UNIVERSIDAD DE LA FRONTERA

UNIVERSIDAD DE LA FRONTERA

Facultad de Ingeniería y Ciencias
Departamento de Ingeniería Matemática

Tarea 1 – IMA543

Redes Neuronales Avanzadas de Aprendizaje Profundo

Integrantes: Javier Luengo Abarzúa – Monserrath Álvarez Arriagada
Docente: Benjamin Higuera
Fecha: 6 de mayo de 2025

Introducción

El presente informe tiene objetivo el desarrollo de la realización de la Tarea 1 del curso IMA543 – Redes Neuronales Avanzadas de Aprendizaje Profundo. En esta tarea se aborda el análisis diseño e implementación de arquitecturas de redes neuronales, tanto del tipo Densely Connected Convolutional Networks (DenseNet), como del tipo Deep Residual Networks (ResNet)

A lo largo del documento, se describen los fundamentos teóricos empleados, las metodologías utilizadas y los resultados obtenidos en cada uno de los ejercicios planteados. El objetivo es basandose en los códigos vistos en clases, implementar y entrenar los modelos convolucionales densos —DenseNet y ResNet— utilizando el conjunto de datos **FER2013**.

1. Desarrollo DenseNet

En la implementacion de los 3 programas se utilizo la función `extraer_imagenes("FER")` y la función `lr_schedule`, donde la primera carga imagenes de 48x48 píxeles en escala de grises, las organiza en conjuntos de entrenamiento y prueba, normaliza los valores de píxeles dividiendolos en 255 y convierte las etiquetas a formato one hot (7 clases de emociones). La segunda función implementa una reducción programada del learning rate, donde comienza con 1e-3, para deducir por factores especificos espues de las epocas 80, 120, 160 y 180, esto evita el estancamiento del entrenamiento y permite un ajuste mejor.

Los modelos comparten la misma arquitectura tipo **DenseNet** compuesta por una capa de convolución inicial, seguida de cuatro bloqes densos, cada uno con 12 capas de tipo cuello de botella, que implican convoluciones **1x1** y **3x3**. Capas de transición entre bloques, qe comprimen el número de filtros mediante **compression_factor** espec´fico en cada modelo. Pooling global y unca capa densa fical de activación **softmax** para la clasificación milticlases.

Se consideran callbacks importantes como **ModelCheckpoint** que guarda el mejor modelo segun precisión, **LearningRateScheduler** aplixa la programación de tasa de aprendizaje y **ReducelLROnPlateau** que reduceel learning rate si la pérdida se estanca.

Los parametros utilizados para la realización de los modelos DeseNet son los siguientes:

Parámetro (código)	**Valor**
`batch_size`	32
`epochs`	200
`data_augmentation`	`True`
`num_classes`	7
`num_dense_blocks`	4
`growth_rate`	12
`depth`	100
`use_max_pool`	`False`
`num_bottleneck_layers`	12
`num_filters_bef_dense_block`	24
`optimizer`	RMSprop(1e-3)
`loss`	`'categorical_crossentropy'`
`metrics`	`['acc']`

Cuadro 1: Parámetros comunes utilizados en los tres modelos DenseNet

2. Resultados DenseNet

2.1. DenseNet con tasa de compresión 0.3

Luego de ejecutar el archivo en el servidor Khipu, se obtuvo una **Test accurasy de 0.6780** y unos graficos de exactitud y perdida:

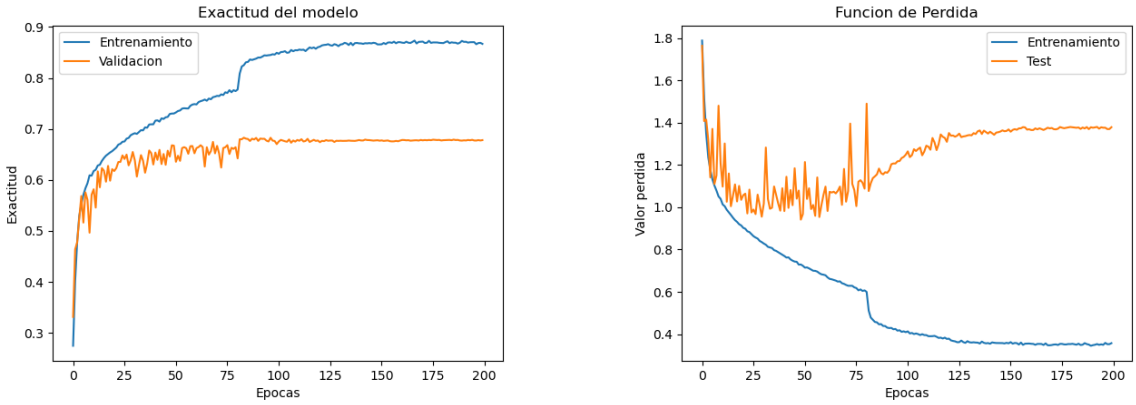

Figura 1: Curvas de exactitud y pérdida para DenseNet con compresión 0.3

2.2. DeseNet con tasa de compresión 0.5

Luego de ejecutar el archivo en el servidor Khipu, se obtuvo **Test accurasy de 0.6892** y unos graficos de exactitud y perdida:

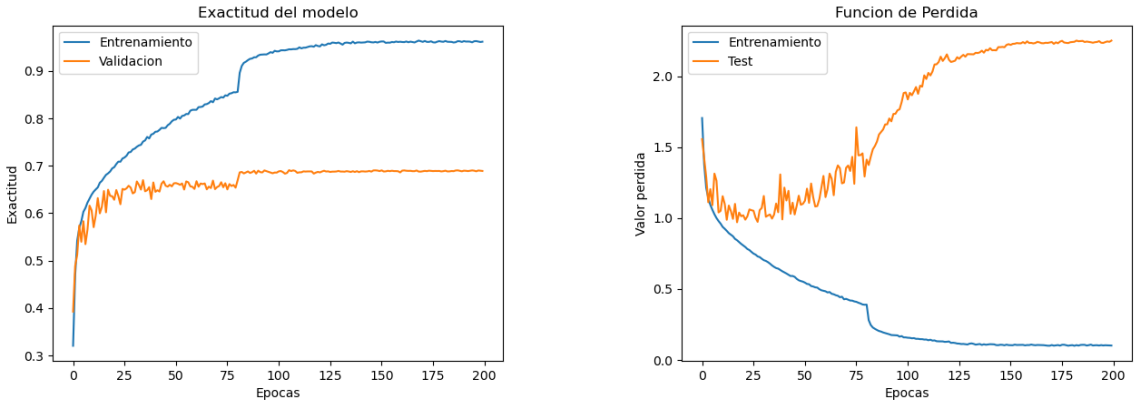

Figura 2: Curvas de exactitud y pérdida para DenseNet con compresión 0.5

2.3. DeseNet con tasa de compresión 0.7

Luego de ejecutar el archivo en el servidor Khipu, se obtuvo **Test accurasy de 0.6858** y unos graficos de exactitud y perdida:

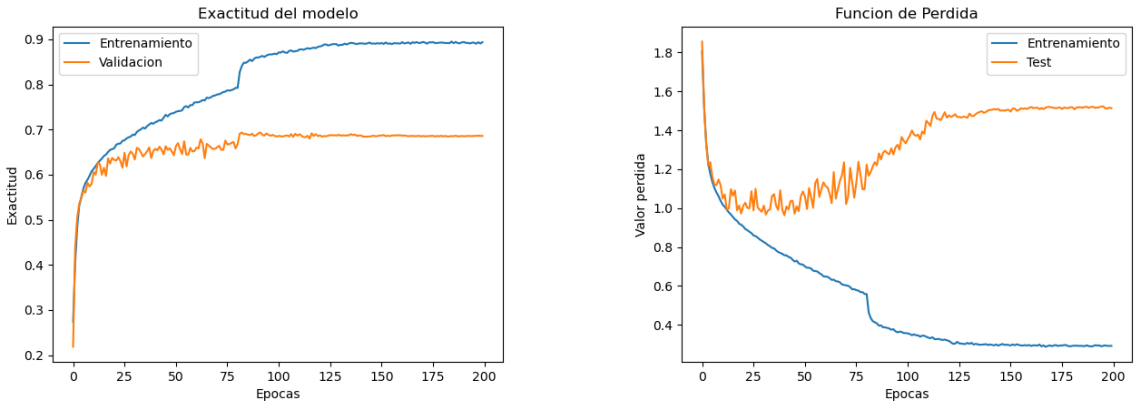

Figura 3: Curvas de exactitud y pérdida para DenseNet con compresión 0.7

3. Conclusiones DenseNet

En la Figura 1, se observa un aumento progresivo en la exactitud en el conjunto de entrenamiento, alcanzando un valor cercano a 0.92 desde aproximadamente la época 80 en adelante, sin embargo, la exactitud en validación se estabiliza en torno a 0.68 sin lograr mejorar significativamente, lo que sugiere un sobreajuste. En el gráfico de pérdida, la curva de entrenamiento disminuye de forma constante hasta llegando a 0.4, mientras que la pérdida en validación se mantiene elevada y con oscilaciones alrededor del 1.2. Esto puede significar que el modelo aprende con eficacia sobre el conjunto de entrenamiento pero no sobre datos no vistos.

En la Figura 2, al igual que el anterior, la exactitud en entrenamiento se eleva hasta aproximadamente 0.93 y se mantiene estable tras la época 100. La exactitud en validación se estanca cerca de 0.65 y no muestra mejoras después de la época 80 aproximadamente. En la pérdida, la curva de entrenamiento sigue una trayectoria descendente hasta llegar aproximadamente a 0.3. Para la pérdida en validación se alcanzan valores superiores a 2, lo que se ve un desajuste aún mayor respecto al modelo anterior, esto sugiere que, aunque la compresión intermedia reduce el número de parámetros, también disminuye la capacidad de generalización del modelo.

Por último, en la Figura 3 se aprecia un factor de compresión 0.7. En este caso, la exactitud en entrenamiento se estabiliza en torno a 0.9 desde la época 100, mientras que la validación se mantiene cerca de 0.65. La pérdida de entrenamiento se reduce progresivamente hasta alcanzar valores cercanos a 0.35, pero la pérdida en validación, si bien no es tan elevada como en el modelo anterior, se estabiliza en un nivel alto, alrededor de 1.5 y esto sugiere que al aumentar el nivel de compresión, el modelo tiene una menor efectividad para capturar patrones complejos en los datos.

En resumen, puede concluirse que los tres modelos comparten un comportamiento similar en cuanto a la capacidad de aprendizaje del entrenamiento, pero se diferencian en la capacidad de desempeño en la validación. El modelo con compresión 0.3 ofrece el mejor rendimiento, aunque con mayor riesgo de sobreajuste. El modelo con compresión 0.5 presenta el peor desempeño en validación, probablemente debido a una pérdida de capacidad significativa. En cambio, el modelo con compresión 0.7 muestra un comportamiento más estable, pero limitado, lo que indica que una compresión excesiva puede comprometer la eficacia del aprendizaje.

4. Desarrollo ResNet

Se utilizó la función **extraer_imagenes()** para cargar las imágenes desde carpetas separadas en entrenamiento y prueba, con 48x48 píxeles en escala de grises, luego se normalizo los valores dividiendo por 255 y codificar las etiquetas en formato **one-hot** para 7 clases.Tanto en ResNet v1 como en v2, se implemento el aprendizaje con **lr_schedule()** tal como para DenseNet.

En ambas implementaciones, la estructura de red neuronal es construida a partir de la función **resnet_layer()**, que define una pila de capas **Conv2D**, **BatchNormalization** y **Activation**, esta función se reutiliza múltiples veces. Las arquitecturas terminan con capas **AveragePooling2D**, **Flatten** y una capa **Dense** con activación **softmax** para clasificación final.

Asimismo, ambos modelos emplean los mismos **callbacks** de entrenamiento: **ModelCheckpoint** para guardar el mejor modelo en validación, **LearningRateScheduler** para aplicar la programación de tasa de aprendizaje, y **ReduceLROnPlateau** para reducir la tasa cuando la función de pérdida de validación se estabiliza.

Los parametros utilizados para la realización de los modelos ResNet son los siguientes:

Parámetro (código)	**Valor**
`batch_size`	64
`epochs`	200
`data_augmentation`	`True`
`n_clases`	7
`loss`	`'categorical_crossentropy'`
`optimizer`	`Adam(learning_rate=1e-3)`
`metrics`	`['accuracy']`
`input_shape`	`(48, 48, 1)`

Cuadro 2: Parámetros comunes utilizados en los modelos ResNet v1 y v2

5. Resultados ResNet

5.1. RseNet V1

Luego de ejecutar el archivo en el servidor Khipu, se obtuvo **Test accurasy de 0.6652** y unos graficos de exactitud y perdida:

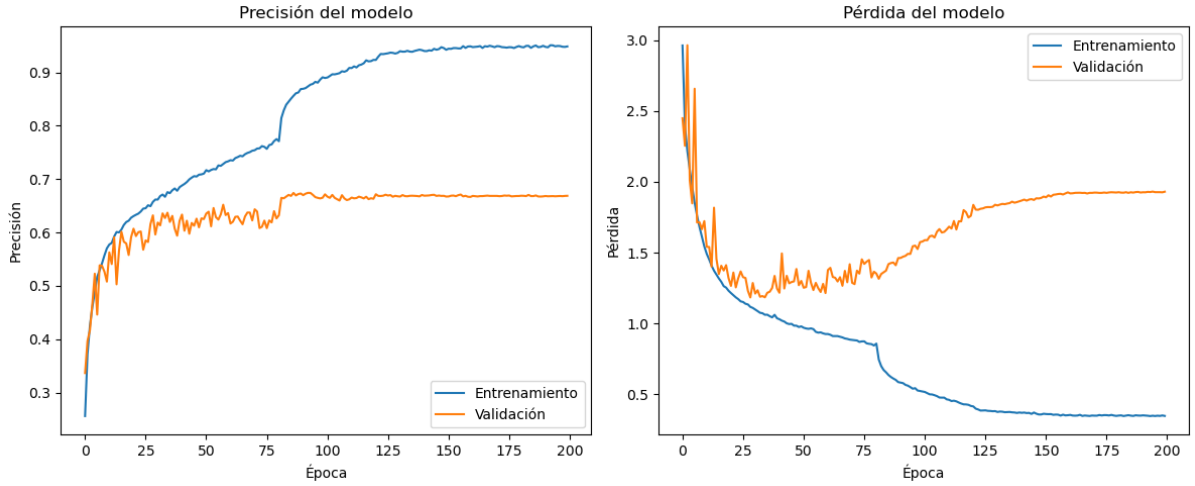

Figura 4: Curvas de exactitud y pérdida para ResNet V1

5.2. ReseNet V2

Luego de ejecutar el archivo en el servidor Khipu, se obtuvo **Test accurasy de 0.6625** y unos graficos de exactitud y perdida:

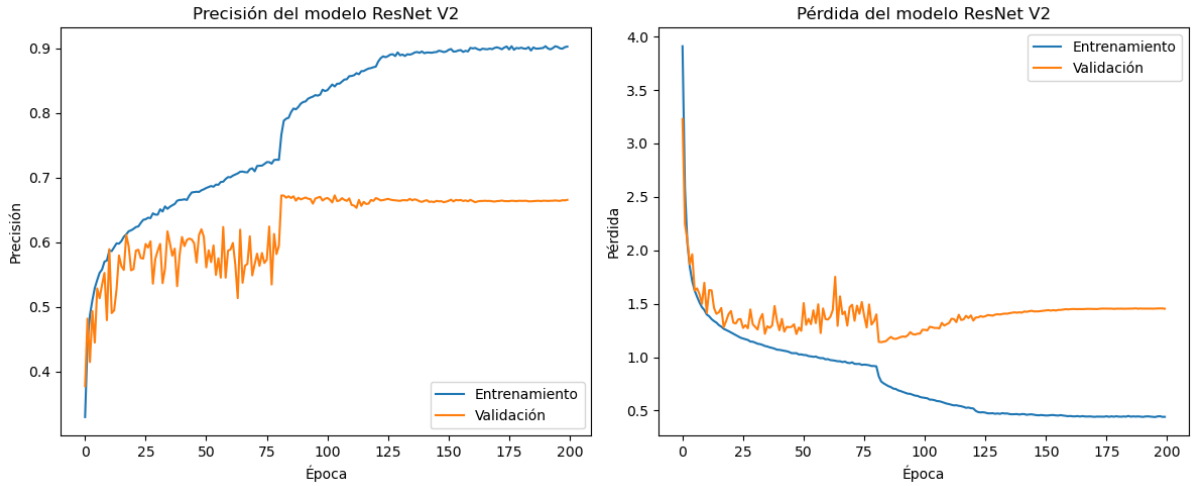

Figura 5: Curvas de exactitud y pérdida para ResNet V1

6. Conclusiones ResNet V1 y V2

En la Figura 1 se observa que la precisión en entrenamiento aumenta progresivamente y alcanza un valor cercano a 0.93 a partir de la época 100, manteniéndose luego estable. La precisión en validación, en cambio, se estabiliza en 0.68 aproximadamente, sin mostrar mejoras relevantes despues de la epoca 100. En cuanto a la pérdida, la curva de entrenamiento disminuye de forma constante hasta un valor cercano a 0.35. Sin embargo, la pérdida de validación presenta oscilaciones considerables desde el inicio y se estabiliza alrededor de 1.9 a partir de la época 120. Esto indica que el modelo ajusta bien los datos de entrenamiento, pero presenta dificultades para funcionar correctamente con los datos de validación..

En la Figura 2, correspondiente a ResNet v2, el comportamiento de entrenamiento es similar al modelo anterior, alcanzando una precisión de aproximadamente 0.92 después de la época 100. En validación, la precisión se estabiliza en torno a 0.65. La curva de pérdida en entrenamiento decrece hasta llegar cercanos a 0.35. Por otro lado, la pérdida de validación desciende inicialmente, pero a partir de la época 80 muestra un comportamiento errático con una tendencia a estabilizarse en valores alrededor de 1.5. Esta diferencia sugiere una separación creciente entre el rendimiento en entrenamiento y validación.

En resumen, aunque ResNet v2 tiene más capas y un diseño distinto en las conexiones internas, su rendimiento en validación es similar al de ResNet v1. Esto sugiere que aumentar la profundidad y modificar la estructura no garantiza una mejora en la capacidad del modelo para adaptarse a datos nuevos. En ambos casos, el entrenamiento progresa de manera efectiva, pero los resultados en validación se estabilizan sin mostrar una mejora significativa después de cierta cantidad de épocas.