Deep Learning y Big Data con Python

Laboratorio 2

Estudiante:

Jose Javier Arce Solorzano

Profesor:

Adan de Jesus Mora Fallas

1. Preprocesamiento de los Datos

- Carga del dataset: Se utilizó el conjunto de datos California Housing de sklearn.datasets, el cual contiene variables socioeconómicas para predecir el valor medio de una vivienda.
- **Estandarización:** Se aplicó StandardScaler para normalizar las variables independientes a media cero y desviación estándar uno. Esto es fundamental para mejorar la estabilidad y desempeño de redes neuronales.
- **Conversión a tensores:** Los datos fueron convertidos a tensores de PyTorch, requisito para su procesamiento. La variable objetivo y se reformateó a una forma de vector columna.
- **División de datos:** Se dividieron los datos en conjunto de entrenamiento (80 %) y prueba (20 %) usando train_test_split, con el fin de evaluar el modelo sobre datos no vistos.

2. Entrenamiento y Diseño Experimental

- **Tasa de aprendizaje (*learning rate*):** Se utilizó un valor de 0.01. Esta tasa es un punto de partida común que permite al modelo aprender de manera eficiente sin que los pasos de optimización sean demasiado grandes, lo cual podría dificultar la convergencia.
- **Número de épocas:** Se entrenó cada modelo durante 100 épocas. Esta cantidad permite observar una curva de aprendizaje suficientemente larga para evaluar si el modelo converge, sobreentrena o subentrena, sin extender innecesariamente el tiempo de cómputo.
- **Tamaño del lote (*batch size*):** Se empleó un tamaño de lote de 64, que es un compromiso entre estabilidad del gradiente y velocidad de entrenamiento. Lot sizes demasiado pequeños pueden producir curvas de pérdida muy ruidosas, y tamaños muy grandes requieren mayor capacidad de memoria.
- **Optimización:** Se probaron dos algoritmos de optimización:
  - **SGD (Stochastic Gradient Descent):** Método clásico, sensible a la tasa de aprendizaje, útil para comparar con métodos modernos.
  - **Adam:** Un optimizador adaptativo más robusto que ajusta dinámicamente la tasa de aprendizaje, ofreciendo generalmente una convergencia más rápida y estable.
- **Funciones de pérdida:**
  - `MSELoss`: Pérdida cuadrática media, estándar en problemas de regresión.
  - `SmoothL1Loss`: Alternativa robusta a MSE que penaliza menos los errores grandes, útil si hay valores atípicos.

- **Variación de arquitecturas:** Se evaluaron diferentes configuraciones de capas ocultas:
  - `[64]`: Una sola capa con 64 neuronas.
  - `[16, 8]`: Arquitectura más profunda con dos capas sucesivas de 16 y 8 neuronas.

  Esto permite comparar modelos simples vs. más complejos en términos de capacidad de representación y generalización.
- **Evaluación:** Se calcularon las métricas `RMSE` (raíz del error cuadrático medio) y $\texttt{R}^2$ (coeficiente de determinación) sobre el conjunto de prueba. Además, se generaron gráficas de:
  - Evolución de la pérdida durante el entrenamiento.
  - Dispersión entre predicciones y valores reales.

A manera de resumen, se presenta la siguiente tabla resumiendo los ocho diferentes experimentos realizados y los parámetros usados en cada uno.

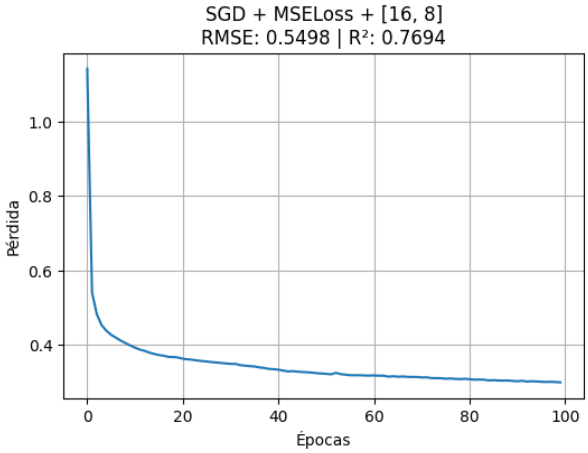
Experimento	Arquitectura	Pérdida	Optimizador	LR	Epochs	Batch Size
1	[16, 8]	MSELoss	SGD	0.01	100	64
2	[16, 8]	MSELoss	Adam	0.01	100	64
3	[16, 8]	SmoothL1Loss	SGD	0.01	100	64
4	[16, 8]	SmoothL1Loss	Adam	0.01	100	64
5	[64]	MSELoss	SGD	0.01	100	64
6	[64]	MSELoss	Adam	0.01	100	64
7	[64]	SmoothL1Loss	SGD	0.01	100	64
8	[64]	SmoothL1Loss	Adam	0.01	100	64

Cuadro 1: Configuraciones empleadas en cada experimento.

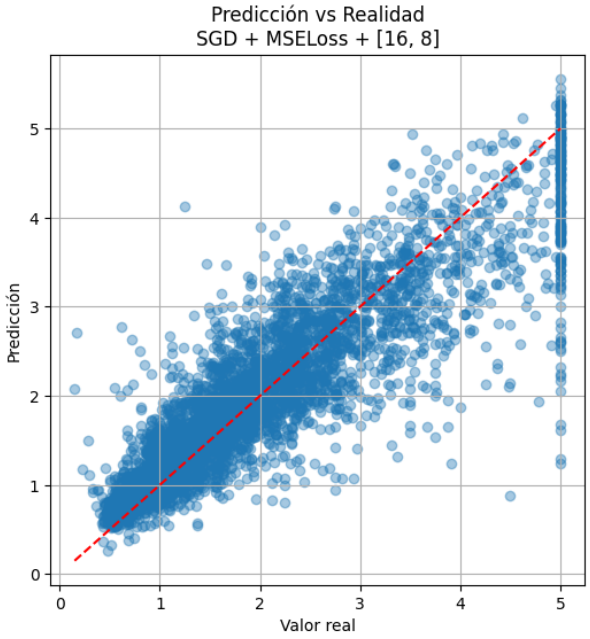
3. Resultados Obtenidos

A continuación se muestran los gráficos de evolución de la pérdida en función de la época y de predicción vs realidad para los 8 distintos experimentos.

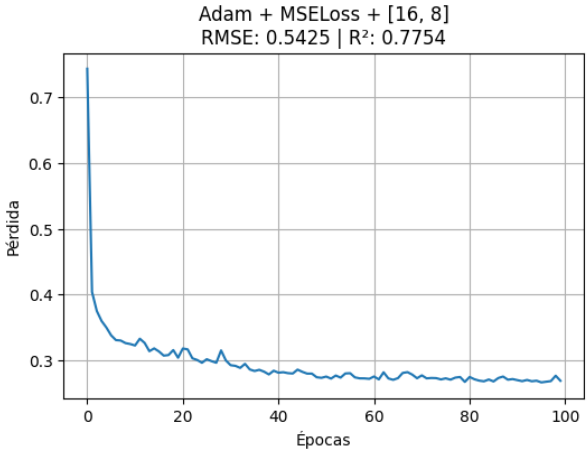
Experimento 1 y 2



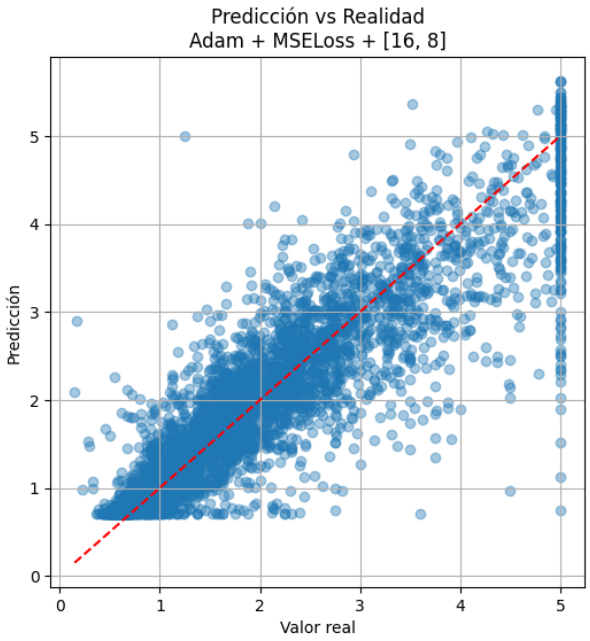
(a) Exp. 1: Evolución de la pérdida



(b) Exp. 1: Predicción vs realidad



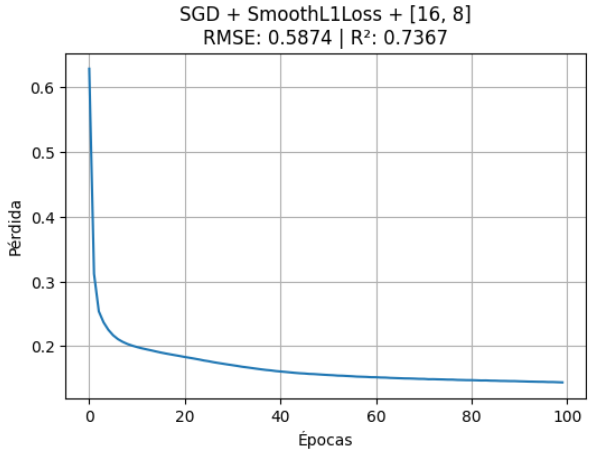
(c) Exp. 2: Evolución de la pérdida



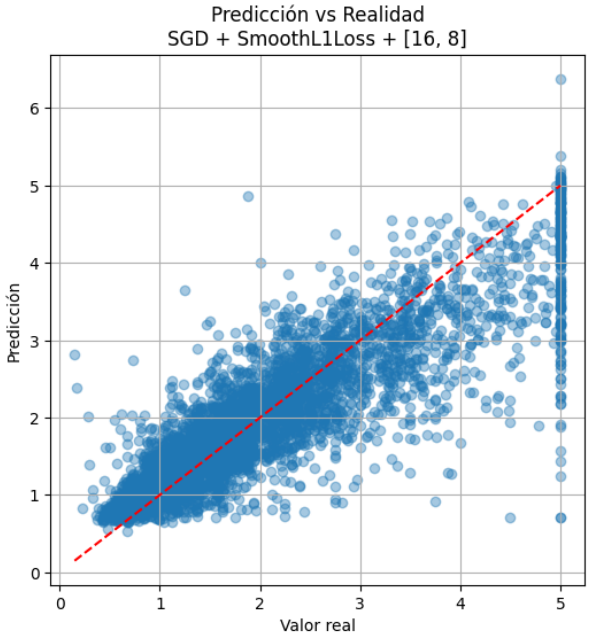
(d) Exp. 2: Predicción vs realidad

Figura 1: Resultados de los Experimentos 1 y 2.

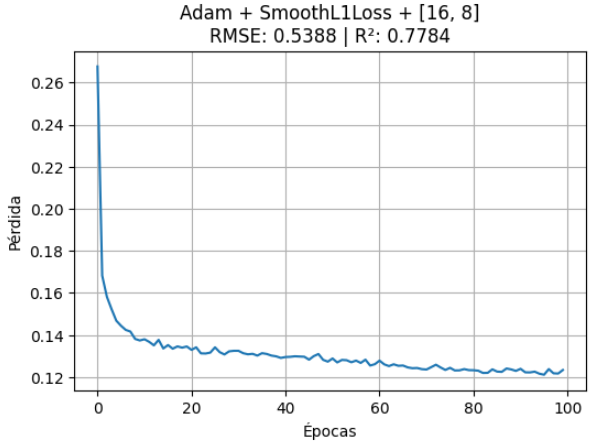
Experimento 3 y 4



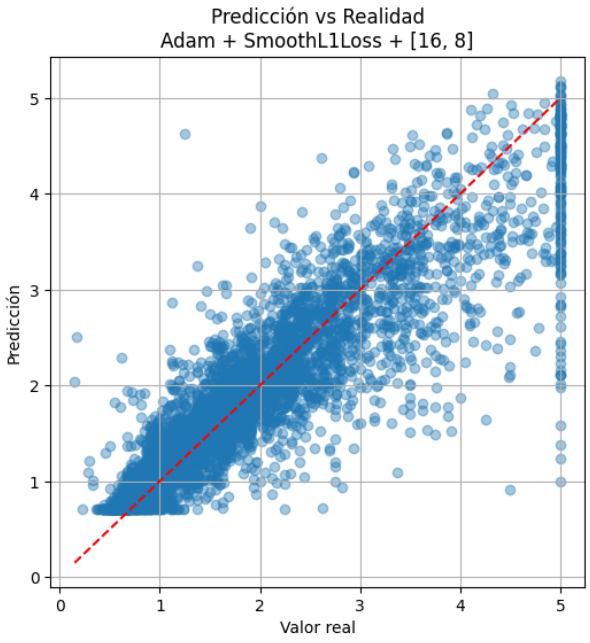
(a) Exp. 3: Evolución de la pérdida



(b) Exp. 3: Predicción vs realidad



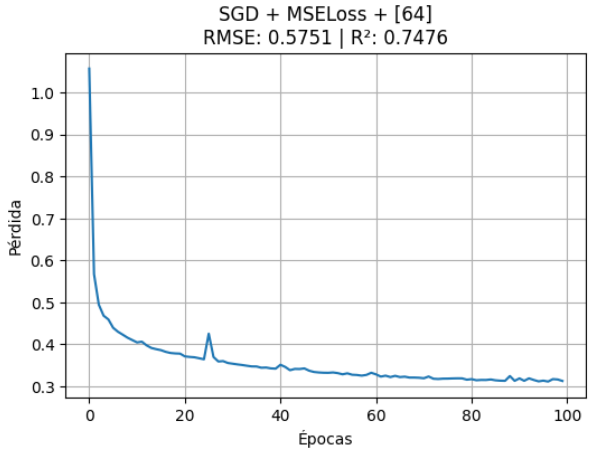
(c) Exp. 4: Evolución de la pérdida



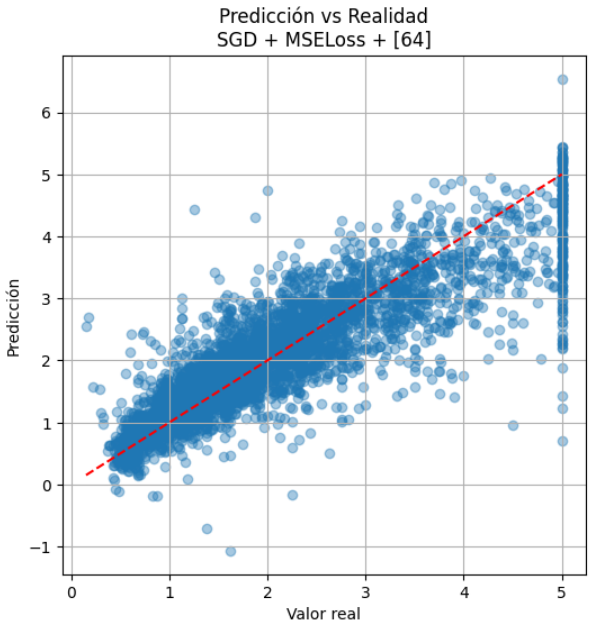
(d) Exp. 4: Predicción vs realidad

Figura 2: Resultados de los Experimentos 3 y 4.

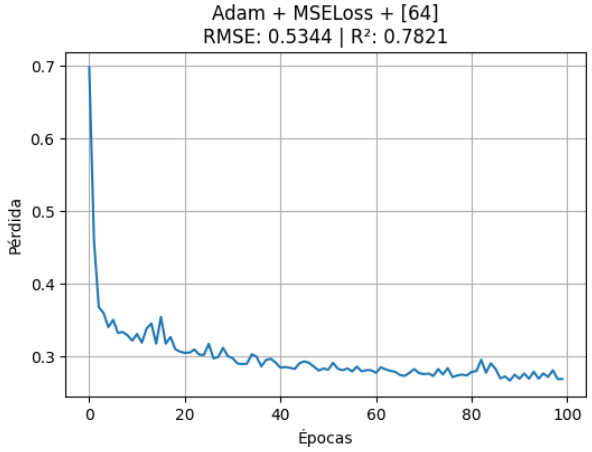
Experimento 5 y 6



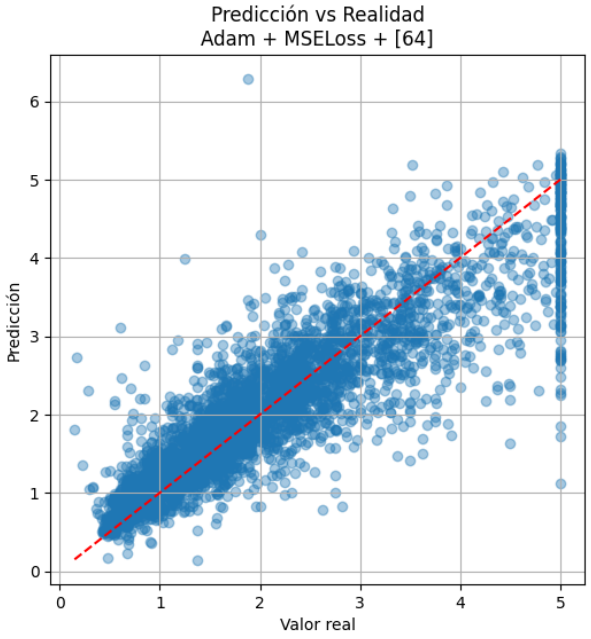
(a) Exp. 5: Evolución de la pérdida



(b) Exp. 5: Predicción vs realidad



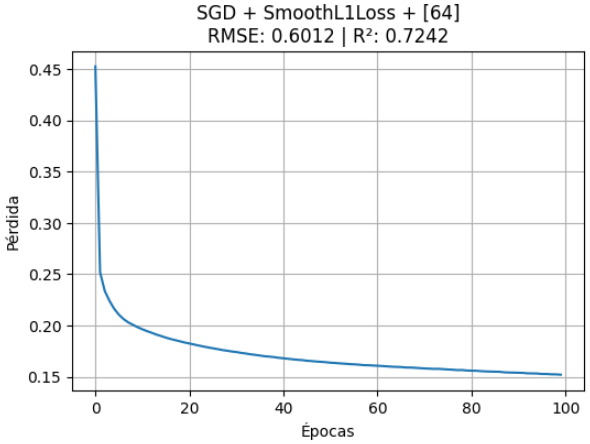
(c) Exp. 6: Evolución de la pérdida



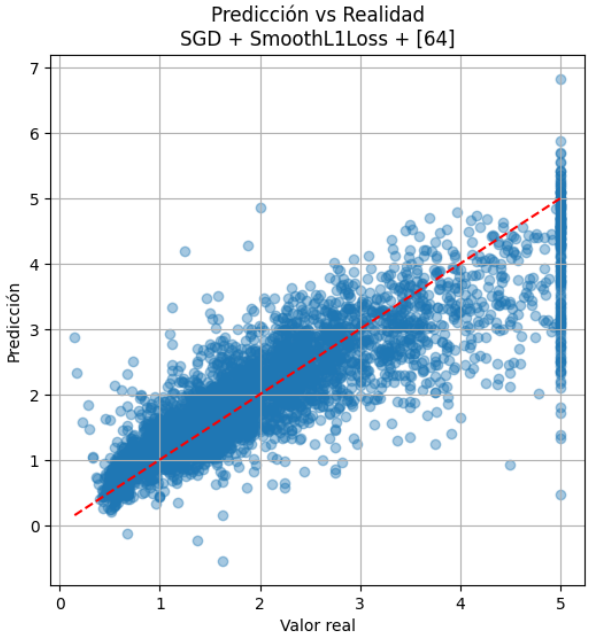
(d) Exp. 6: Predicción vs realidad

Figura 3: Resultados de los Experimentos 5 y 6.

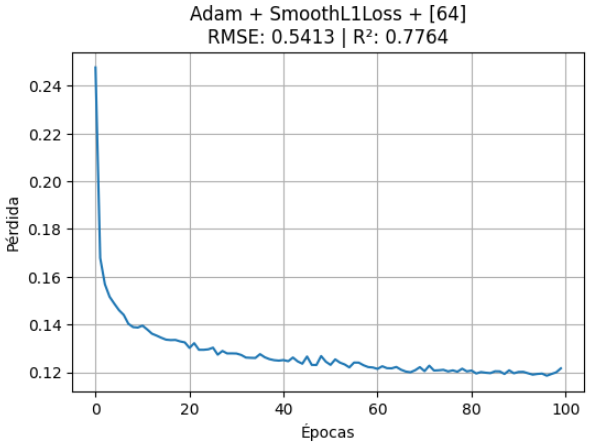
Experimento 7 y 8



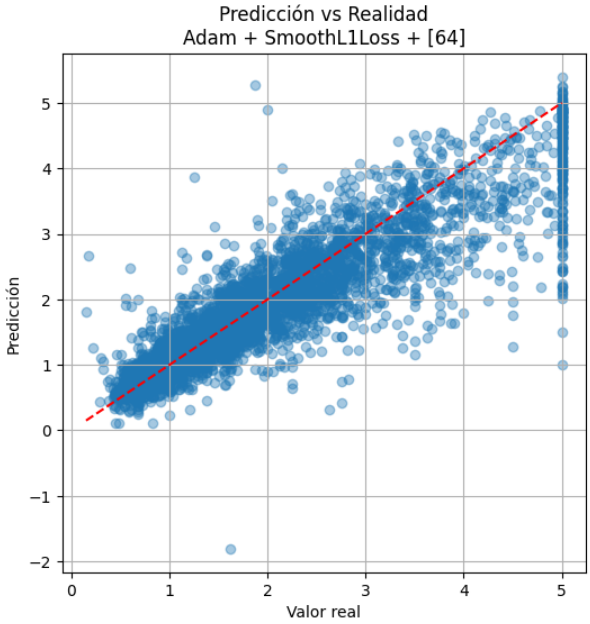
(a) Exp. 7: Evolución de la pérdida



(b) Exp. 7: Predicción vs realidad



(c) Exp. 8: Evolución de la pérdida



(d) Exp. 8: Predicción vs realidad

Figura 4: Resultados de los Experimentos 7 y 8.

Comparación de Resultados Obtenidos

A continuación se presenta una tabla resumen con los valores de RMSE y R^2

Experimento	Optimizador	Función de pérdida	Arquitectura	RMSE	R^2
1	SGD	MSELoss	[16, 8]	0.5498	0.7694
2	Adam	MSELoss	[16, 8]	0.5425	0.7754
3	SGD	SmoothL1Loss	[16, 8]	0.5874	0.7367
4	Adam	SmoothL1Loss	[16, 8]	0.5388	0.7784
5	SGD	MSELoss	[64]	0.5751	0.7476
6	Adam	MSELoss	[64]	0.5344	0.7821
7	SGD	SmoothL1Loss	[64]	0.6012	0.7242
8	Adam	SmoothL1Loss	[64]	0.5413	0.7764

Cuadro 2: Resumen de métricas por experimento.

4. Análisis de Resultados

Se realizaron ocho experimentos variando tres aspectos de la red neuronal para regresión sobre el *California Housing Dataset*: la configuración de las capas ocultas, la función de pérdida y el optimizador empleado. Para evaluar el desempeño de cada combinación, se utilizaron las métricas RMSE y R^2.

Influencia del optimizador En todos los casos se observa que el optimizador Adam mejora de manera consistente tanto el error cuadrático medio como el coeficiente de determinación en comparación con SGD. Esta tendencia sugiere que Adam, al adaptar las tasas de aprendizaje individualmente para cada parámetro y aprovechar momentos de primer y segundo orden, facilita una convergencia más estable y acelerada en problemas de regresión con datos heterogéneos.

Efecto de la función de pérdida Al comparar las dos funciones de pérdida, MSELoss y SmoothL1Loss, destaca que SmoothL1Loss tiende a producir valores de R^2 superiores, lo cual puede reflejar una mayor robustez frente a valores atípicos o distribuciones de error no perfectamente gaussianas. No obstante, en algunos experimentos el RMSE asociado a SmoothL1Loss resulta ligeramente mayor, indicando un equilibrio entre penalizar errores grandes y mantener la estabilidad durante el entrenamiento.

Impacto de la arquitectura En lo referente a la profundidad y tamaño de la red, la variante con una sola capa oculta de mayor dimensión suele ofrecer un mejor rendimiento cuando se combina con Adam. Esto puede explicarse porque una capa más amplia resulta más capaz de modelar relaciones no lineales complejas en los datos. En contraste, al usar SGD, la diferencia de desempeño entre la arquitectura de dos capas pequeñas y la de una capa grande se atenúa, posiblemente debido a la sensibilidad de SGD a la tasa de aprendizaje fija.

5. Conclusiones

- El optimizador Adam presenta una convergencia más estable y un desempeño general superior en comparación con SGD.
- La función de pérdida SmoothL1Loss muestra una mejor capacidad de generalización en términos de R^2, aunque puede aumentar ligeramente el RMSE.
- Una arquitectura con una sola capa oculta de mayor tamaño resulta más efectiva al combinarse con Adam, al capturar con mayor fidelidad la complejidad de los datos.
- Cuando se emplea SGD, las diferencias entre arquitecturas son menos marcadas, lo que indica que la selección del optimizador es clave para aprovechar la arquitectura de la red.