TECMILENIO

Nombre:	Luis de la Garza González
Matrícula:	al03101869
Nombre del curso:	Aprendizaje Profundo.
Trabajo:	Actividad 04.- Exploración conceptual, análisis crítico y práctica de CNN.
Nombre del profesor:	Rodrigo Francisco Román Godínez
Fecha:	02.15.2026

Contenido

Instrucciones 3

Objetivo 3

Desarrollo 4

1. Mapa conceptual sobre arquitecturas y componentes 4

2. Análisis crítico de un caso real con CNN. 6

CNNs aplicadas al diagnóstico médico. 6

Descripción del problema a resolver, la arquitectura utilizada, los beneficios y limitaciones. 9

Tareas típicas de deep learning en enfermedad carotídea 10

Arquitecturas de deep learning más usadas 12

Dónde está más madura la IA hoy 13

Segmentación carotídea 13

Modelos concretos y tendencias técnicas 14

Qué limita hoy el salto a la práctica clínica 14

Flujo de trabajo típico de un proyecto de deep learning en enfermedad carotídea 16

Retos y limitaciones actuales 17

Líneas de desarrollo futuro 18

3. Implementación comparativa en código. 19

Conclusiones 24

1. Rendimiento final en el set de prueba 24

2. Comportamiento durante el entrenamiento 24

3. Tiempos de entrenamiento 25

4. Gráficas de pérdida y precisión 25

5. Conclusión general 26

Liga al Código 26

Instrucciones

El informe debe abordar las siguientes secciones:

1. **Mapa conceptual sobre arquitecturas y componentes**

Elabora un mapa conceptual que incluya las diferencias entre aprendizaje profundo y aprendizaje automático tradicional, arquitecturas principales, componentes esenciales y factores que impulsan el Aprendizaje Profundo.

2. **Análisis crítico de un caso real con CNN**
   - Investiga un caso reciente donde las CNNs se hayan aplicado en diagnóstico médico, vehículos autónomos y agricultura inteligente.
   - Describe el problema que resolvió, la arquitectura utilizada, los beneficios y limitaciones.
   - Reporta los hallazgos en un informe.
3. **Implementación comparativa en código**
   - Implementa dos modelos CNN con TensorFlow o PyTorch:
     - Uno básico (similar al ejemplo del tema 10).
     - Otro con mejoras (Dropout, Batch Normalization, optimizador distinto).
   - Entrena ambos con el dataset Fashion-MNIST y compara la precisión, el tiempo de entrenamiento y las gráficas de pérdida y precisión.
   - Crea un notebook en Jupyter o Google Colab con código comentado.
   - Reporta los resultados y el análisis realizado.

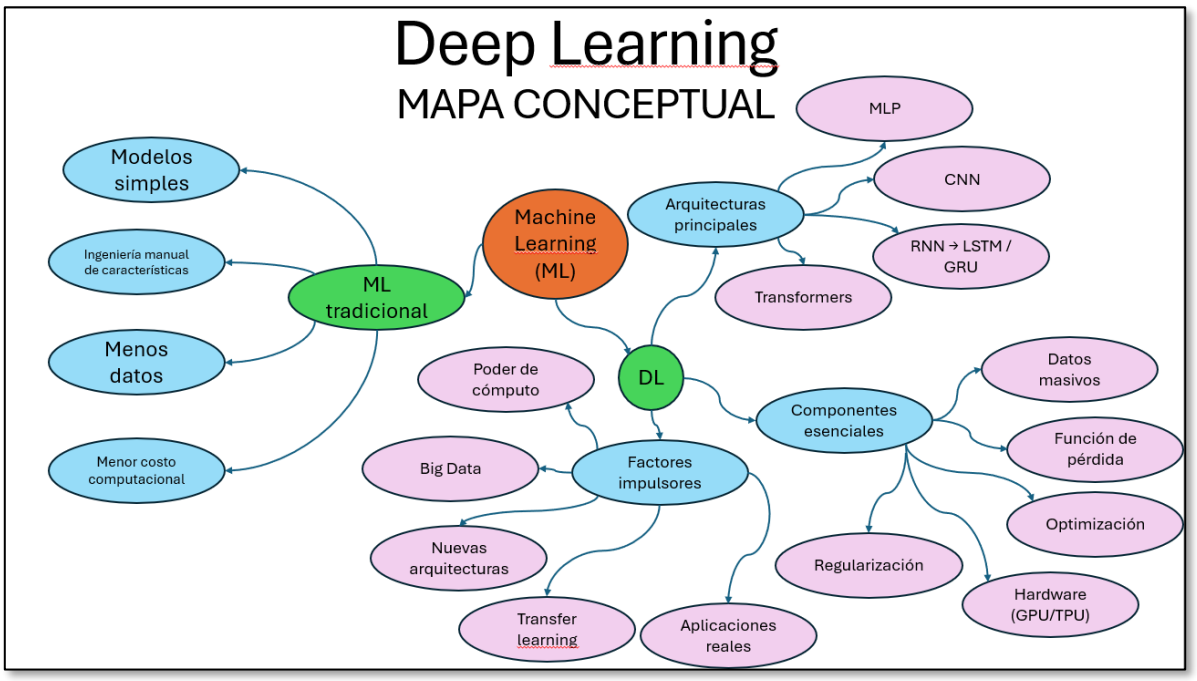
Objetivo

Analizar los principios del aprendizaje profundo y comprender el funcionamiento y aplicaciones de las redes neuronales convolucionales (CNN). A partir de la investigación y los contenidos revisados, elaborar un informe que integre conceptos teóricos, análisis crítico y ejemplos prácticos mediante implementación en código.

Desarrollo

1. Mapa conceptual sobre arquitecturas y componentes

La siguiente imagen muestra un mapa conceptual de aprendizaje profundo y su comparación con el aprendizaje automático tradicional.



APRENDIZAJE AUTOMÁTICO TRADICIONAL

- Modelos simples (SVM, regresión)
- Ingeniería manual de características
- Menos datos
- Menor costo computacional

APRENDIZAJE PROFUNDO

- Arquitecturas principales
  - MLP
  - CNN
  - RNN → LSTM / GRU
  - Transformers (atención)
- Componentes esenciales
  - Datos masivos
  - Función de pérdida

    - Optimización (backprop)
    - Hardware (GPU/TPU)
    - Regularización
- Factores impulsores
    - Poder de cómputo
    - Big Data
    - Nuevas arquitecturas
    - Transfer learning
    - Aplicaciones reales

2. Análisis crítico de un caso real con CNN.

CNNs aplicadas al diagnóstico médico.

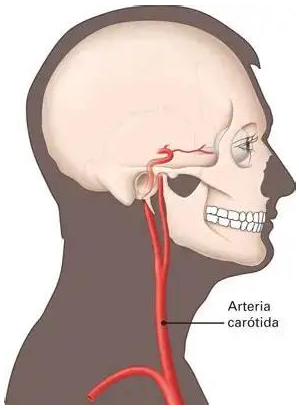
Como caso real para realizar un análisis crítico con CNN se selecciona el "diagnóstico de enfermedad carotidea".

Primero definimos que es la enfermedad carotidea y sus características.

¿Qué es la enfermedad carotídea?

La enfermedad carotídea es un proceso en el que las arterias carótidas —los vasos principales que llevan sangre oxigenada al cerebro— se van estrechando u obstruyendo debido a la aterosclerosis.

La aterosclerosis es la acumulación progresiva de grasa, colesterol, calcio y tejido fibroso en la pared arterial, formando lo que se conoce como placa ateromatosa.



La imagen muestra la ubicación de la arteria carótida derecha, se puede observar a nivel del cuello la bifurcación de la arteria carótida común en arteria carótida interna y arteria carótida externa

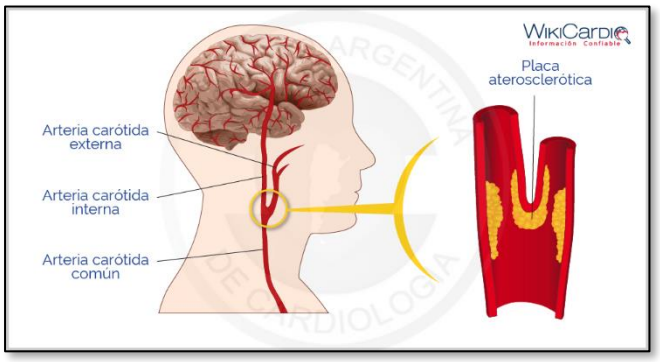
¿Por qué es importante?

Porque la enfermedad carotídea es una de las principales causas de ictus isquémico.

El ictus ocurre cuando una parte del cerebro deja de recibir sangre, ya sea por:

- Estenosis severa: la arteria se estrecha tanto que disminuye el flujo sanguíneo.
- Tromboembolismo: fragmentos de la placa o coágulos viajan al cerebro y bloquean arterias más pequeñas.
- Rotura de placa: una placa "vulnerable" puede romperse y generar un coágulo súbito.

La siguiente imagen muestra una arteria carótida con placa.



¿Qué ocurre dentro de la arteria?

1. Formación de placa

La pared arterial se inflama y acumula lípidos. Con el tiempo, la placa puede:

- crecer hacia el lumen (estrechando la arteria),
- calcificarse,
- ulcerarse,
- sangrar internamente (hemorragia intraplaca).

2. Estenosis

El lumen se estrecha. Se suele clasificar en:

Grado de estenosis	Descripción
Leve	**<50%**
Moderada	**50–69%**
Severa	**≥70%**
Oclusión	**100%**

3. Placa vulnerable

No todas las placas peligrosas están muy estrechas. Algunas tienen características que aumentan el riesgo de ictus:

- núcleo lipídico grande
- capa fibrosa delgada
- inflamación
- hemorragia intraplaca
- ulceración

Síntomas

Muchas personas no tienen síntomas hasta que ocurre un evento.

Cuando aparecen, pueden incluir:

- **AIT (ataque isquémico transitorio): síntomas neurológicos que duran minutos u horas.**
- **Ictus: déficit neurológico persistente.**
- **Síntomas oculares: pérdida súbita de visión en un ojo (amaurosis fugaz).**

Diagnóstico

Las herramientas más comunes son:

- Ecografía Doppler: primera línea, evalúa flujo y grado de estenosis.
- Angio-TC (CTA): excelente para ver calcificaciones y medir estenosis.
- Angio-RM (MRA): útil para caracterizar la placa.
- Angiografía: estándar de referencia, aunque más invasiva.

¿Por qué se produce?

Los factores de riesgo son los mismos de la aterosclerosis:

- hipertensión
- colesterol elevado
- diabetes
- tabaquismo
- edad avanzada
- obesidad
- sedentarismo
- antecedentes familiares

Tratamiento:

Depende del grado de estenosis y de si hay síntomas:

- Tratamiento médico: antiagregantes, estatinas, control de factores de riesgo.

- Endarterectomía carotídea: cirugía para retirar la placa.

- Stent carotídeo: alternativa mínimamente invasiva en casos seleccionados.

Descripción del problema a resolver, la arquitectura utilizada, los beneficios y limitaciones.

Introducción al problema clínico

La enfermedad carotídea (estenosis u oclusión de las arterias carótidas) es una causa mayor de ictus isquémico. El reto clínico es:

- **Detectar temprano**: identificar placas ateroscleróticas y su grado de estenosis antes de que produzcan eventos.
- **Caracterizar el riesgo**: no solo cuánto se estrecha la luz, sino cuán "vulnerable" es la placa (lípidos, hemorragia intraplaca, rotura de la capa fibrosa).
- **Tomar decisiones terapéuticas**: vigilar, intensificar tratamiento médico o indicar endarterectomía/stent.

El deep learning entra justo aquí: automatiza y mejora la detección, cuantificación y caracterización de la enfermedad a partir de imágenes médicas.

Modalidades de imagen usadas en enfermedad carotídea

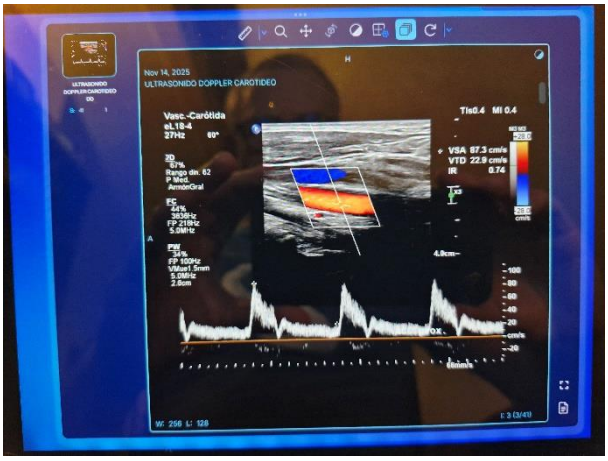
1. Ecografía Doppler (US)

- Ventajas: barata, disponible, sin radiación, primera línea.

- Información: grosor íntima-media (GIM), presencia de placa, grado de estenosis, flujo.

- Reto para DL: imágenes ruidosas, gran variabilidad entre operadores, ángulos y presets.

Ejemplo de imagen de ecografía Doopler:



2. Angio-TC (CTA)

- Ventajas: buena resolución espacial, visualiza calcificaciones, lumen y pared.

- Uso: cuantificación de estenosis, planificación de cirugía o stent.

- Reto: artefactos por calcio, necesidad de contraste y radiación.

3. Angio-RM (MRA) y RM de pared carotídea

- Ventajas: mejor caracterización tisular de la placa (lípidos, hemorragia, trombo).

- Uso: evaluación de "placa vulnerable", no solo del grado de estenosis.

- Reto: menor disponibilidad, protocolos más largos, variabilidad entre equipos.

4. Otras: PET/CT, PET/MR para inflamación de placa, menos frecuentes pero muy interesantes para modelos multimodales.

Ejemplo de imagen de resonancia magnética:

Tareas típicas de deep learning en enfermedad carotídea

Clasificación

Objetivo: decidir si hay o no enfermedad significativa, o clasificar el grado de estenosis (p.ej. <50 %, 50–69 %, ≥70 %).

- Entrada:

- Imágenes 2D (cortes de US o TC).
- Volúmenes 3D (CTA/MRA).

- Salida:

- Etiqueta binaria (enfermedad vs no enfermedad).
- Etiquetas ordinales (leve, moderada, severa).

Modelos típicos: CNN 2D/3D, a veces con atención (attention mechanisms) para resaltar la región carotídea.

Segmentación

- Objetivo: delinear estructuras: lumen, pared, placa, componentes de la placa.

- Aplicaciones:

  - Cálculo automático del porcentaje de estenosis.
  - Medición de volumen de placa.
  - Análisis de morfología (ulceraciones, irregularidad).

Modelos típicos: U-Net y variantes (2D, 2.5D, 3D), a veces con bloques residuales o atención.

Detección y localización

- Objetivo: localizar automáticamente la bifurcación carotídea, la placa o segmentos con estenosis.

- Modelos:

  - Detectores tipo YOLO, Faster R-CNN para 2D.
  - Redes 3D con “region proposal” para volúmenes.

Caracterización de placa (fenotipado)

- **Objetivo**: clasificar la placa según su composición (fibrosa, lipídica, calcificada, mixta) o su vulnerabilidad (estable vs vulnerable).

- **Entrada**: secuencias multiparamétricas de RM, CTA con diferentes ventanas, o incluso datos clínicos combinados.

- **Salida**: probabilidad de que la placa sea de alto riesgo.

Aquí se usan modelos multimodales (imágenes + datos clínicos) y, cada vez más, arquitecturas tipo transformers para integrar información heterogénea.

Arquitecturas de deep learning más usadas

1. CNN 2D y 3D

- 2D CNN: aplicadas a cortes axiales, sagitales o longitudinales (muy común en ecografía).
- 3D CNN: aprovechan el contexto volumétrico en CTA/MRA, mejorando la detección de estenosis y la segmentación de la placa.

2. U-Net y variantes

- U-Net clásica: encoder–decoder con “skip connections”, muy usada para segmentar lumen y pared.

- Variantes: U-Net++ (más densa), Attention U-Net (resalta regiones relevantes), 3D U-Net (para volúmenes completos).

3. Modelos híbridos CNN + RNN / Transformers

- CNN + LSTM/GRU: en ecografía, para secuencias de video Doppler o B-mode, capturando la dinámica del flujo.

- Vision Transformers (ViT): dividen la imagen en “patches” y usan auto-atención para aprender relaciones espaciales más globales.

4. Modelos multimodales

- Combinan:

  - Imágenes: CTA, MRA, US.

  - Datos clínicos: edad, sexo, factores de riesgo, síntomas.

- Se suelen fusionar en capas densas finales o mediante mecanismos de atención cruzada.

Dónde está más madura la IA hoy

Tabla rápida de estado del arte

Tarea	Modalidad principal	Modelos dominantes	Nivel de madurez clínica*
Detección de placa/estenosis	US, CTA	CNN 2D/3D, U-Net	Medio
Segmentación de carótida	US, CTA, RM	U-Net y variantes	Medio–alto (investigación avanzada)
Cuantificación de GIM	US	CNN + segmentación lineal	Medio
Placa vulnerable vs estable	CTA, RM de pared	CNN/3D-CNN, modelos híbridos	Prometedor, pero inmaduro
Predicción de eventos	Imagen + datos clínicos	Modelos multimodales	Bajo (muy experimental)

***Madurez clínica = qué tan cerca está de uso rutinario, no solo a nivel de "paper".**

Placa vulnerable: el foco caliente del momento

La línea más activa ahora mismo es la identificación de placa inestable/vulnerable (no solo cuánta estenosis hay).
Una revisión sistemática y metaanálisis reciente sobre IA para detectar placa vulnerable reporta:

- Sensibilidad ≈ 91 %
- Especificidad ≈ 84 %
- AUC ≈ 0.94

Es decir, en entornos controlados, los modelos (ML y DL) discriminan bastante bien entre placa estable e inestable.
Pero: casi no hay validación externa y la heterogeneidad entre estudios es muy alta, lo que limita la generalización**.**

Segmentación carotídea

En segmentación de carótida y placa (lumen, íntima-media, media-adventicia, volumen de placa), el estado del arte está dominado por:

- U-Net 2D/3D y variantes (U-Net++, Attention U-Net, Residual U-Net).
- Aplicadas a:

- Ecografía: grosor íntima-media (GIM), contornos de lumen y pared.
- CTA/RM: lumen, pared, componentes de placa.

Resultados típicos en estudios bien hechos:

- Dice para lumen/parede en rangos de 0.85–0.95.
- Errores de estenosis de solo unos pocos puntos porcentuales frente a expertos.

Pero de nuevo: la mayoría son datasets monocéntricos, con protocolos muy homogéneos.

Modelos concretos y tendencias técnicas

1) Detección de placa en CTA con DL

Modelos 3D-CNN o pipelines tipo:

- localización automática de la bifurcación carotídea,
- recorte de volumen,
- segmentación de placa,
- clasificación de vulnerabilidad.

2) Ecografía carotídea

- Segmentación automática de GIM y lumen.
- Clasificación de placa (ecolúcida, ecogénica, mixta) con CNN 2D.
- Uso de secuencias de video con CNN + LSTM para integrar dinámica del flujo.

3) Multimodalidad y datos clínicos

- Fusión de imagen + factores de riesgo (edad, HTA, DM, lípidos) en capas densas finales o con atención.
- Objetivo: pasar de “imagen bonita” a predicción de riesgo de ictus.

Qué limita hoy el salto a la práctica clínica

Aunque los números de AUC y Dice son muy buenos en artículos, el estado del arte tiene varios “peros” claros:

- Poca validación externa real (otros hospitales, otros equipos, otras poblaciones).
- Tamaños muestrales modestos y sesgos (p. ej., muchos casos severos, pocos leves).
- Falta de estandarización en protocolos de imagen y anotación.
- Interpretabilidad limitada: se necesitan mapas de calor, explicaciones y flujos que el clínico entienda.

- Escasez de estudios prospectivos que demuestren impacto en decisiones y resultados clínicos.

En resumen: el estado del arte en IA para enfermedad carotídea es muy prometedor en métricas técnicas, especialmente en:

- segmentación de carótida y placa,
- detección de placa vulnerable,

pero todavía está en fase de transición hacia herramientas robustas, validadas y reguladas para uso rutinario.

Flujo de trabajo típico de un proyecto de deep learning en enfermedad carotídea

1. Curación y anotación de datos

- Selección de casos: pacientes con estudios de US, CTA o MRA y diagnóstico confirmado (por angiografía, cirugía, seguimiento clínico).

- Etiquetado:

- Grado de estenosis (p.ej. NASCET).
- Segmentaciones de lumen, pared y placa.
- Etiquetas de vulnerabilidad (si se dispone de histología o RM avanzada).

Aquí el cuello de botella es el tiempo de los radiólogos/neurólogos para anotar.

2. Preprocesamiento

- Normalización de intensidades: especialmente en CTA/MRA.

- Registro y recorte: centrar la región de interés (bifurcación carotídea) para reducir ruido.

- Aumento de datos: rotaciones, traslaciones, cambios de brillo/contraste, ruido, para mejorar generalización.

3. Entrenamiento del modelo

- División de datos: entrenamiento, validación, prueba, idealmente por paciente (no por imagen) para evitar “leakage”.

- Pérdidas comunes:

- Clasificación: entropía cruzada, focal loss (si hay desbalance).

- Segmentación: Dice loss, combinación Dice + cross-entropy.

- Regularización: dropout, weight decay, early stopping.

4. Evaluación

- Clasificación:

- Exactitud, sensibilidad, especificidad, AUC-ROC.

- En contexto clínico, suele priorizarse alta sensibilidad para no perder estenosis significativas.

- Segmentación:

- Coeficiente de Dice, IoU, error de volumen.
- Diferencia en porcentaje de estenosis respecto a un experto.

- Impacto clínico:

- ¿Cambiaría la decisión terapéutica?
- ¿Reduce tiempo de lectura?
- ¿Mejora la reproducibilidad entre observadores?

5. Validación externa y despliegue

- Validación externa: en hospitales distintos, con equipos de imagen diferentes.

- Integración en PACS/RIS: el modelo debe integrarse en el flujo de trabajo del radiólogo, no ser una herramienta aislada.

- Interfaz: mostrar mapas de calor, segmentaciones superpuestas y medidas numéricas claras.

Retos y limitaciones actuales

1. Tamaño y calidad de los datasets

- Muchos estudios son monocéntricos y con pocos pacientes.

- Hay sesgos de selección (p.ej. solo casos severos o solo pacientes candidatos a cirugía).

- Falta de anotaciones detalladas de placa vulnerable.

2. Variabilidad entre centros y equipos

- Diferentes protocolos de US, CTA y MRA.

- Diferentes fabricantes y parámetros de adquisición.

- Esto puede hacer que un modelo funcione bien en un hospital y mal en otro.

3. Interpretabilidad

- Para que un neurólogo o cirujano vascular confíe en el modelo, necesita:

- Ver qué región de la imagen “miró” el modelo (mapas de atención, Grad-CAM).

- Entender por qué clasifica una placa como de alto riesgo.

4. Integración en la práctica clínica

- No basta con un buen AUC (Area Under the Curve) en un paper:
    - Hay que demostrar beneficio en flujo de trabajo, tiempos, costes y resultados clínicos.
    - Se requieren estudios prospectivos y, idealmente, ensayos clínicos.

5. Aspectos regulatorios y éticos

- Marcado regulatorio (p.ej. FDA) para uso clínico.
- Protección de datos, anonimización y gobernanza de IA.
- Evitar modelos entrenados en poblaciones muy específicas que luego se aplican a otras sin validación.

Líneas de desarrollo futuro

- Modelos auto-supervisados y foundation models de imagen médica: entrenados en grandes volúmenes de CTA/MRA/US para luego afinar en tareas carotídeas con pocos datos etiquetados.
- Aprendizaje federado: entrenar modelos entre múltiples hospitales sin compartir datos crudos, solo actualizaciones de parámetros.
- Predicción de eventos clínicos: no solo “grado de estenosis hoy”, sino riesgo de ictus a 1–3 años combinando imagen + clínica + laboratorio.
- Integración multimodal real: imagen carotídea + imagen cerebral (RM de perfusión, difusión) + datos de circulación colateral para una visión global del riesgo cerebrovascular.

3. Implementación comparativa en código.

- Se implementan dos modelos CNN con TensorFlow:
  - Uno básico.
  - Otro con mejoras (Dropout, Batch Normalization, optimizador distinto).
- Se crea un notebook en Colab con el código comentado.
- Se entrenan ambos modelos con el dataset Fashion-MNIST y se comparan la precisión, el tiempo de entrenamiento y las gráficas de pérdida y precisión.
- Se reportan los resultados y el análisis realizado.

Código

```
# importar las librerías necesarias

import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import fashion_mnist

import numpy as np
import matplotlib.pyplot as plt
import time
```

```
# Cargar Fashion-MNIST
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

# Normalizar a valores de 0 a 1
x_train = x_train / 255.0
x_test = x_test / 255.0

# Añadir canal (28,28,1)
x_train = x_train[..., np.newaxis]
x_test = x_test[..., np.newaxis]

print("Shape entrenamiento:", x_train.shape)
print("Shape prueba:", x_test.shape)
```

```
# Modelo 1 - Corresponde a la CNN básica
def build_basic_cnn():
    model = models.Sequential([
        # Conv2D (32 filtros, 3×3, ReLU)
        # Extrae bordes y patrones simples:
        layers.Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)),
        #  MaxPooling2D  (2×2)  Reduce  tamaño  y  retiene  características
importantes:
        layers.MaxPooling2D((2,2)),
        # Conv2D (64 filtros, 3×3, ReLU)  Aprende patrones más complejos:
        layers.Conv2D(64, (3,3), activation='relu'),
```

```
        # Reduce dimensionalidad:
        layers.MaxPooling2D((2,2)),
        # Convierte el mapa de características en un vector:
        layers.Flatten(),
        # Capa densa para combinar características:
        layers.Dense(128, activation='relu'),
        # Clasificación final en 10 clases:
        layers.Dense(10, activation='softmax')
    ])

    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

```
# Modelo 2 - CNN mejorada (Dropout + BatchNorm + optimizador distinto)
def build_improved_cnn():
    model = models.Sequential([
        layers.Conv2D(32,     (3,3),     padding='same',     activation='relu',
input_shape=(28,28,1)),
        layers.BatchNormalization(),
        layers.Conv2D(32, (3,3), activation='relu'),
        layers.MaxPooling2D((2,2)),
        layers.Dropout(0.25),

        layers.Conv2D(64, (3,3), padding='same', activation='relu'),
        layers.BatchNormalization(),
        layers.Conv2D(64, (3,3), activation='relu'),
        layers.MaxPooling2D((2,2)),
        layers.Dropout(0.25),

        layers.Flatten(),
        layers.Dense(256, activation='relu'),
        layers.BatchNormalization(),
        layers.Dropout(0.5),

        layers.Dense(10, activation='softmax')
    ])
    # Compilación
    # Optimizador: Adam
    # Pérdida: sparse_categorical_crossentropy
    # Métrica: accuracy
    model.compile(optimizer=tf.keras.optimizers.AdamW(learning_rate=1e-3),
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

```
# 3. Entrenamiento de ambos modelos
basic_model = build_basic_cnn()
improved_model = build_improved_cnn()
```

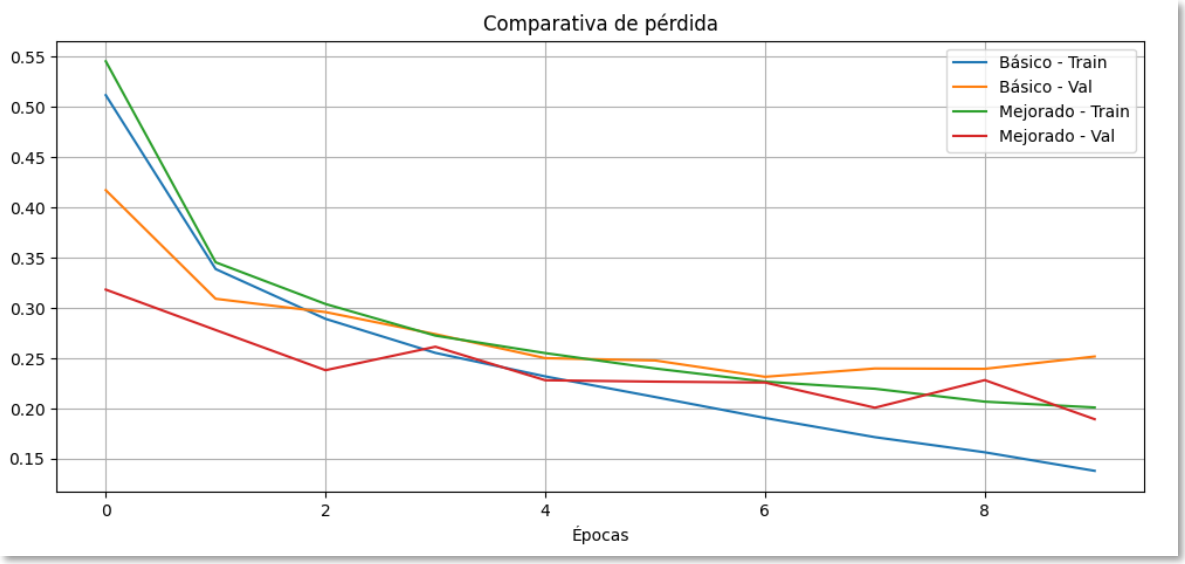
```
EPOCHS = 10
BATCH = 64
```

```
# Entrenamiento con medición de tiempo:

# Modelo básico
start = time.time()
history_basic = basic_model.fit(
    x_train, y_train,
    epochs=EPOCHS,
    batch_size=BATCH,
    validation_split=0.1,
    verbose=1
)
time_basic = time.time() - start

# Modelo mejorado
start = time.time()
history_improved = improved_model.fit(
    x_train, y_train,
    epochs=EPOCHS,
    batch_size=BATCH,
    validation_split=0.1,
    verbose=1
)
time_improved = time.time() - start
```

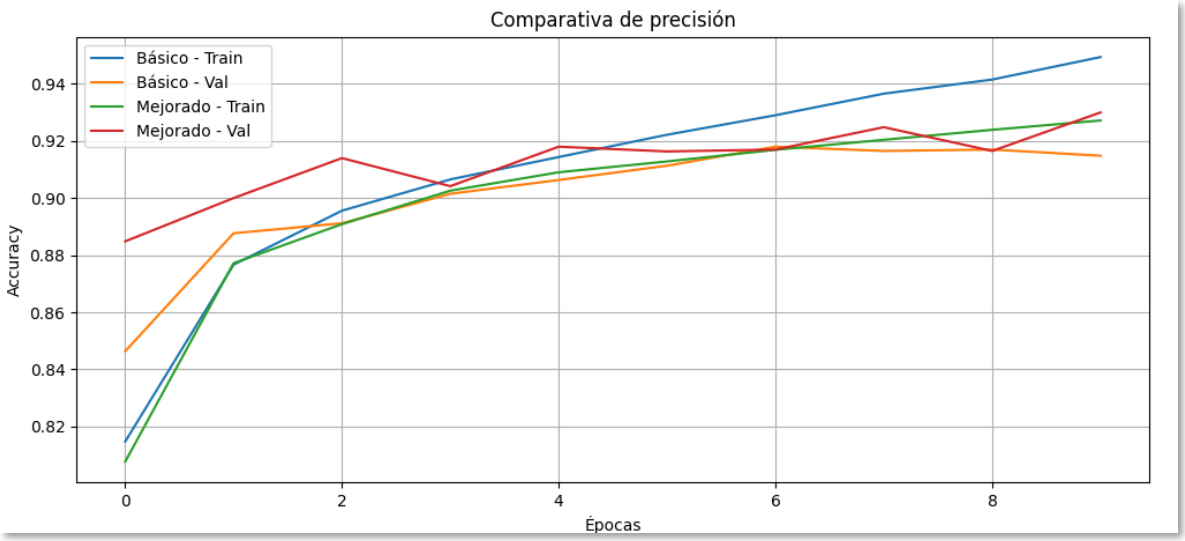
```
# 4. Evaluación en el set de prueba
test_basic = basic_model.evaluate(x_test, y_test, verbose=0)
test_improved = improved_model.evaluate(x_test, y_test, verbose=0)

print("Modelo básico - Loss:", test_basic[0], "Accuracy:", test_basic[1])
print("Modelo    mejorado    -    Loss:",    test_improved[0],    "Accuracy:",
test_improved[1])

print("Tiempo entrenamiento básico:", time_basic, "segundos")
print("Tiempo entrenamiento mejorado:", time_improved, "segundos")
```

```
Modelo básico - Loss: 0.2794267237186432 Accuracy: 0.9092000126838684

Modelo mejorado - Loss: 0.2079528570175171 Accuracy: 0.9230999946594238

Tiempo entrenamiento básico: 610.3689386844635 segundos

Tiempo entrenamiento mejorado: 2215.019844532013 segundos
```

```
# Gráficas comparativas:

plt.figure(figsize=(12,5))
plt.plot(history_basic.history['loss'], label='Básico - Train')
plt.plot(history_basic.history['val_loss'], label='Básico - Val')
plt.plot(history_improved.history['loss'], label='Mejorado - Train')
plt.plot(history_improved.history['val_loss'], label='Mejorado - Val')
plt.title("Comparativa de pérdida")
plt.xlabel("Épocas")
plt.ylabel("Loss")
plt.legend()
plt.grid(True)
plt.show()
```



```
# Precisión

plt.figure(figsize=(12,5))
plt.plot(history_basic.history['accuracy'], label='Básico - Train')
plt.plot(history_basic.history['val_accuracy'], label='Básico - Val')
plt.plot(history_improved.history['accuracy'], label='Mejorado - Train')
plt.plot(history_improved.history['val_accuracy'], label='Mejorado - Val')
plt.title("Comparativa de precisión")
plt.xlabel("Épocas")
plt.ylabel("Accuracy")
plt.legend()
plt.grid(True)
plt.show()
```



Conclusiones

Derivado del análisis de los resultados de la corrida con ambos modelos (básico y mejorado, basado directamente en el contenido del notebook en Colab [colab.research.google.com](https://colab.research.google.com/drive/11h9byjiWJMU5mRauKNrKUq6vePYowjbD), vemos que:

1. Rendimiento final en el set de prueba

Modelo	Accuracy	Loss
Básico	0.9092	0.2794
Mejorado	0.9231	0.2079

Podemos ver que:

- El modelo mejorado supera al básico en ambas métricas.

- La diferencia de accuracy (~1.4%) es modesta pero consistente.

- La diferencia en pérdida es más marcada, lo que indica que el modelo mejorado generaliza mejor.

Esto confirma que las mejoras (BatchNorm, Dropout, más filtros, AdamW) sí aportan valor real.

2. Comportamiento durante el entrenamiento

Modelo básico

- Converge rápido.

- La validación se estabiliza alrededor de 0.91–0.92.

- No muestra señales fuertes de sobreajuste.

- La pérdida de validación baja de forma suave y estable.

Modelo mejorado

- La validación llega a 0.93, lo cual es excelente para Fashion-MNIST.

- La pérdida de validación baja más que en el modelo básico.

- El entrenamiento es más ruidoso (normal con Dropout + BatchNorm).

- No hay sobreajuste significativo, lo cual indica que la regularización está funcionando.

Conclusión:

El modelo mejorado aprende más lentamente, pero termina en un punto más alto. Esto es típico de arquitecturas más profundas y regularizadas.

---

3. Tiempos de entrenamiento

Modelo	Tiempo total (seg)
Básico	610
Mejorado	2,215

El modelo mejorado tarda 3.6 veces más en entrenar, debido a:

- Cuenta con más capas convolucionales.

- BatchNorm (opera por canal y por batch).

- Dropout (más operaciones por batch).

- AdamW (más costoso que Adam estándar).

Conclusiones:

- Para producción o prototipos rápidos: el modelo básico es suficiente.

- Para maximizar la precisión: el mejorado es claramente superior.

4. Gráficas de pérdida y precisión

Pérdida

- El modelo mejorado tiene una pérdida más baja, tanto en entrenamiento como en validación.

- El modelo básico es más estable, pero no llega tan abajo.

Precisión

- El modelo mejorado supera al básico en casi todas las épocas.

- La validación del mejorado es más ruidosa, pero termina más arriba.

Conclusiones:

Las gráficas confirman que el modelo mejorado:

- Aprende más lentamente.

- Generaliza mejor.

- Tiene más variabilidad (normal con Dropout).

5. Conclusión general

El modelo mejorado es mejor en rendimiento, pero mucho más costoso en tiempo.

Si el objetivo es calidad del modelo, el mejorado es superior.

Si el objetivo es eficiencia, el básico es más que suficiente.

Liga al Código

https://github.com/luisgg121/DL-Actividad-004.git

Archivo: DL-Actividad-04.ipynb