Práctica 1: Experimentación con Adaline y Perceptrón Multicapa

Redes de Neuronas Artificiales

Noviembre 2023

Grado en Ingeniería Informática

Pablo Hidalgo Delgado. NIA: 100451225
Marcos Caballero Cortés. NIA: 100451047

ÍNDICE

1. Introducción

En esta práctica, nos enfrentamos al desafío de abordar un problema de regresión utilizando dos modelos de redes de neuronas: el modelo lineal Adaline y el modelo no lineal Perceptrón Multicapa. Para lograrlo vamos a utilizar el conjunto de datos proporcionado, el cual consta de medidas recopiladas a lo largo de 4 años de una turbina de gas.

El objetivo de la práctica es predecir el rendimiento energético de la turbina (TEY), para ello dividiremos la práctica en dos partes. En la primera, nos enfocaremos en el modelo Adaline, desarrollando el programa desde cero para realizar el aprendizaje y la evaluación de esta red. En la segunda parte nos centraremos en el Perceptrón Multicapa.

2. Análisis exploratorio de datos (EDA)

Primero, realizamos un breve Análisis Exploratorio de Datos (EDA) para resumir las características clave de nuestro conjunto de datos. Esto nos permite comprender mejor los datos y optimizar la construcción de nuestro modelo. A través de este análisis, identificamos que el conjunto de datos consta de 36,733 instancias y 11 atributos de tipo float64, sin valores nulos ni atributos constantes.

3. Preproceso

Previamente a entrenar los modelos, debemos realizar una transformación de los datos para que puedan ser interpretados por los algoritmos de manera eficiente. Las transformaciones de datos que realizamos son las siguientes:

3.1 División en datos de train, validation y test

Dividimos el conjunto de datos en train (70%), validation (15%) y test (15%) para el proceso de entrenamiento, validación y evaluación de nuestros modelos. Esto lo realizamos mediante la función train_test_split() de la librería de scikit-learn.

3.2 Aleatorización

En nuestro caso, puesto que no buscamos realizar un análisis de series temporales, es importante aleatorizar los datos ya que vienen ordenados cronológicamente. Este hecho puede dar lugar a patrones en el tiempo que los modelos son capaces de captar. Esto introduciría un sesgo en los modelos y perderíamos capacidad de generalización.
La aleatorización la realizamos a la vez que la división de datos especificando el parámetro shuffle = True en la función train_test_split().

3.3 Normalización

Algunos algoritmos de aprendizaje automático son sensibles a las diferencias en escala y pueden dar resultados menos precisos por esta razón. En consecuencia, para evitar estas diferencias de escala, normalizamos nuestros datos haciendo uso de la fórmula proporcionada en el enunciado de la práctica

Cabe destacar que los valores máximos y mínimos utilizados en la normalización, se escogen exclusivamente a partir de los datos de entrenamiento. El propósito de esta práctica es evitar cualquier tipo de information leakage que se produzca al dar información al modelo sobre datos no utilizados para entrenar.

4. Adaline

4.1 Experimentación

Hemos realizado una serie de experimentos utilizando el modelo Adaline desarrollado, en los cuales hemos variado los valores de sus hiperparámetros (tasa de aprendizaje y número de épocas). Los valores que hemos seleccionado son los siguientes: tasas de aprendizaje de [0.4, 0.2, 0.1, 0.01, 0.005] y números de épocas de [10, 50, 100]. Como resultado, para cada valor de la tasa de aprendizaje, hemos construido 3 modelos distintos con diferentes números de épocas. Cada modelo ha sido evaluado con el MSE de entrenamiento, validación y test.
Esta variedad de experimentos nos permite examinar en detalle las diferencias en el rendimiento de cada modelo en función de los valores introducidos de sus hiperparámetros.

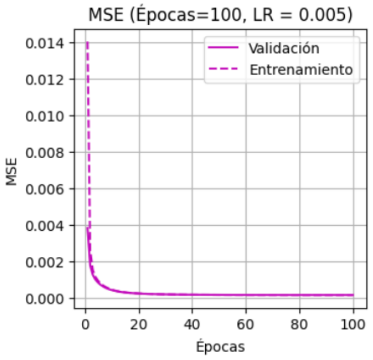
4.2 Resultados obtenidos

Hemos seleccionado el valor mínimo de error de validación registrado durante todo el proceso de entrenamiento en las siguientes tablas de resultados. También proporcionamos la época en la que se alcanzó este mínimo en la columna "Best Epoch" en cada uno de los experimentos.

Learning Rate	Epochs	Best Epoch	MSE Train	MSE Validation	MSE Test
0.4	30	25	1.26e-03	2.78e-04	2.81e-04
0.4	50	48	1.26e-03	2.78e-04	2.81e-04
0.4	100	48	1.26e-03	2.78e-04	2.81e-04
0.2	30	3	2.63e-04	2.64e-04	2.65e-04
0.2	50	3	2.63e-04	2.64e-04	2.65e-04
0.2	100	3	2.63e-04	2.64e-04	2.65e-04
0.1	30	4	1.89e-04	2.06e-04	2.09e-04
0.1	50	4	1.89e-04	2.06e-04	2.09e-04
0.1	100	4	1.89e-04	2.06e-04	2.09e-04
0.01	30	30	1.56e-04	1.60e-04	1.61e-04
0.01	50	50	1.51e-04	1.56e-04	1.55e-04
0.01	100	55	1.50e-04	1.56e-04	1.55e-04
0.005	30	30	1.89e-04	1.89e-04	1.91e-04
0.005	50	50	1.60e-04	1.62e-04	1.63e-04
0.005	100	100	1.49e-04	1.53e-04	1.52e-04

Tabla 1: Resultados obtenidos para cada modelo Adaline creado.

Escogemos el mejor modelo con el menor error de validación (lr = 0.005, epochs = 100). La evolución de su error de train y validation es la siguiente:



Gráfica 1: Evolución del error de entrenamiento y validación del mejor modelo Adaline

4.3 Análisis de resultados

Al examinar los resultados de la tabla en la sección anterior, podemos observar que las tasas de aprendizaje más bajas tienden a producir errores más pequeños. Esto puede deberse en gran medida a la extensa cantidad de épocas que utilizamos en nuestros experimentos. Al ir aprendiendo más poco a poco, el proceso de convergencia puede resultar más efectivo y preciso, lo que desencadena un menor sobreajuste a los datos de entrenamiento.

Además, al observar las épocas en las que se alcanzó el error de validación más bajo para cada modelo, vemos que para las tasas de aprendizaje de 0.1 y 0.2, el modelo alcanza su mínimo en muy pocas épocas (3 y 4 respectivamente) y no aumenta nuevamente. Sin embargo, con una tasa de aprendizaje de 0.4, las épocas en las que se obtiene el mínimo cambian en función del número de épocas. Esto sugiere la presencia de mínimos locales en el error de validación, siendo uno en la época 25 y otro en la época 48, aunque sus valores son iguales, posiblemente debido a la notación científica que redondea los valores.

Es relevante mencionar que se realizaron pruebas con tasas de aprendizaje más altas, pero los resultados nos mostraron valores de error extremadamente altos que causaron desbordamiento y errores en Python. Esto indica que estamos ante un problema de explosión de gradientes, que ocurre cuando los gradientes de los pesos en la red neuronal se vuelven excesivamente grandes. A pesar de varios intentos para abordar este problema, como cambiar las semillas de aleatoriedad o ajustar la normalización de los datos, no pudimos resolverlo de manera efectiva. Otras posibles soluciones son explorar métodos de inicialización de pesos más apropiados o, simplemente, como hemos observado, optar por tasas de aprendizaje más bajas.

5. Perceptrón multicapa

5.1 Experimentación

Para el Perceptrón Multicapa, además de probar con diferentes tasas de aprendizaje y números de épocas, cambiamos el número de neuronas ocultas. Así, hemos escogido los siguientes valores para experimentar: learning rate = [0.5, 0.4, 0.2, 0.1, 0.01], epochs = [30, 50, 100] y número de neuronas = [10, 20, 30]. En total, se han creado 45 modelos para cada función de activación.

5.2 Resultados obtenidos

5.2.1 RELU

Learning Rate	Epochs	Best Epoch	Neuronas	MSE Train	MSE Validation	MSE Test
0.5	30	30	10	1.92e-04	1.86e-04	1.84e-04
0.5	30	30	20	1.90e-04	1.83e-04	1.81e-04
0.5	30	30	30	1.79e-04	1.74e-04	1.71e-04
0.5	50	50	10	1.80e-04	1.76e-04	1.75e-04
0.5	50	50	20	1.85e-04	1.79e-04	1.77e-04
0.5	50	50	30	1.69e-04	1.64e-04	1.63e-04
0.5	100	100	10	3.44e-04	3.31e-04	3.33e-04
0.5	100	100	20	1.84e-04	1.79e-04	1.78e-04
0.5	100	100	30	1.47e-04	1.44e-04	1.41e-04
0.4	30	30	10	2.08e-04	2.00e-04	2.02e-04
0.4	30	30	20	2.01e-04	1.96e-04	1.94e-04
0.4	30	30	30	1.92e-04	1.85e-04	1.86e-04
0.4	50	50	10	1.87e-04	1.82e-04	1.82e-04
0.4	50	50	20	1.85e-04	1.80e-04	1.77e-04
0.4	50	50	30	1.82e-04	1.77e-04	1.77e-04
0.4	100	100	10	1.72e-04	1.69e-04	1.69e-04
0.4	100	100	20	1.78e-04	1.75e-04	1.71e-04
0.4	100	100	30	1.65e-04	1.62e-04	1.60e-04
0.2	30	30	10	2.62e-04	2.52e-04	2.55e-04
0.2	30	30	20	2.09e-04	2.03e-04	2.04e-04
0.2	30	30	30	2.12e-04	2.04e-04	2.07e-04
0.2	50	50	10	1.92e-04	1.87e-04	1.86e-04
0.2	50	50	20	1.95e-04	1.91e-04	1.91e-04
0.2	50	50	30	1.90e-04	1.85e-04	1.85e-04
0.2	100	100	10	1.89e-04	1.86e-04	1.87e-04
0.2	100	100	20	1.61e-04	1.59e-04	1.59e-04
0.2	100	100	30	1.83e-04	1.79e-04	1.79e-04
0.1	30	30	10	3.70e-04	3.60e-04	3.68e-04
0.1	30	30	20	2.46e-04	2.36e-04	2.44e-04
0.1	30	30	30	2.25e-04	2.19e-04	2.21e-04
0.1	50	50	10	3.52e-04	3.51e-04	3.52e-04
0.1	50	50	20	3.50e-04	3.48e-04	3.50e-04
0.1	50	50	30	1.98e-04	1.96e-04	1.93e-04
0.1	100	100	10	2.43e-04	2.38e-04	2.41e-04
0.1	100	100	20	1.85e-04	1.83e-04	1.83e-04
0.1	100	100	30	1.91e-04	1.89e-04	1.87e-04
0.01	30	30	10	7.74e-04	7.49e-04	7.60e-04
0.01	30	30	20	7.24e-04	7.03e-04	7.03e-04
0.01	30	30	30	7.08e-04	6.90e-04	6.76e-04
0.01	50	50	10	8.38e-04	8.18e-04	8.13e-04
0.01	50	50	20	5.27e-04	5.11e-04	5.17e-04
0.01	50	50	30	4.35e-04	4.26e-04	4.23e-04
0.01	100	100	10	4.82e-04	4.69e-04	4.72e-04
0.01	100	100	20	4.79e-04	4.65e-04	4.71e-04
0.01	100	100	30	3.56e-04	3.36e-04	3.49e-04

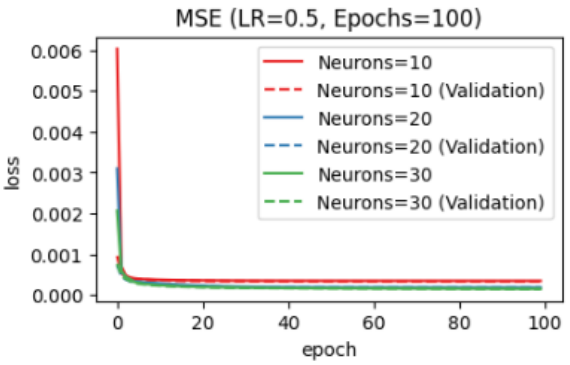
Tabla 2: Resultados obtenidos para cada modelo Perceptrón Multicapa RELU creado.

5.2.2 Sigmoide

Learning Rate	Epochs	Best Epoch	Neuronas	MSE Train	MSE Validation	MSE Test
0.5	30	30	10	2.91e-04	2.83e-04	2.86e-04
0.5	30	30	20	3.06e-04	2.99e-04	3.01e-04
0.5	30	30	30	3.19e-04	3.10e-04	3.14e-04
0.5	50	50	10	2.72e-04	2.63e-04	2.69e-04
0.5	50	50	20	2.94e-04	2.86e-04	2.89e-04
0.5	50	50	30	2.74e-04	2.65e-04	2.70e-04
0.5	100	100	10	2.33e-04	2.27e-04	2.32e-04
0.5	100	100	20	2.42e-04	2.34e-04	2.39e-04
0.5	100	100	30	2.51e-04	2.43e-04	2.47e-04
0.4	30	30	10	3.15e-04	3.07e-04	3.11e-04
0.4	30	30	20	3.13e-04	3.05e-04	3.09e-04
0.4	30	30	30	3.39e-04	3.32e-04	3.35e-04
0.4	50	50	10	2.96e-04	2.86e-04	2.93e-04
0.4	50	50	20	2.97e-04	2.87e-04	2.93e-04
0.4	50	50	30	3.12e-04	3.05e-04	3.08e-04
0.4	100	100	10	2.39e-04	2.33e-04	2.36e-04
0.4	100	100	20	2.39e-04	2.33e-04	2.37e-04
0.4	100	100	30	2.54e-04	2.46e-04	2.51e-04
0.2	30	30	10	3.48e-04	3.43e-04	3.44e-04
0.2	30	30	20	3.66e-04	3.60e-04	3.63e-04
0.2	30	30	30	3.67e-04	3.61e-04	3.62e-04
0.2	50	50	10	3.11e-04	3.04e-04	3.08e-04
0.2	50	50	20	3.30e-04	3.23e-04	3.29e-04
0.2	50	50	30	3.09e-04	3.02e-04	3.07e-04
0.2	100	100	10	2.78e-04	2.72e-04	2.76e-04
0.2	100	100	20	2.77e-04	2.71e-04	2.75e-04
0.2	100	100	30	3.09e-04	3.01e-04	3.07e-04
0.1	30	30	10	4.08e-04	4.02e-04	4.01e-04
0.1	30	30	20	4.26e-04	4.18e-04	4.16e-04
0.1	30	30	30	4.38e-04	4.34e-04	4.33e-04
0.1	50	50	10	4.05e-04	4.02e-04	4.00e-04
0.1	50	50	20	3.64e-04	3.59e-04	3.61e-04
0.1	50	50	30	3.75e-04	3.71e-04	3.74e-04
0.1	100	100	10	3.35e-04	3.30e-04	3.32e-04
0.1	100	100	20	3.14e-04	3.09e-04	3.13e-04
0.1	100	100	30	3.51e-04	3.45e-04	3.50e-04
0.01	30	30	10	7.24e-03	6.92e-03	6.79e-03
0.01	30	30	20	4.90e-03	4.70e-03	4.63e-03
0.01	30	30	30	4.19e-03	4.03e-03	3.99e-03
0.01	50	50	10	1.56e-03	1.52e-03	1.51e-03
0.01	50	50	20	2.38e-03	2.31e-03	2.32e-03
0.01	50	50	30	1.42e-03	1.38e-03	1.39e-03
0.01	100	100	10	1.01e-03	9.97e-04	9.88e-04
0.01	100	100	20	9.08e-04	9.00e-04	8.92e-04
0.01	100	100	30	8.42e-04	8.31e-04	8.29e-04

Tabla 3: Resultados obtenidos para cada modelo Perceptrón Multicapa sigmoide creado.

Escogemos el mejor modelo con el menor error de validación (lr = 0.5, neuronas = 30, epochs = 100) y función de activación RELU. La evolución de su error de train y validation es la siguiente:



Gráfica 2: Evolución del error de entrenamiento y validación del mejor modelo PM RELU

5.3 Análisis de resultados

Analizando los resultados obtenidos, en líneas generales, hemos observado que los valores de error al utilizar la función sigmoide tienden a ser mayores en comparación con la función RELU. Una posible explicación de este fenómeno se detalla en la siguiente sección.

En lo que respecta al número de neuronas ocultas, hemos constatado que no necesariamente un aumento en la cantidad de neuronas conlleva una disminución en los errores. Esto se hace especialmente evidente en redes con tasas de aprendizaje más altas, donde al final, debido al extenso número de épocas que evaluamos, las redes tienden a converger en valores de error similares. Sin embargo, es relevante mencionar que en las primeras etapas de entrenamiento, los modelos con más neuronas ocultas obtienen niveles de error más bajos, como se aprecia en la gráfica 2.

También hemos observado que en todos los experimentos, la época en la que se registra el error de validación más bajo es la última. Esto nos lleva a considerar la posibilidad de que aumentar el número de épocas de entrenamiento podría haber continuado reduciendo el error de validación, aunque la variación adicional sería mínima.

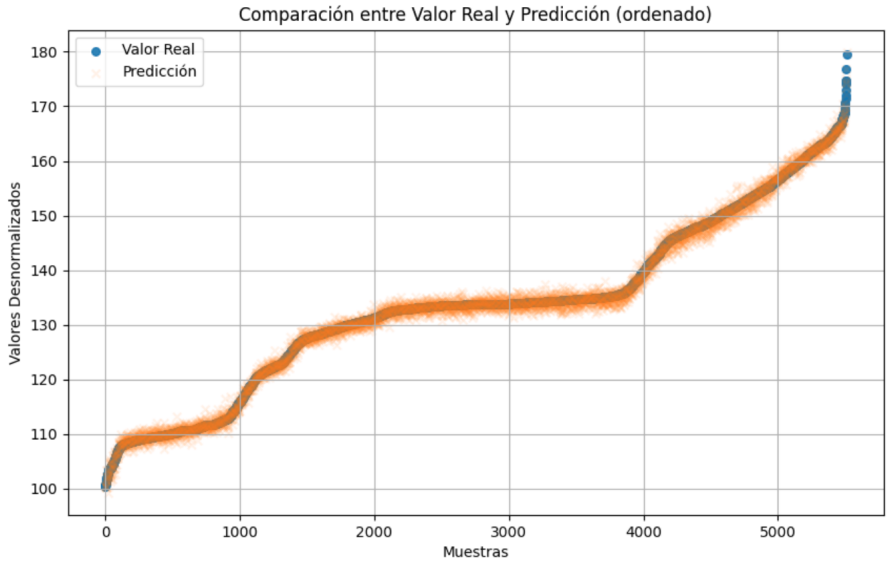
En consonancia con lo observado en Adaline, en términos generales, en cada experimento, los tres tipos de error (test, validación y entrenamiento) arrojan valores similares, lo que indica que el modelo presenta una buena capacidad de generalización hacia los datos.

6. Comparación Adaline y perceptrón multicapa

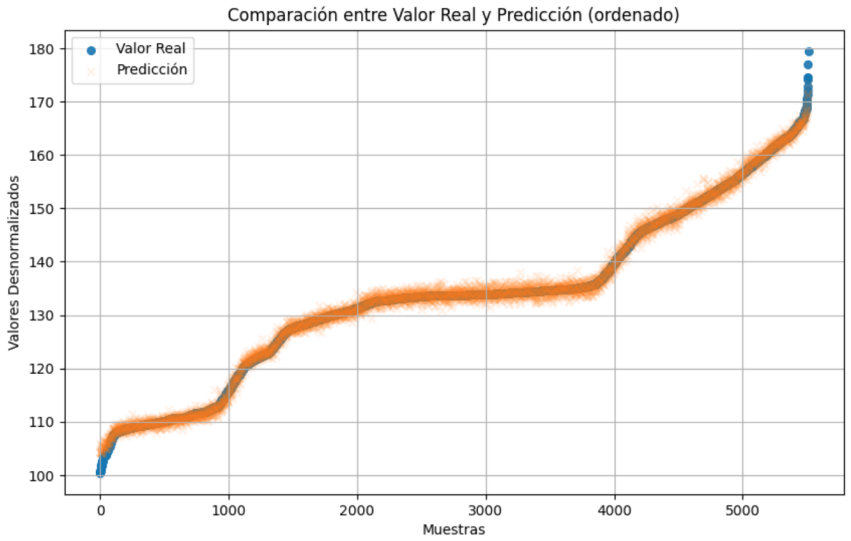
Al comparar los resultados de los dos mejores modelos seleccionados, Adaline (lr = 0.005, epochs = 100) y PM RELU (lr = 0.5, número de neuronas = 30, epochs = 100), observamos que el valor del error de test es bastante similar ($1.520 \cdot 10^{-4}$ y $1.446 \cdot 10^{-4}$ respectivamente). Este hecho podría ser un indicador que nuestros datos mantienen relaciones lineales, ya que el Adaline es eficaz para captar estas relaciones y da buenos resultados, mientras que el PM, además de relaciones lineales, también es capaz de detectar y predecir relaciones no lineales.

La hipótesis de la linealidad de nuestros datos se refuerza aún más cuando comparamos los resultados del PM con función RELU con el PM con función sigmoide. Y es que la

función de activación sigmoide, por su naturaleza, no funciona bien con relaciones lineales. Sin embargo, la función RELU sí funciona bien con relaciones lineales, ya que, si la entrada a la función es positiva, la salida es lineal, similar al comportamiento del Adaline. Esto concuerda con los resultados obtenidos, puesto que los errores con función sigmoide son mayores que los obtenidos con la función RELU.
La linealidad de los datos también se puede analizar observando las siguientes gráficas de comparación entre los valores reales y las predicciones:



Gráfica 3: Comparación entre valor real y predicción mejor modelo Adaline



Gráfica 4: Comparación entre valor real y predicción mejor modelo PM RELU

En estas gráficas, observamos cómo los puntos siguen una tendencia que se asemeja a una línea ascendente, aunque con algunas curvas o desviaciones. En consecuencia, podemos afirmar que nuestros datos tienen una relación lineal general, pero también presentan ciertas no linealidades o características adicionales. Estas características

adicionales no son perceptibles para Adaline, pero sí para el PM multicapa. Esto explica por qué el error de test de este último es ligeramente menor.
También observando las gráficas vemos la gran similitud de las predicciones hechas por ambos modelos y la buena capacidad de generalización al adaptarse realmente bien a los valores reales de los datos.

7. Conclusiones

En resumen, nuestros experimentos nos han permitido sacar diversas ideas sobre nuestros modelos y los datos utilizados. Primero, al analizar el modelo Adaline, observamos que las tasas de aprendizaje más bajas tienden a producir errores más pequeños, lo que sugiere que una convergencia gradual, respaldada por un mayor número de épocas, puede ser crucial para obtener resultados óptimos. No obstante, en esta situación deberíamos tener en cuenta el factor del tiempo, ya que aumentaría notablemente. La elección adecuada de la tasa de aprendizaje es esencial para evitar problemas de convergencia y explosión de gradientes, que se manifestaron al explorar tasas más altas.

En segundo lugar, en el caso del perceptrón multicapa (PM) con funciones de activación sigmoide y RELU, observamos diferencias significativas en el rendimiento. La función sigmoide tendió a producir valores de error más altos en comparación con la función RELU, lo que respalda la hipótesis de la linealidad en nuestros datos. Además, el número de neuronas ocultas no resultó ser un factor crítico, ya que un mayor número de neuronas no necesariamente redujo el error, especialmente en combinación con tasas de aprendizaje más altas puesto que se alcanza la convergencia rápidamente. Es probable que si hubiésemos elegido un mayor número de neuronas ocultas se hubiese evidenciado la diferencia en el rendimiento pero decidimos no hacerlo debido a que el tiempo de entrenamiento aumentaría.

Finalmente, al comparar los mejores modelos Adaline y PM RELU, encontramos que ambos modelos produjeron valores de error de test muy similares. Esto sugiere que nuestros datos mantienen relaciones lineales, las cuales Adaline es capaz de capturar eficazmente. Sin embargo, el PM también puede detectar relaciones no lineales, lo que respalda su ligeramente mejor rendimiento en términos de error. Ambos modelos demostraron una destacada capacidad de generalización, lo que se reflejó en la similitud de las predicciones con los valores reales en las gráficas.

En general, nuestros experimentos proporcionaron conocimientos valiosos para el ajuste de hiperparámetros y la comprensión de la relación entre las características del modelo y el rendimiento en la resolución de problemas de aprendizaje automático. Para futuros trabajos, consideramos la implementación de la validación cruzada como una estrategia para mejorar aún más nuestros resultados.

En un sentido más personal, hemos dedicado mucho tiempo y puesto un gran esfuerzo en esta práctica. En particular, hemos enfrentado dificultades considerables en la comprensión y resolución del problema de explosión de gradientes, ya que, al principio, no lográbamos entender la causa subyacente de los resultados que estábamos obteniendo.