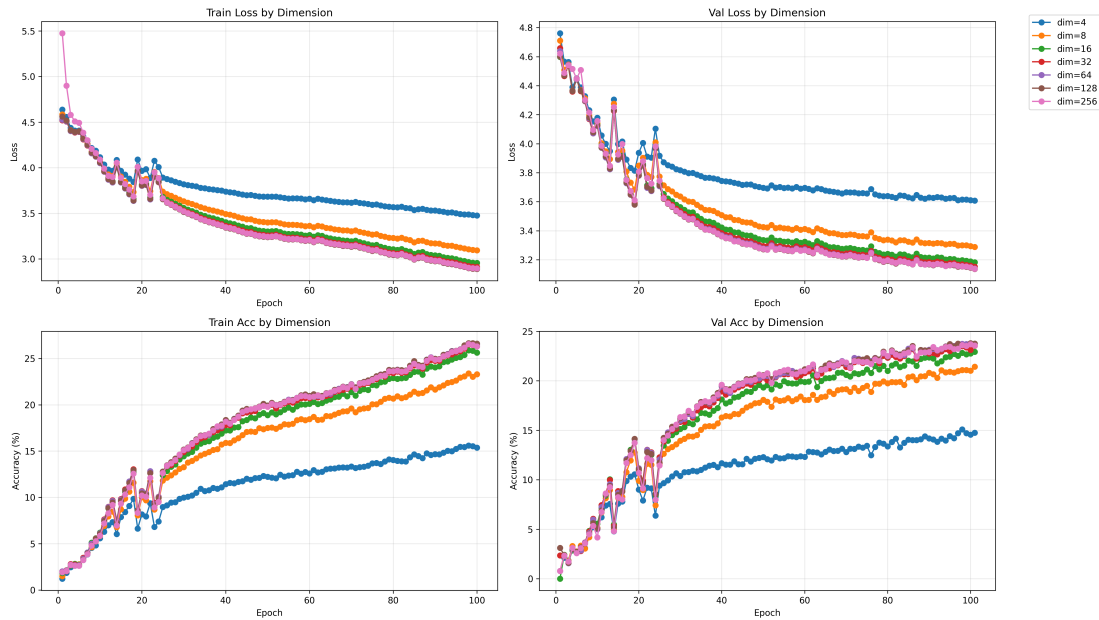


Figura A.38: Resultados entrenamiento EfficientNetV2 hiperbólico sobre CIFAR100.

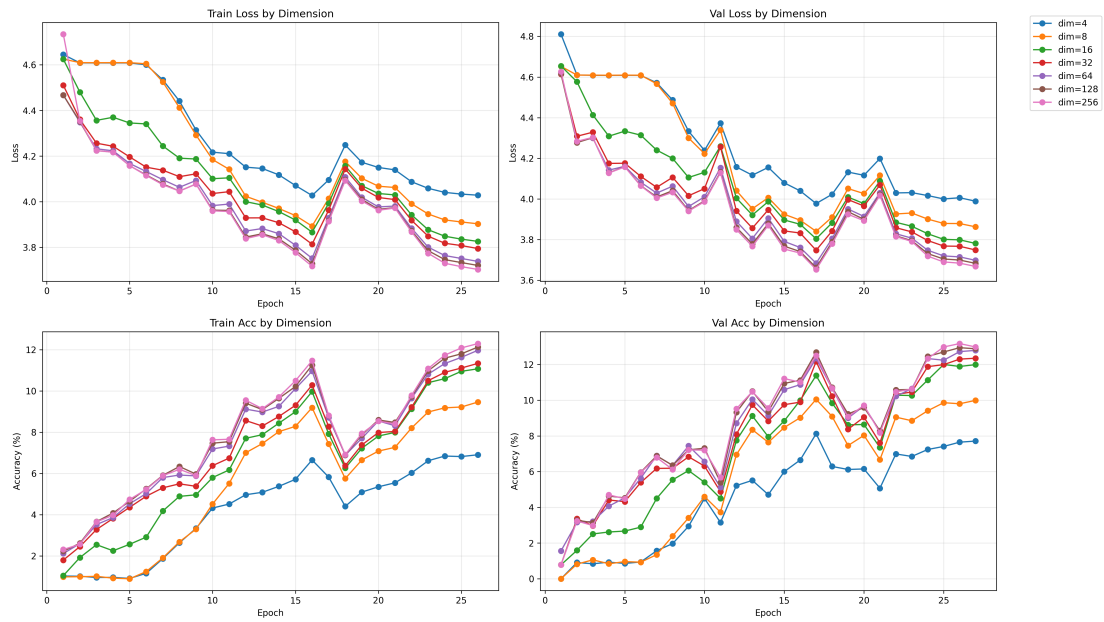


Figura A.39: Resultados entrenamiento EfficientNetV2 híbrido sobre CIFAR100.

Las figuras a continuación comparan las pérdidas de entrenamiento y validación entre los tres modelos.

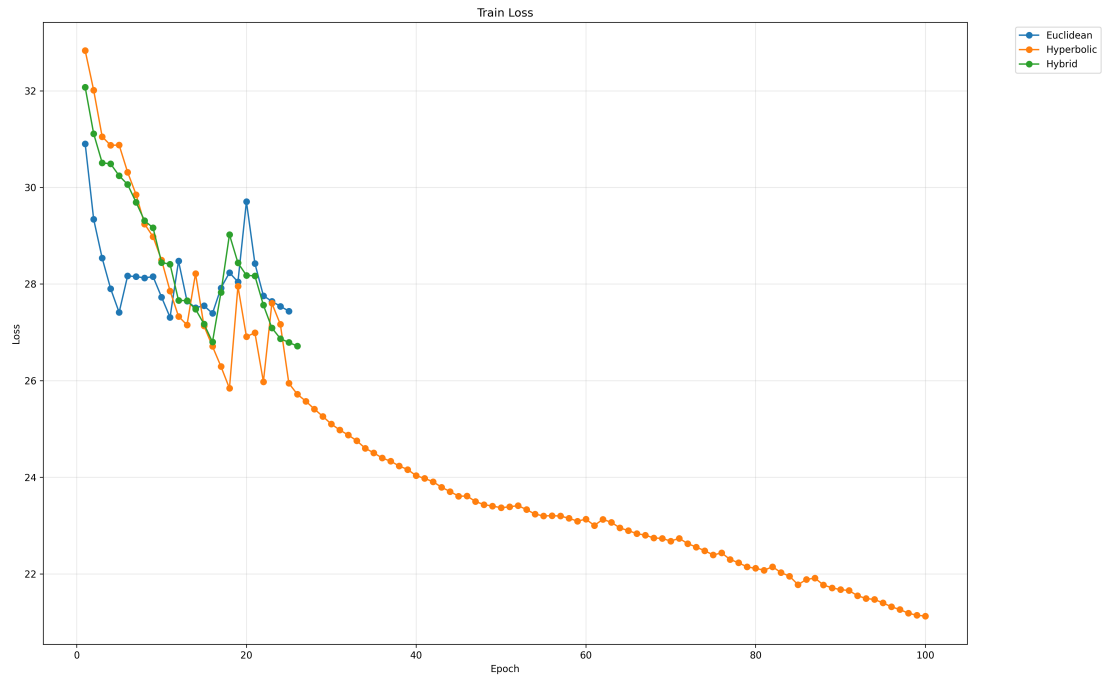


Figura A.40: Comparación entrenamiento EfficientNetV2 sobre CIFAR100.



Figura A.41: Comparación entrenamiento EfficientNetV2 sobre CIFAR100.

Las figuras a continuación presentan la comparación detallada por dimensión.

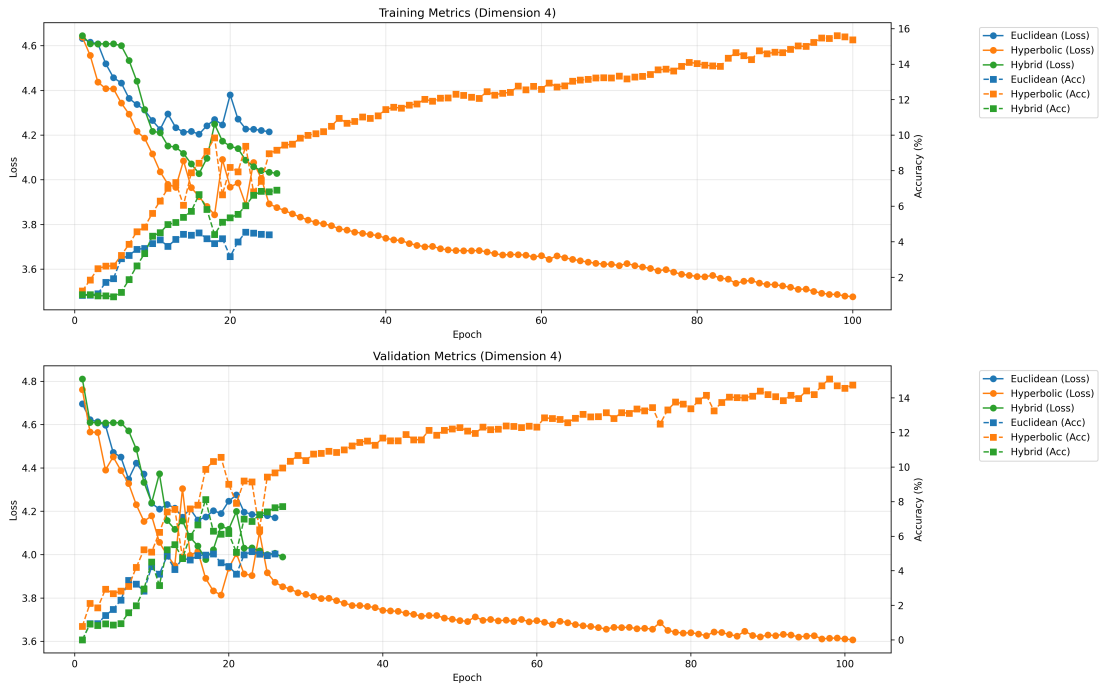
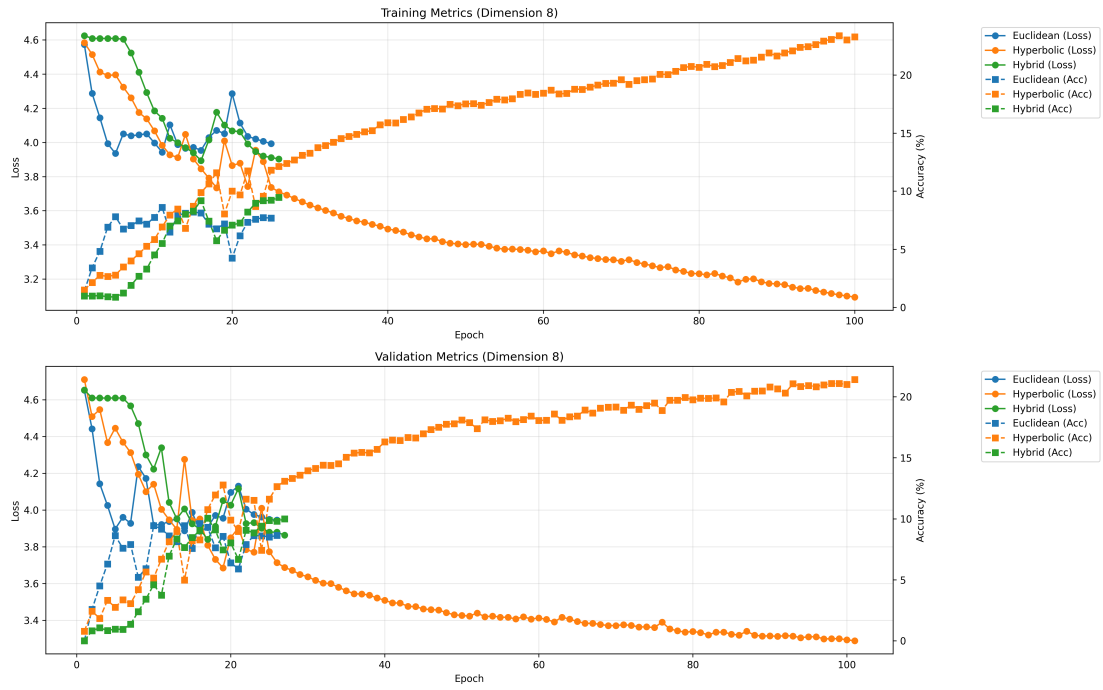
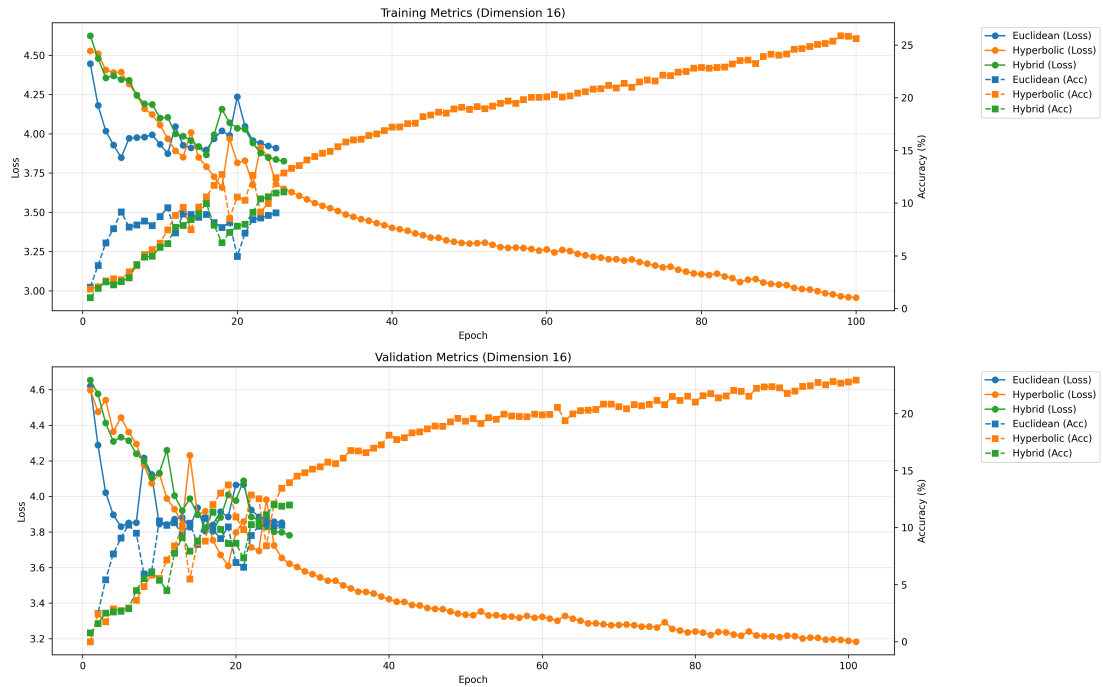


Figura A.42: Comparación entrenamiento EfficientNetV2 sobre CIFAR100 para $d = 4$.

Figura A.43: Comparación entrenamiento EfficientNetV2 sobre CIFAR100 para $d = 8$.Figura A.44: Comparación entrenamiento EfficientNetV2 sobre CIFAR100 para $d = 16$.

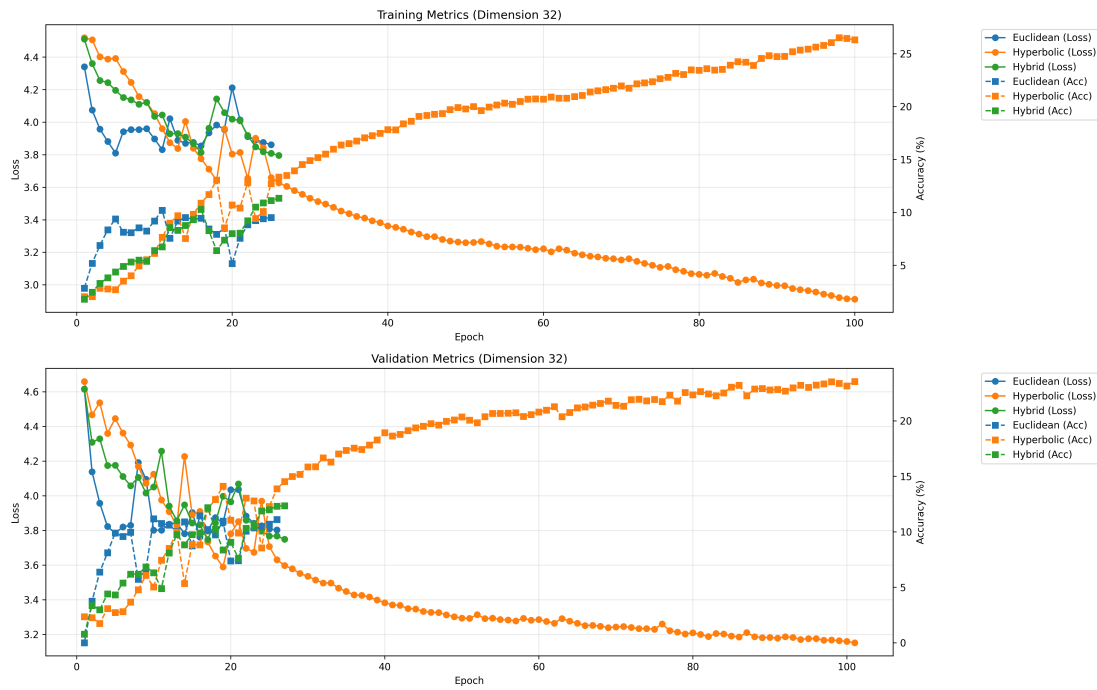


Figura A.45: Comparación entrenamiento EfficientNetV2 sobre CIFAR100 para $d = 32$.

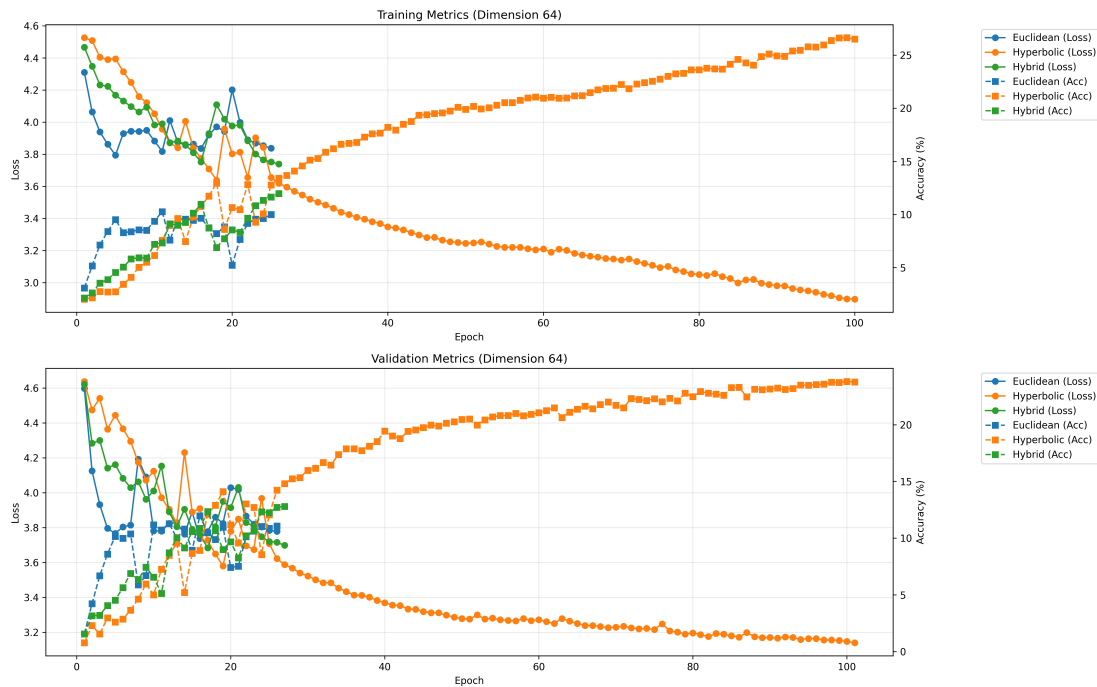
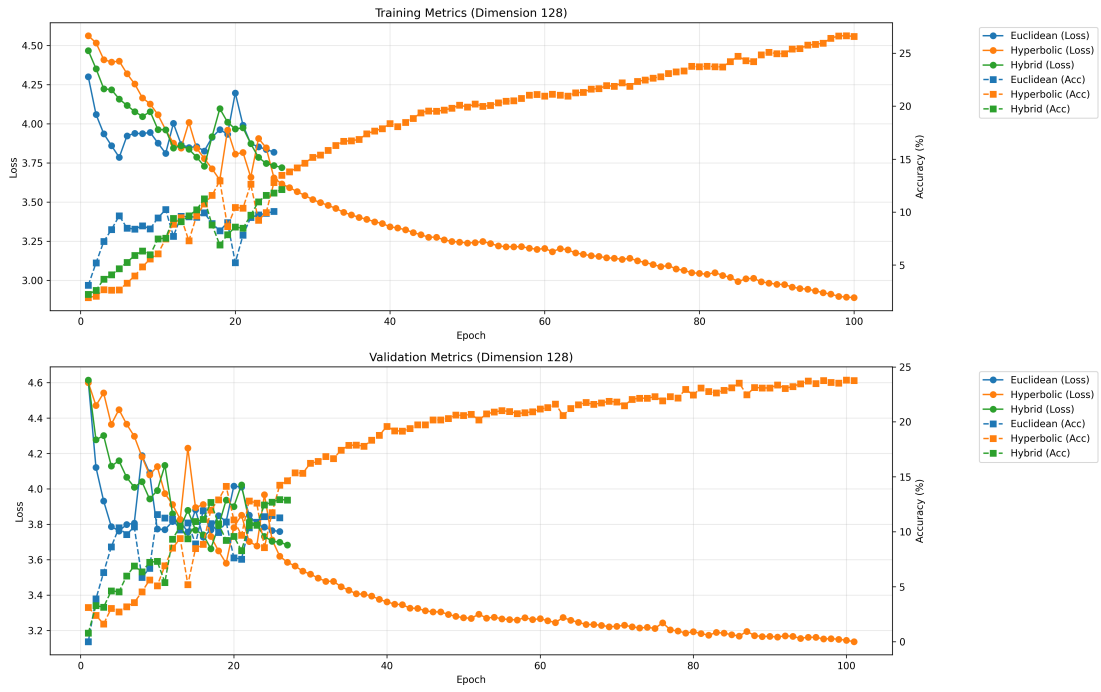
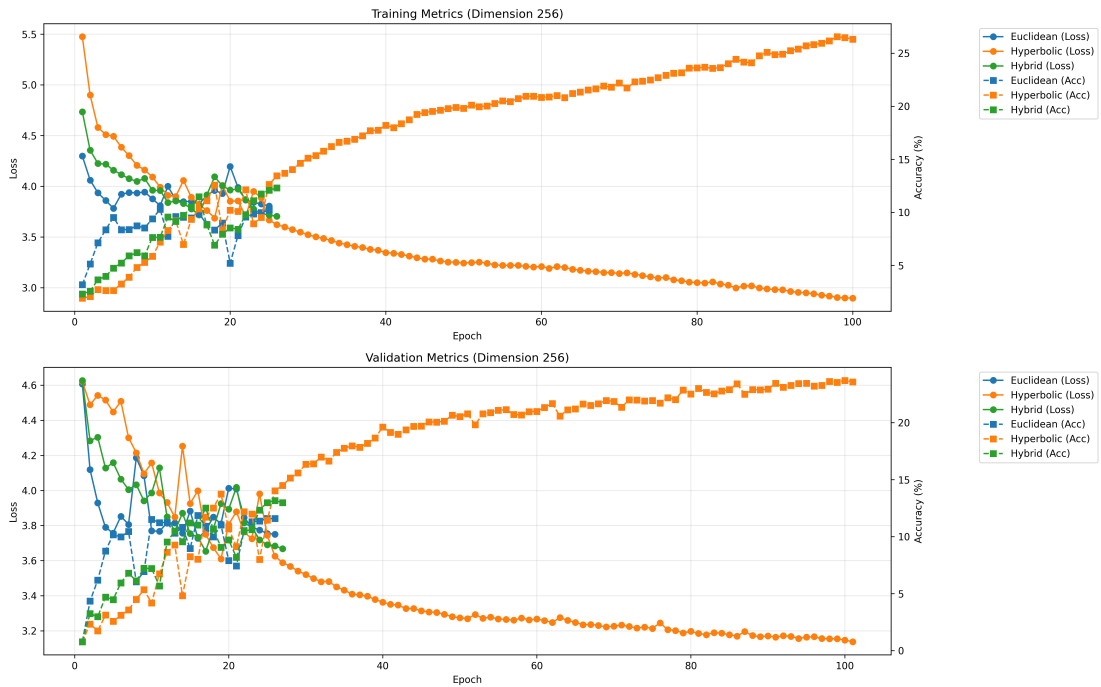


Figura A.46: Comparación entrenamiento EfficientNetV2 sobre CIFAR100 para $d = 64$.

Figura A.47: Comparación entrenamiento EfficientNetV2 sobre CIFAR100 para $d = 128$.Figura A.48: Comparación entrenamiento EfficientNetV2 sobre CIFAR100 para $d = 256$.

A.1.4. Imagenette - ResNet50

Las figuras a continuación presentan los resultados de entrenamiento por dimensión para cada modelo.

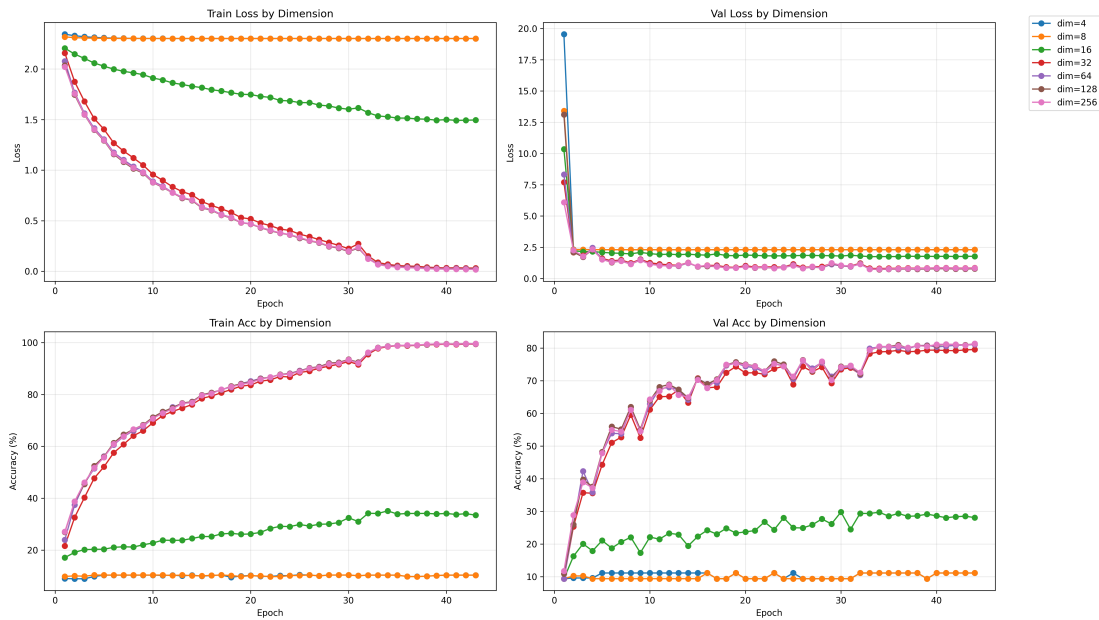


Figura A.49: Resultados entrenamiento ResNet50 euclidiano sobre Imagenette.

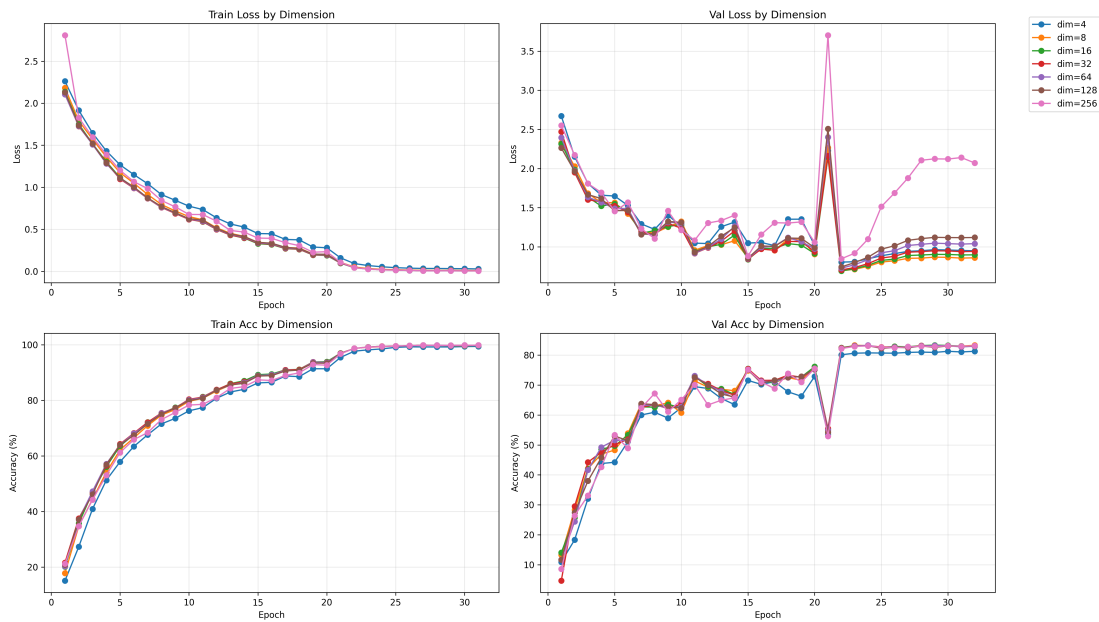


Figura A.50: Resultados entrenamiento ResNet50 hiperbólico sobre Imagenette.

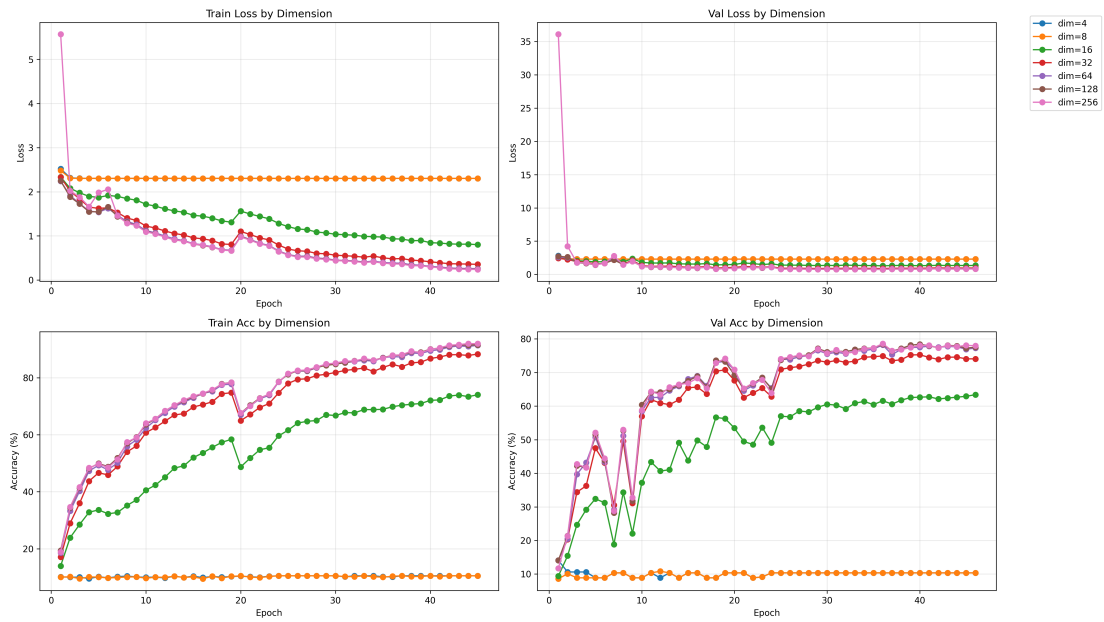


Figura A.51: Resultados entrenamiento ResNet50 híbrido sobre Imagenette.

Las figuras a continuación comparan las pérdidas de entrenamiento y validación entre los tres modelos.

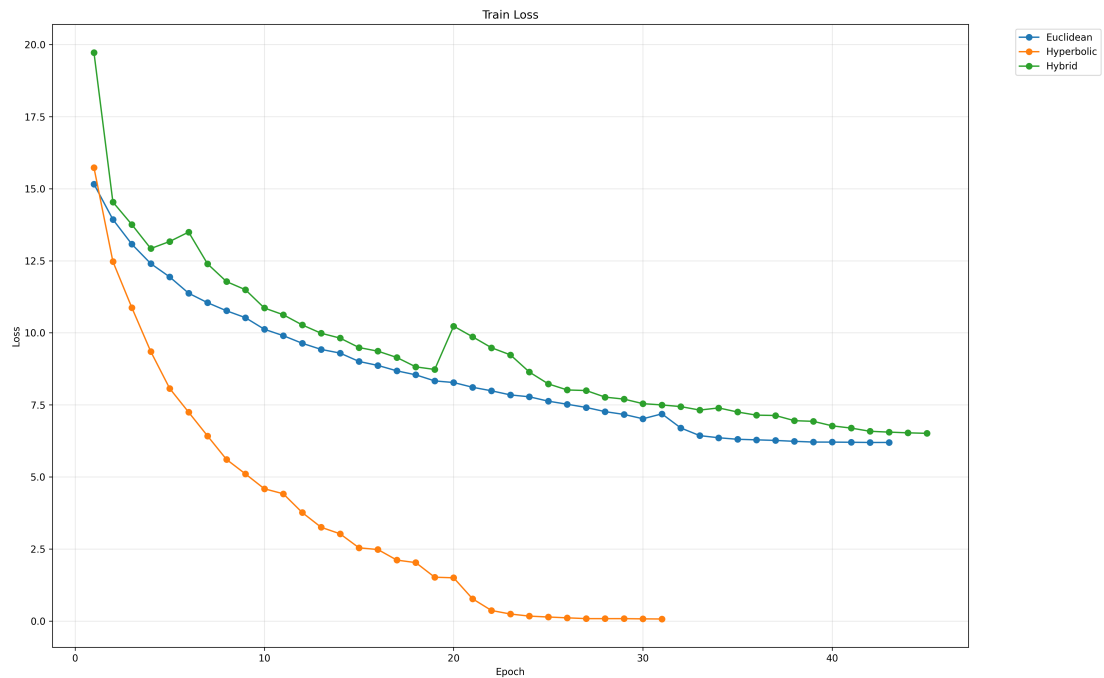


Figura A.52: Comparación entrenamiento ResNet50 sobre Imagenette.

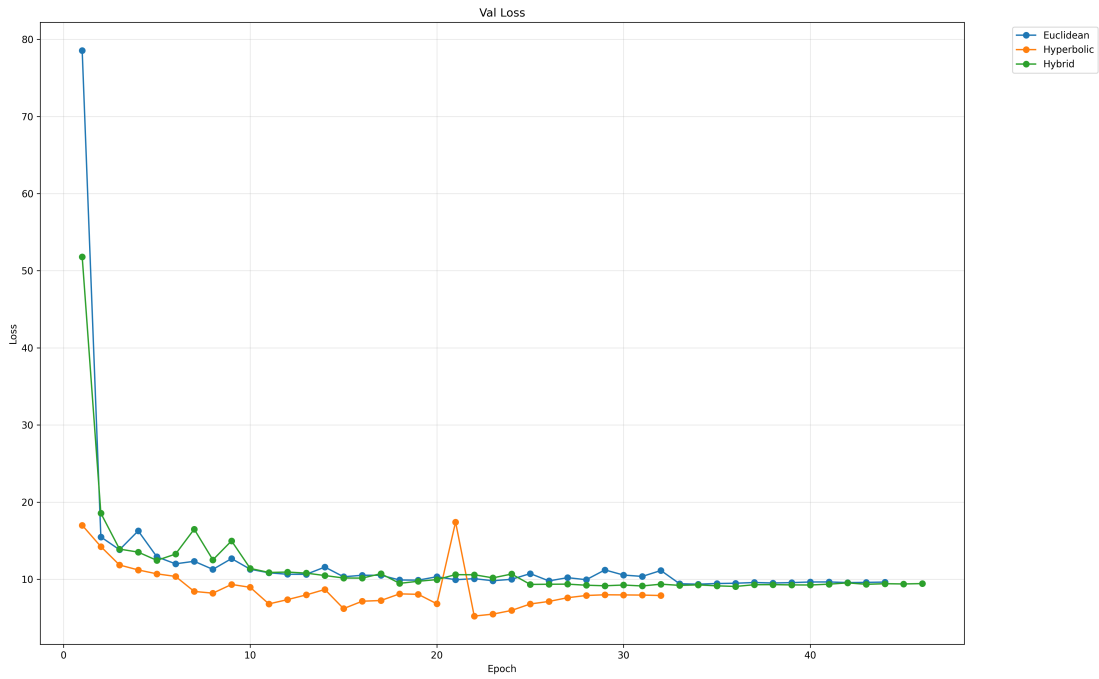
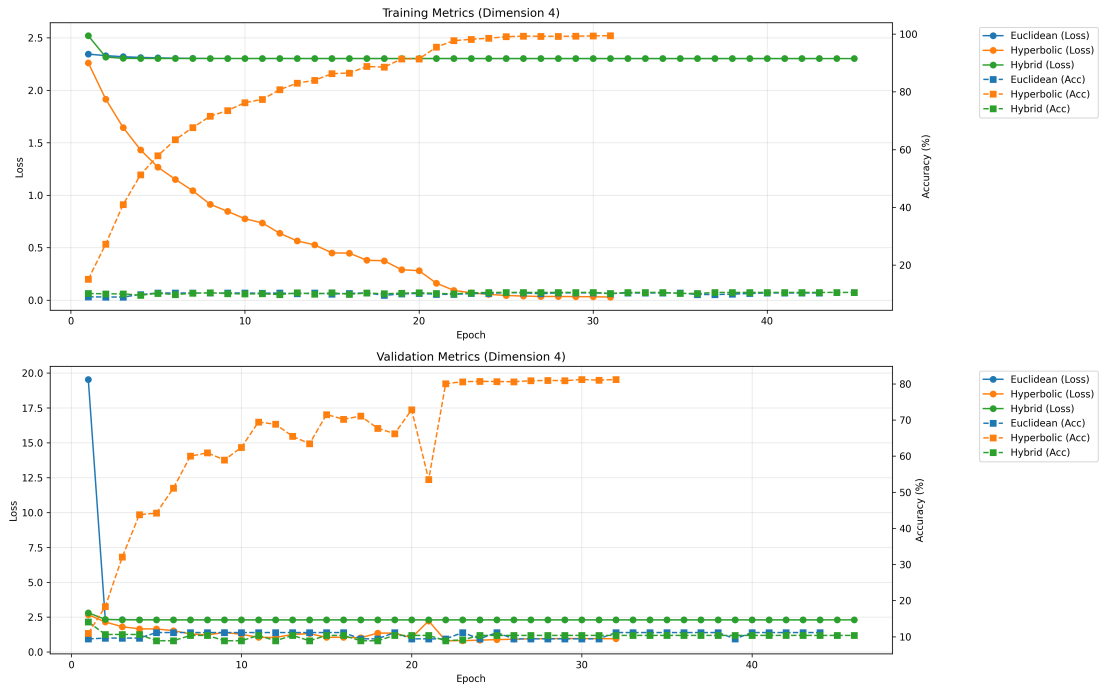
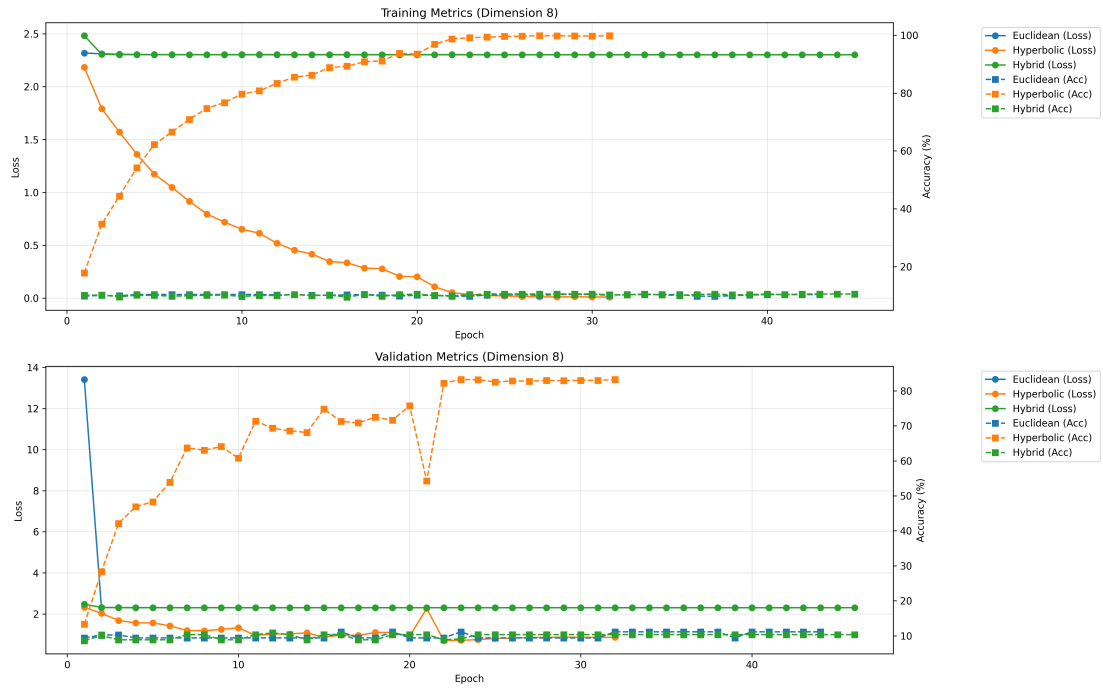
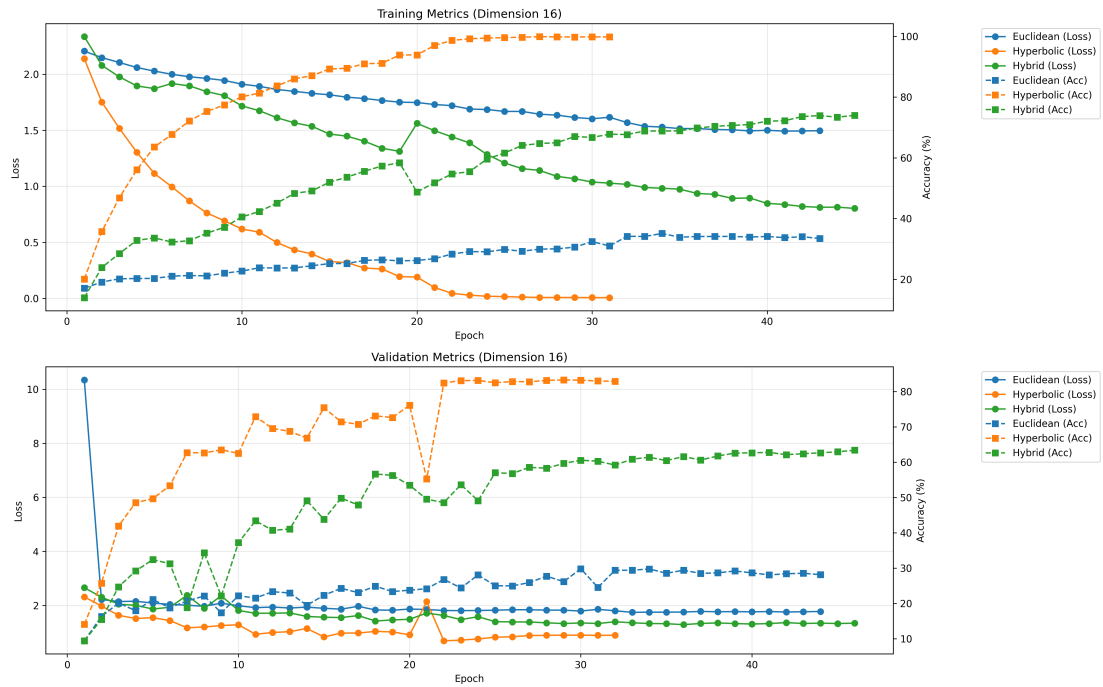


Figura A.53: Comparación validación ResNet50 sobre Imagenette.

Las figuras a continuación presentan la comparación detallada por dimensión.

Figura A.54: Comparación entrenamiento ResNet50 sobre Imagenette para $d = 4$.

Figura A.55: Comparación entrenamiento ResNet50 sobre Imagenette para $d = 8$.Figura A.56: Comparación entrenamiento ResNet50 sobre Imagenette para $d = 16$.

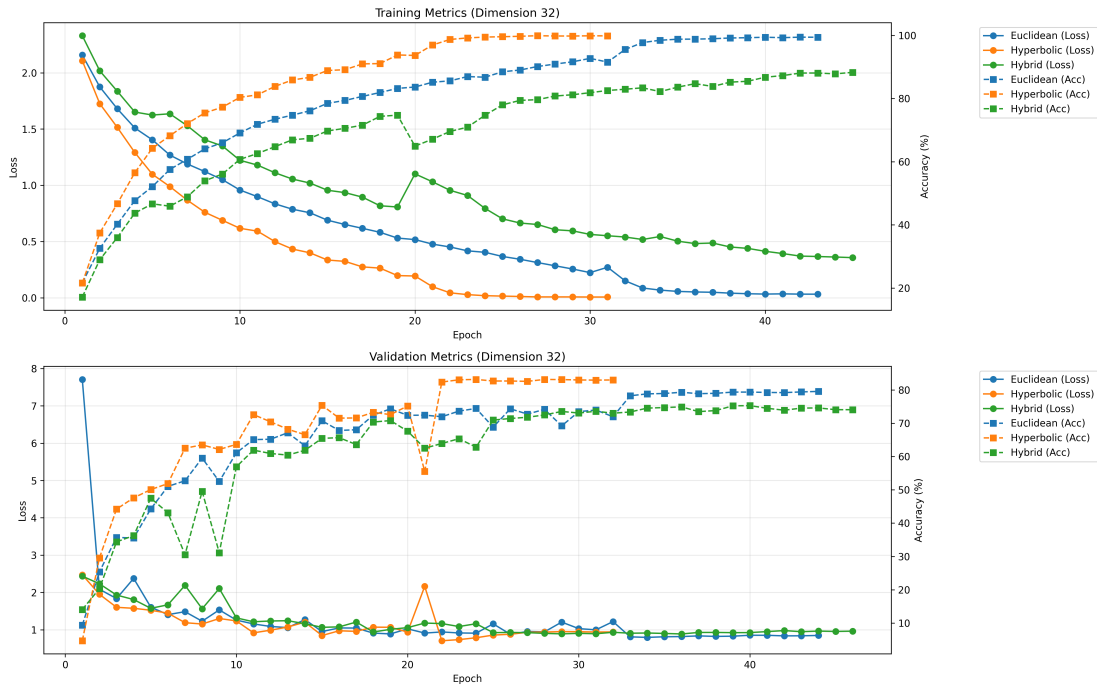


Figura A.57: Comparación entrenamiento ResNet50 sobre Imagenette para $d = 32$.

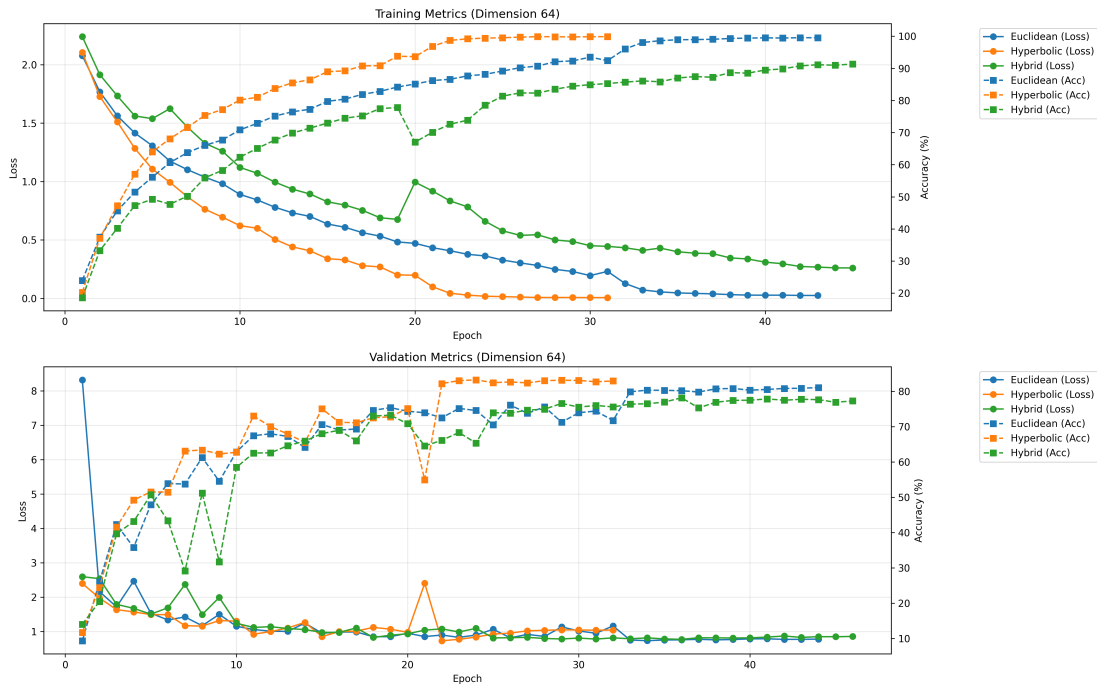
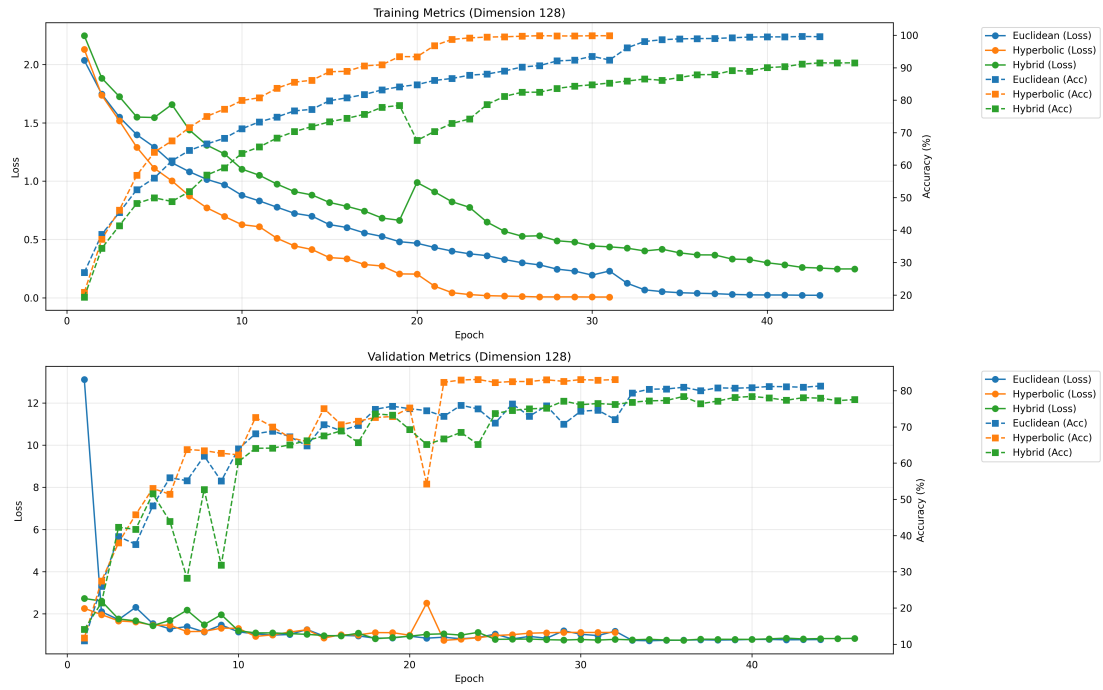
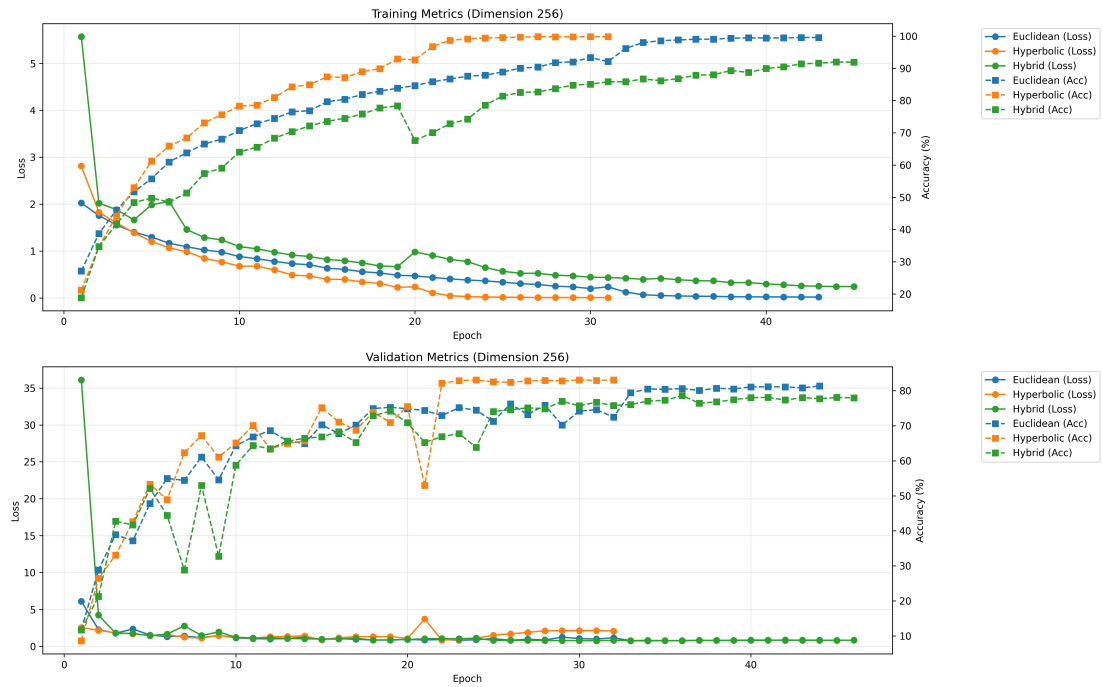


Figura A.58: Comparación entrenamiento ResNet50 sobre Imagenette para $d = 64$.

Figura A.59: Comparación entrenamiento ResNet50 sobre Imagenette para $d = 128$.Figura A.60: Comparación entrenamiento ResNet50 sobre Imagenette para $d = 256$.

A.1.5. Imagenette - EfficientNetV2

Las figuras a continuación presentan los resultados de entrenamiento por dimensión para cada modelo.

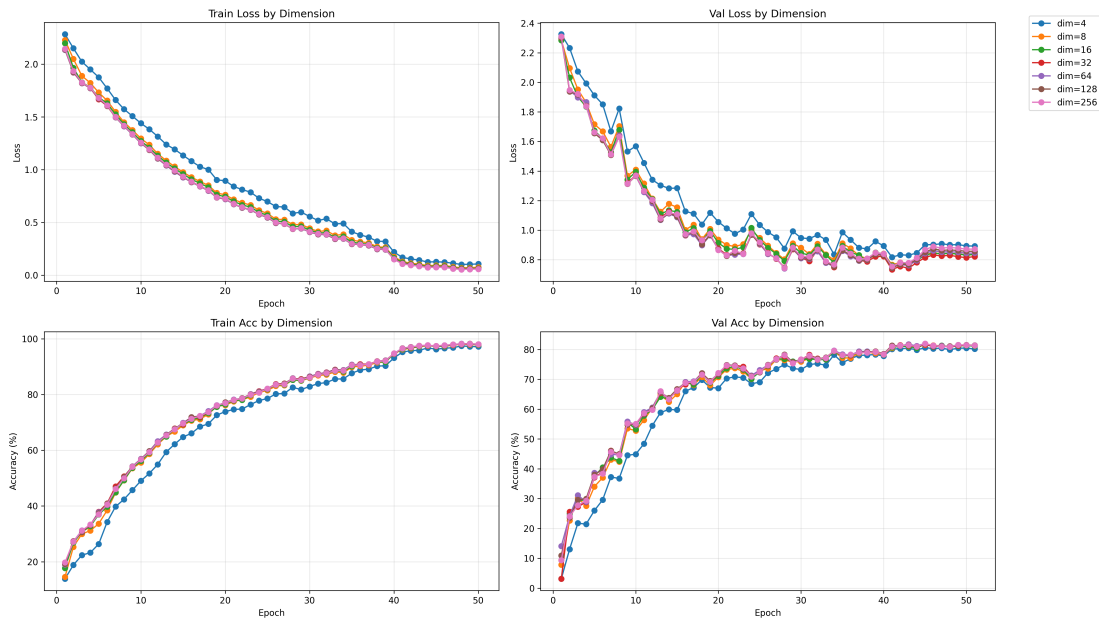


Figura A.61: Resultados entrenamiento EfficientNetV2 euclidiano sobre Imagenette.

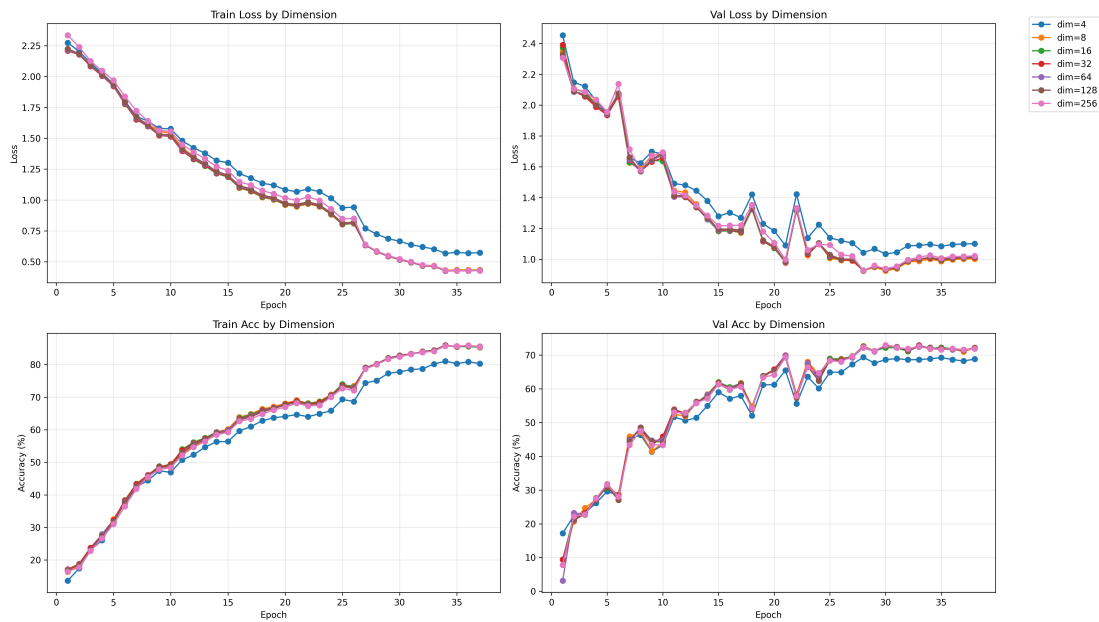


Figura A.62: Resultados entrenamiento EfficientNetV2 hiperbólico sobre Imagenette.

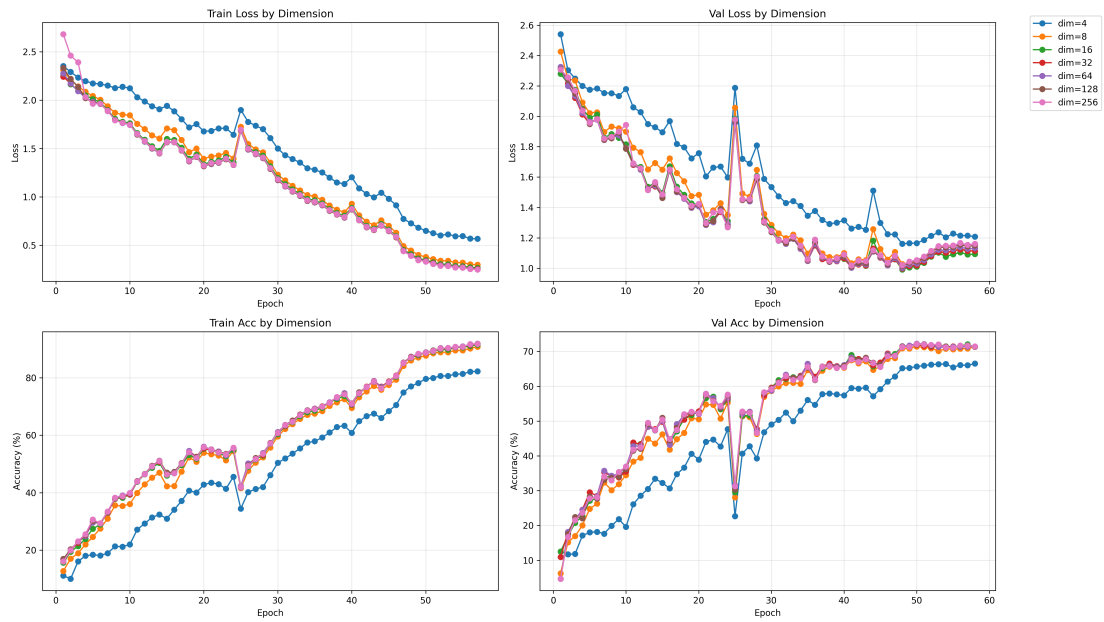


Figura A.63: Resultados entrenamiento EfficientNetV2 híbrido sobre Imagenette.

Las figuras a continuación comparan las pérdidas de entrenamiento y validación entre los tres modelos.

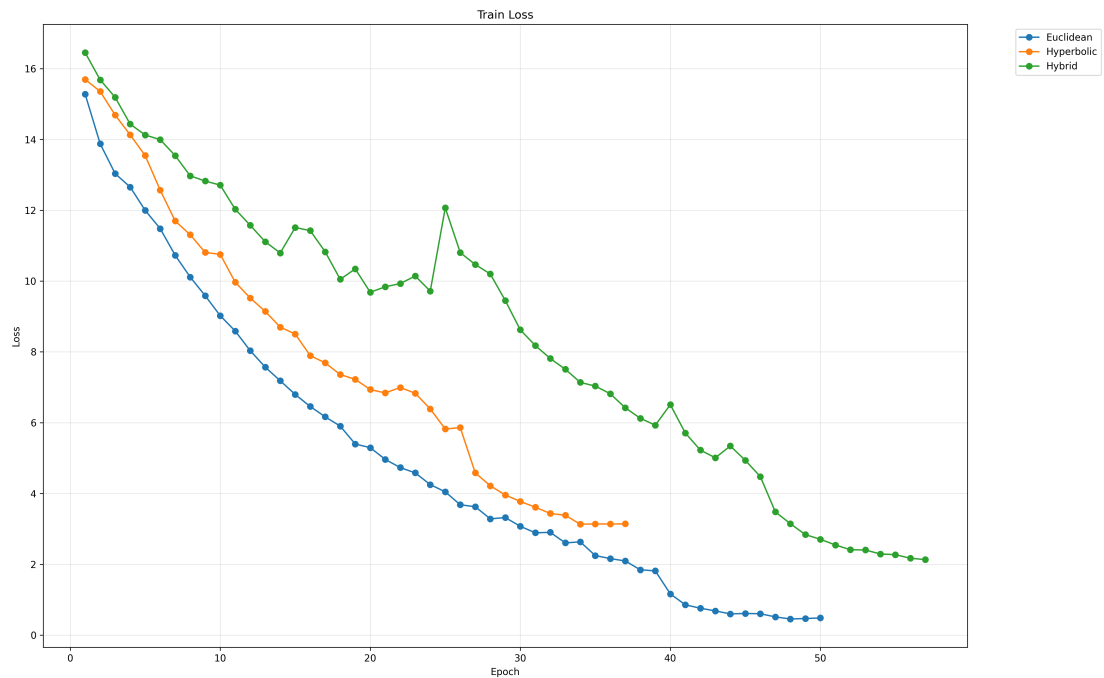


Figura A.64: Comparación entrenamiento EfficientNetV2 sobre Imagenette.

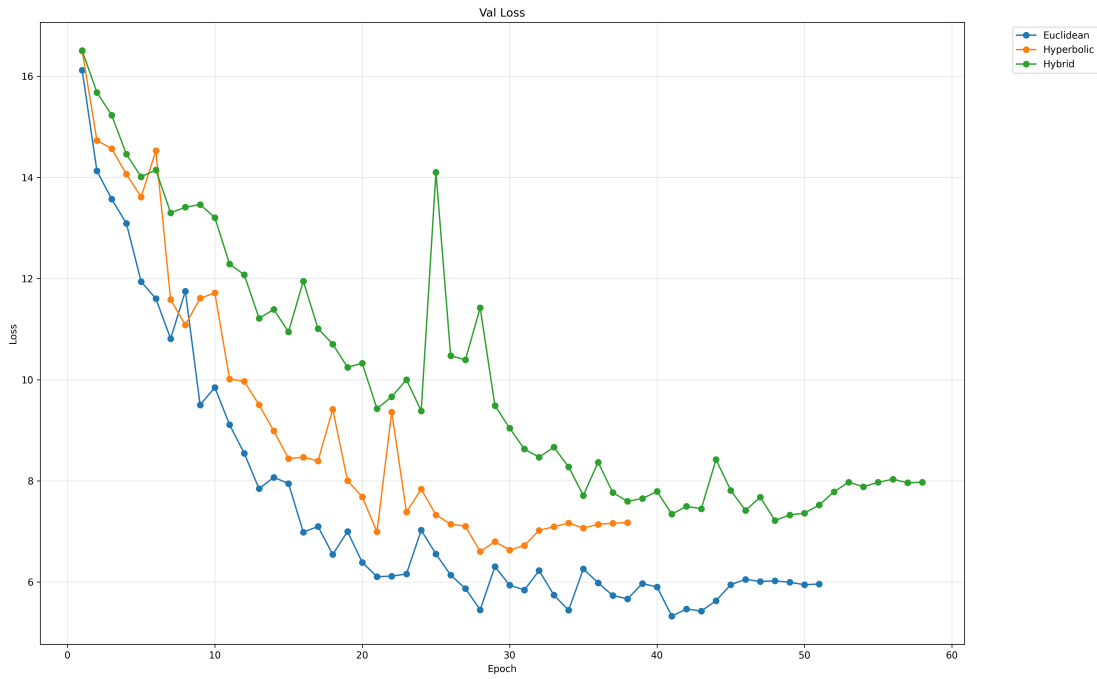
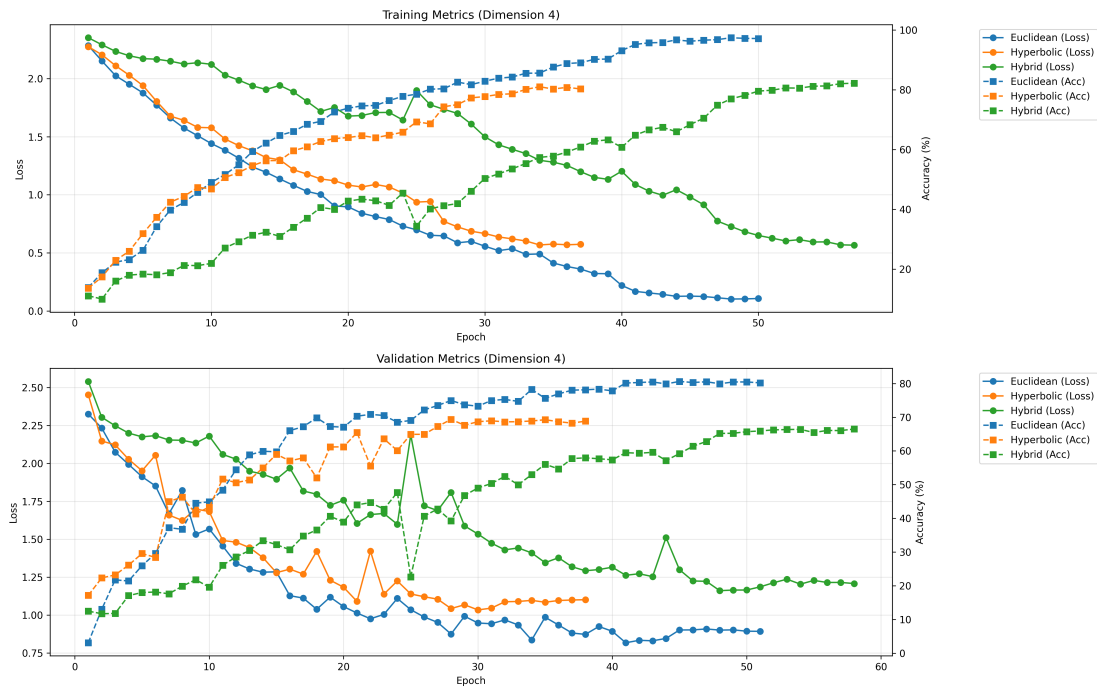
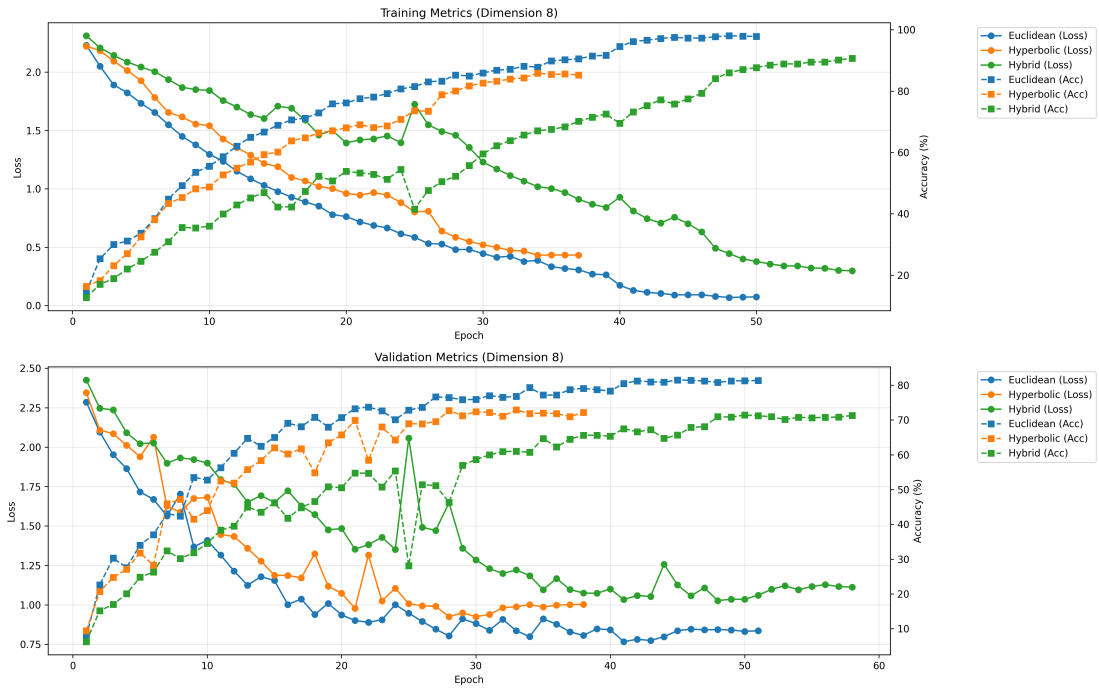
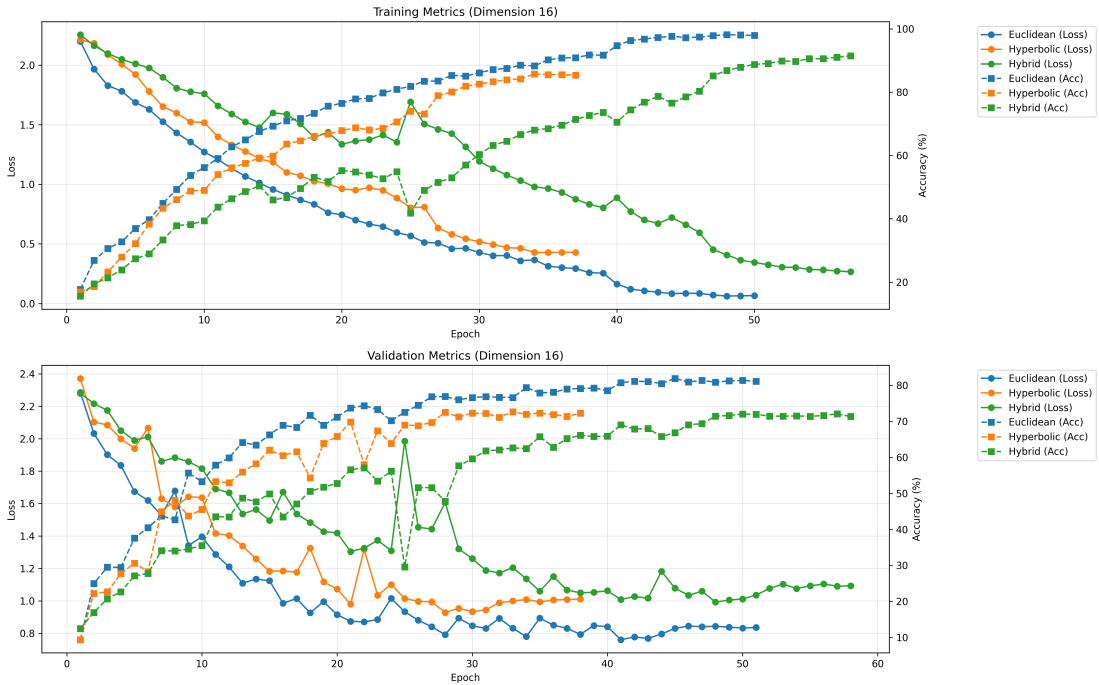


Figura A.65: Comparación entrenamiento EfficientNetV2 sobre Imagenette.

Las figuras a continuación presentan la comparación detallada por dimensión.

Figura A.66: Comparación entrenamiento EfficientNetV2 sobre Imagenette para $d = 4$.

Figura A.67: Comparación entrenamiento EfficientNetV2 sobre Imagenette para $d = 8$.Figura A.68: Comparación entrenamiento EfficientNetV2 sobre Imagenette para $d = 16$.

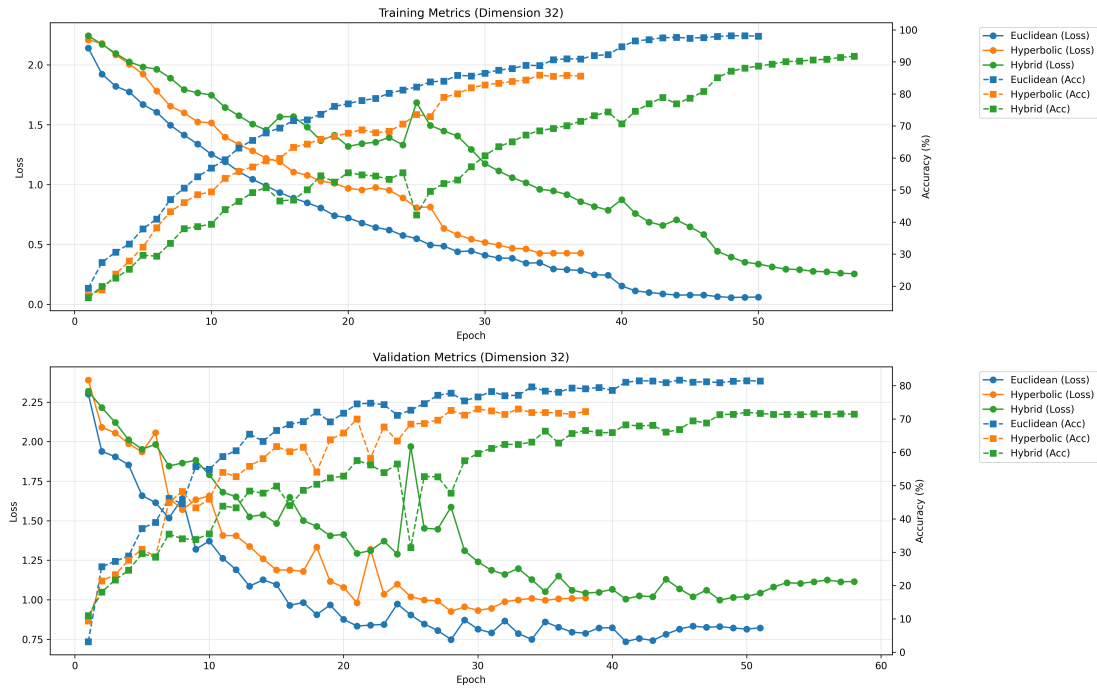


Figura A.69: Comparación entrenamiento EfficientNetV2 sobre Imagenette para $d = 32$.

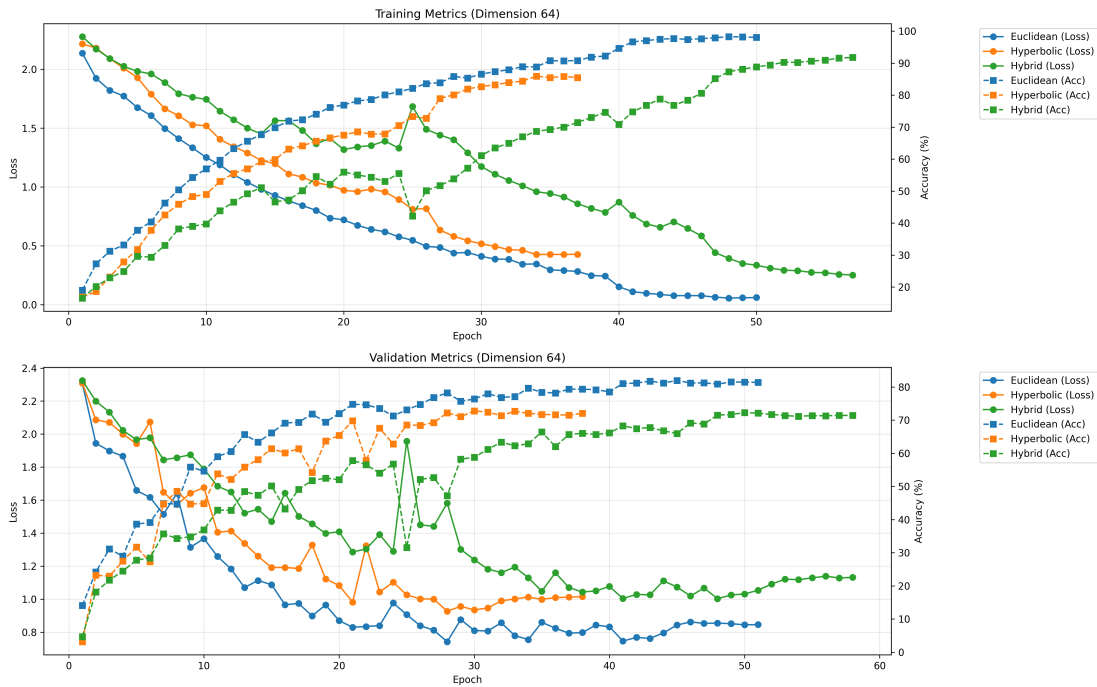
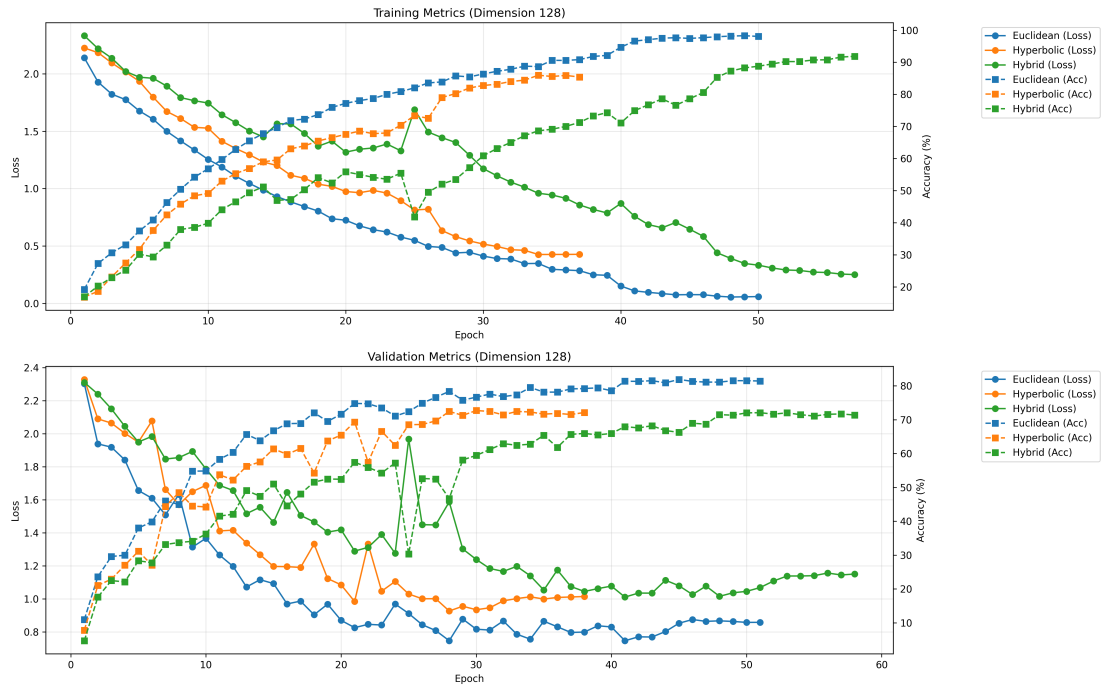
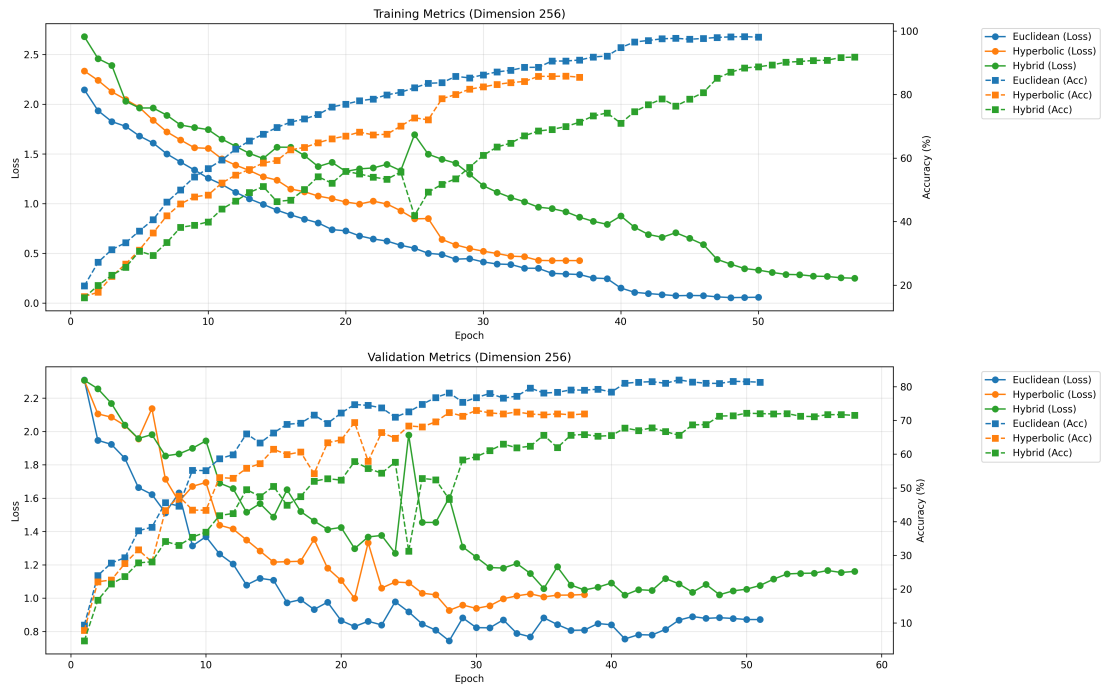


Figura A.70: Comparación entrenamiento EfficientNetV2 sobre Imagenette para $d = 64$.

Figura A.71: Comparación entrenamiento EfficientNetV2 sobre Imagenette para $d = 128$.Figura A.72: Comparación entrenamiento EfficientNetV2 sobre Imagenette para $d = 256$.

Apéndice B

Figuras DeepCluster

A continuación, se presentan todas las figuras obtenidas durante el entrenamiento del método DeepCluster.

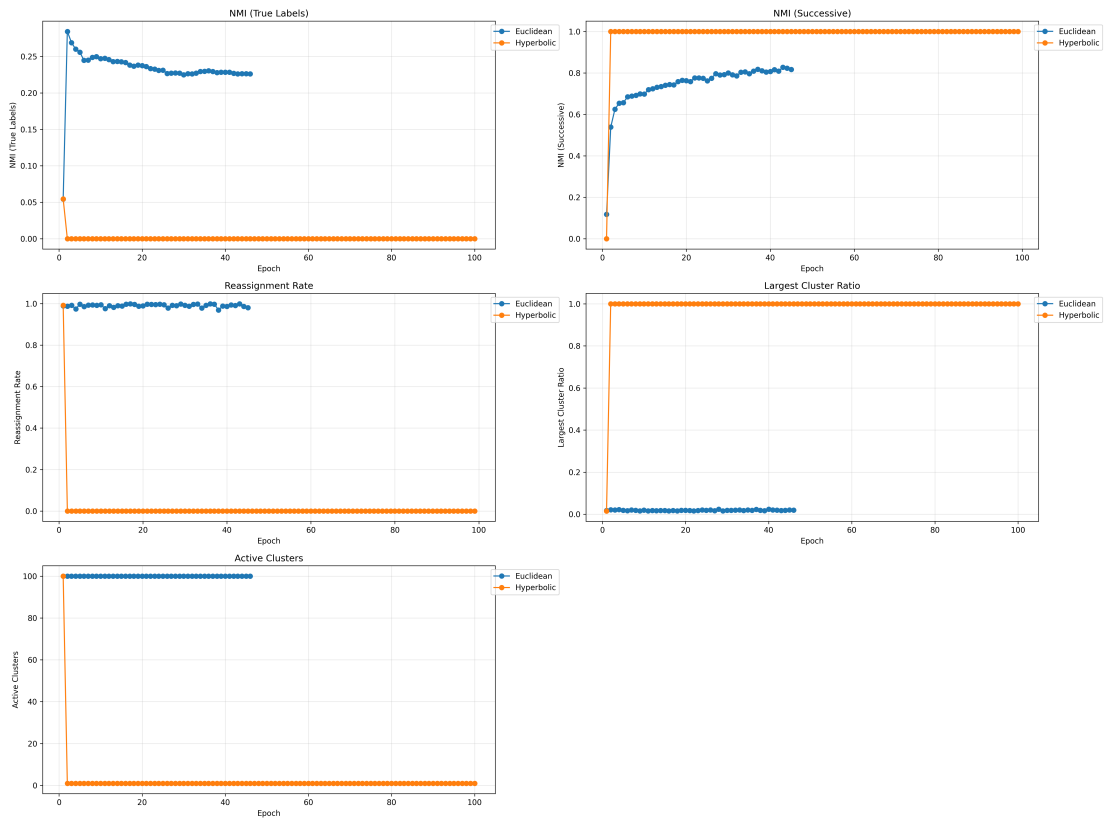


Figura B.1: Resultados entrenamiento ResNet50 sobre CIFAR10.

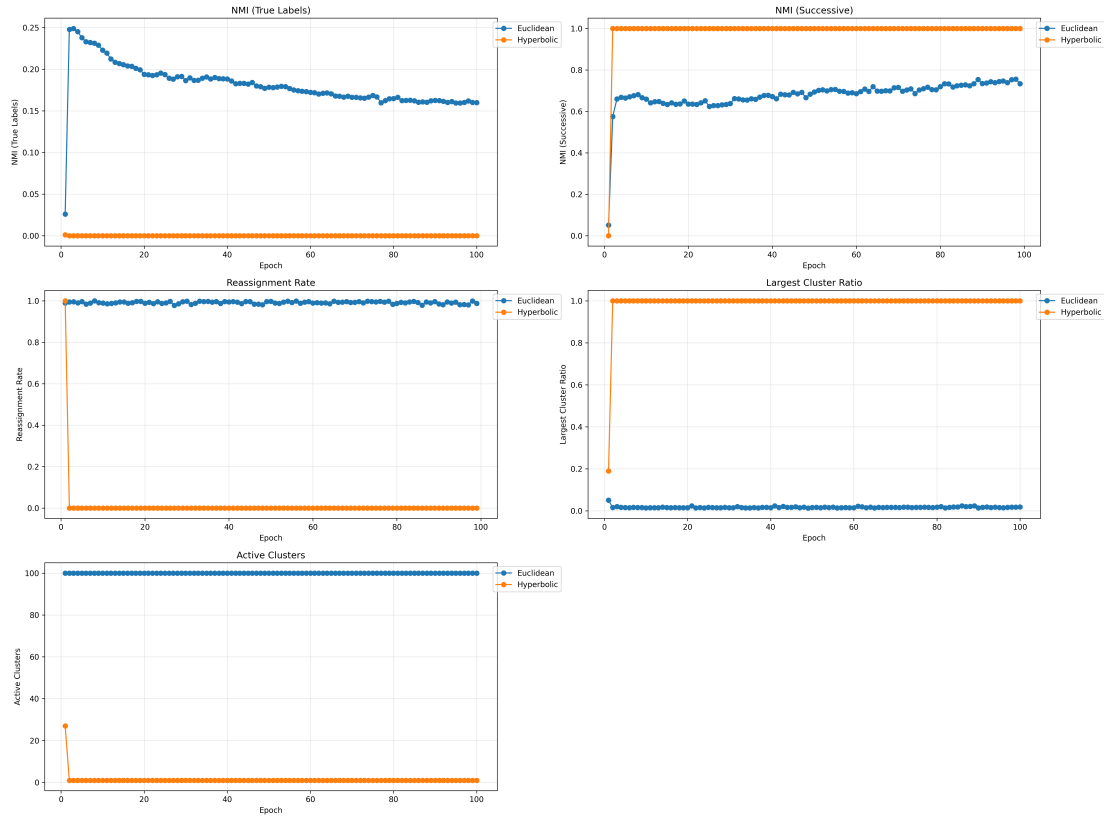


Figura B.2: Resultados entrenamiento EfficientNetV2 sobre CIFAR10.

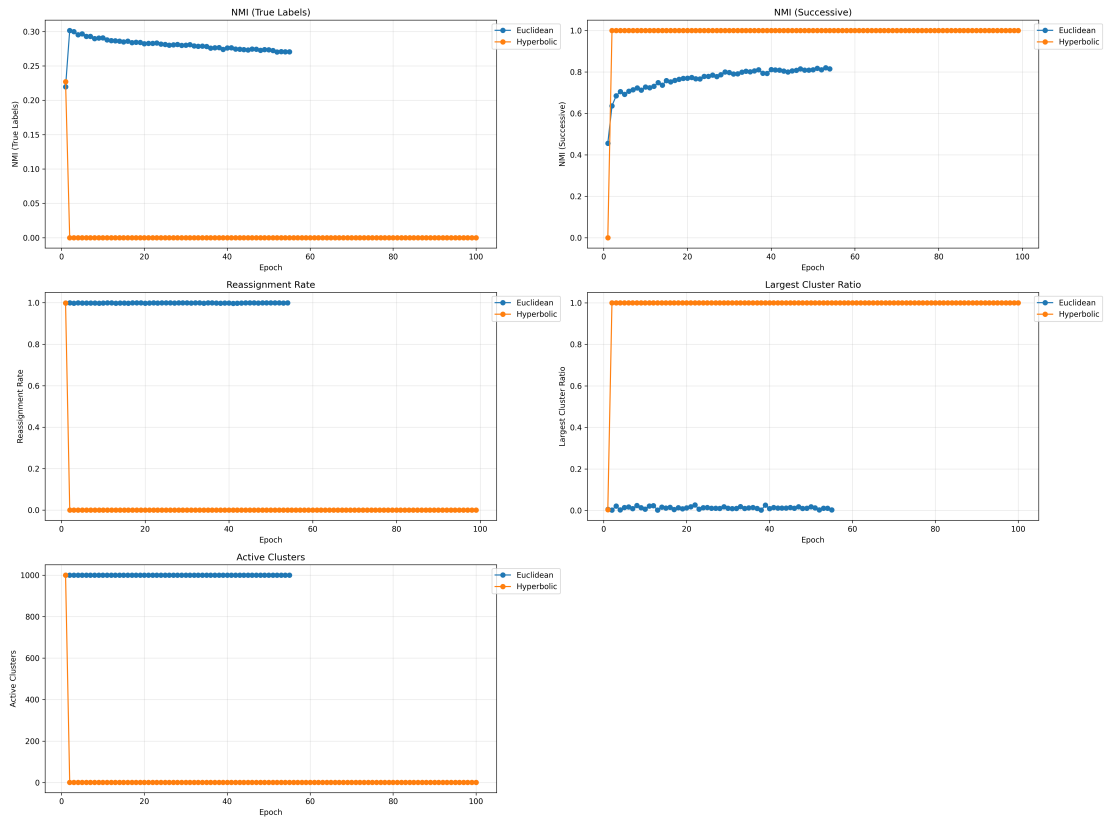


Figura B.3: Resultados entrenamiento ResNet50 sobre CIFAR100.

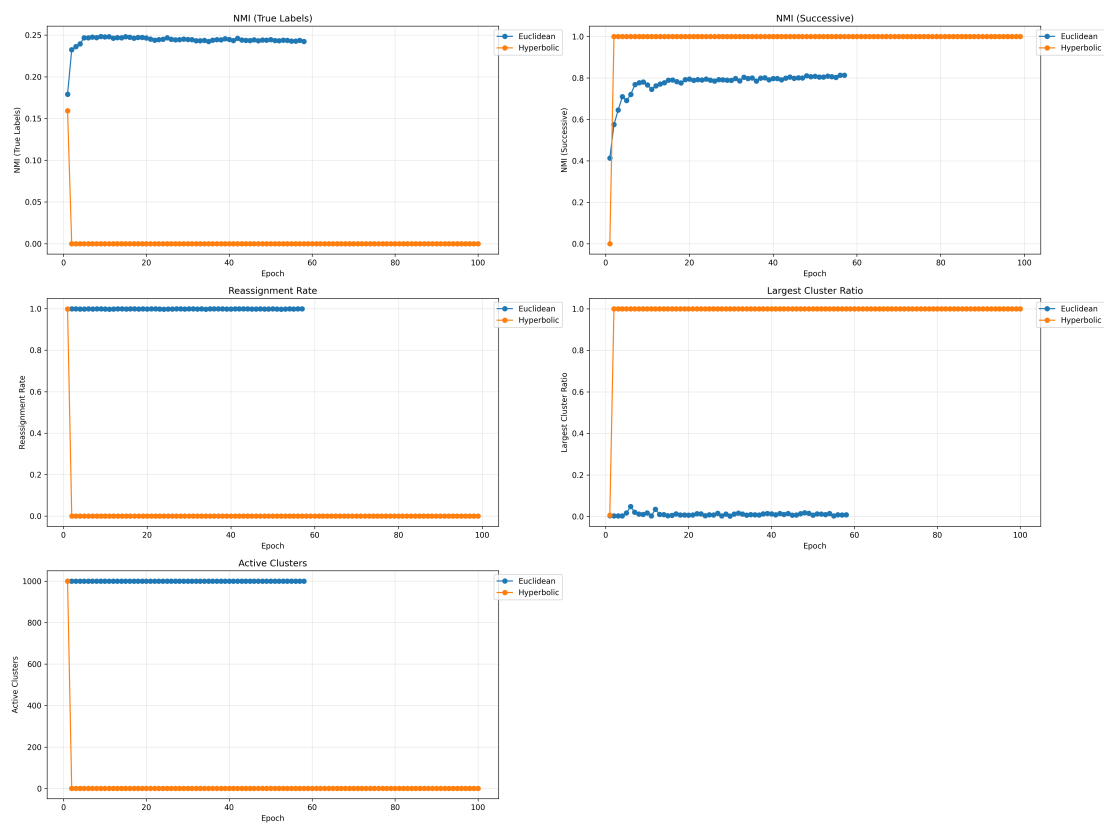


Figura B.4: Resultados entrenamiento EfficientNetV2 sobre CIFAR100.

Bibliografía

- [1] K. Anitha, R. Dhanalakshmi, Kavın Naresh, and Rukmani Devi. Hyperbolic hopfield neural networks for image classification in content-based image retrieval. *International Journal of Wavelets, Multiresolution and Information Processing*, 19:2050059, 10 2020.
- [2] Silvére Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, 2013.
- [3] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features, 2019.
- [4] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments, 2020.
- [5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers, 2021.
- [6] Ines Chami, Rex Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. *CoRR*, abs/1910.12933, 2019.
- [7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.
- [8] Karan Desai, Maximilian Nickel, Tanmay Rajpurohit, Justin Johnson, and Ramakrishna Vedantam. Hyperbolic image-text representations, 2024.
- [9] Aleksandr Ermolov, Leyla Mirvakhabova, Valentin Khrulkov, Nicu Sebe, and Ivan Oseledets. Hyperbolic vision transformers: Combining improvements in metric learning, 2022.
- [10] Octavian Ganea, Gary Becigneul, and Thomas Hofmann. Hyperbolic entailment cones for learning hierarchical embeddings. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th*

- International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1646–1655. PMLR, 10–15 Jul 2018.
- [11] Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. *CoRR*, abs/1805.09112, 2018.
- [12] Vivien Sainte Fare Garnot and Loïc Landrieu. Metric-guided prototype learning. *CoRR*, abs/2007.03047, 2020.
- [13] M. Gromov. *Hyperbolic Groups*, pages 75–263. Springer New York, New York, NY, 1987.
- [14] Xifeng Guo, Xinwang Liu, En Zhu, and Jianping Yin. Deep clustering with convolutional autoencoders. In Derong Liu, Shengli Xie, Yuanqing Li, Dongbin Zhao, and El-Sayed M. El-Alfy, editors, *Neural Information Processing*, pages 373–382, Cham, 2017. Springer International Publishing.
- [15] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*, volume 2, pages 1735–1742, 2006.
- [16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.
- [17] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.
- [18] Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan V. Oseledets, and Victor S. Lempitsky. Hyperbolic image embeddings. *CoRR*, abs/1904.02239, 2019.
- [19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [20] Alex Krizhevsky. Learning multiple layers of features from tiny images, 2009.
- [21] Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. Matryoshka representation learning, 2024.
- [22] Guy Lebanon and John Lafferty. Hyperplane margin classifiers on the multinomial manifold. *Twenty-first international conference on Machine learning; - ICML ’04*, 2004.
- [23] Nicholas Monath, Manzil Zaheer, Daniel Silva, Andrew McCallum, and Amr Ahmed. Gradient-based hierarchical clustering using continuous representations of trees in hyperbolic space, 2019.

-
- [24] Maximilian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations, 2017.
- [25] Ian Roberts, Mauricio Araya, Ricardo Nanculef, and Mario Mallea. Recovering latent hierarchical relationships in image datasets through hyperbolic embeddings. In Ruber Hernández-García, Ricardo J. Barrientos, and Sergio A. Velastin, editors, *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 92–103, Cham, 2025. Springer Nature Switzerland.
- [26] Rik Sarkar. Low distortion delaunay embedding of trees in hyperbolic plane. In Marc van Kreveld and Bettina Speckmann, editors, *Graph Drawing*, pages 355–366, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
- [27] Ryohei Shimizu, Yusuke Mukuta, and Tatsuya Harada. Hyperbolic neural networks++. *CoRR*, abs/2006.08210, 2020.
- [28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Machine Learning Representations - ICLR*, 2014.
- [29] Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training, 2021.
- [30] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis, 2016.
- [31] Dingyi Zhang, Yingming Li, and Zhongfei Zhang. Deep metric learning with spherical embedding, 2020.
- [32] Xinqi Zhu and Michael Bain. B-CNN: branch convolutional neural network for hierarchical classification. *CoRR*, abs/1709.09890, 2017.
- [33] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *CoRR*, abs/1911.02685, 2019.