Assignment 3: TensorFlow Benchmarking of Architectures and Activations

Course: Fundamentals of Deep Learning

Assoc. Prof.: Halit Bakır

Student: Muhammad Usman – 250110009

Date: November 2, 2025

I. Introduction and Experimental Setup

1. Objective

The primary objective of this assignment was to conduct a controlled, empirical study using **TensorFlow/Keras** to evaluate how three critical hyperparameter choices—**Activation Function, Network Depth, and Network Width**—affect the training dynamics, convergence, and final accuracy (R^2) of a Feedforward Neural Network (FNN) applied to the standardized house-price regression dataset.

2. Dataset and Preprocessing

The benchmark utilized the same synthetic house-price dataset (500 samples, 3 features: Area, Rooms, Location Score) as Assignments 1 and 2. Crucially, the data was **Standardized (Z-score normalized)** and partitioned using the **exact same fixed Train/Validation/Test split** across all assignments to ensure direct comparability of results.

3. Experimental Design and Benchmark Grid

The experiment followed a systematic, controlled grid search involving **54 unique FNN models** ≥ 6 Activations$\times \geq 3$ Depths$\times \geq 3$ Widths.

Factor	Values Tested	Control Parameters (Fixed)
Activations	ReLU, LeakyReLU (α =0.01), ELU, SELU, GELU, Swish	**Optimizer:** Adam (Learning Rate = 10−3)
Depth	1, 2, 3 hidden layers	**Loss Function:** Mean Squared Error (MSE)
Width	8, 16, 32 neurons per layer	**Regularization:** Early Stopping (Patience=15 on `val_loss`)

The benchmark logged all required metrics, including Test MSE, Test parameters, for every configuration.

II. Results and Architectural Analysis

1. Activation Function Performance

The overall performance was assessed by identifying the single best architecture (Depth and Width) for each activation function and ranking them by their achieved Test R^2 score (Figure 1).

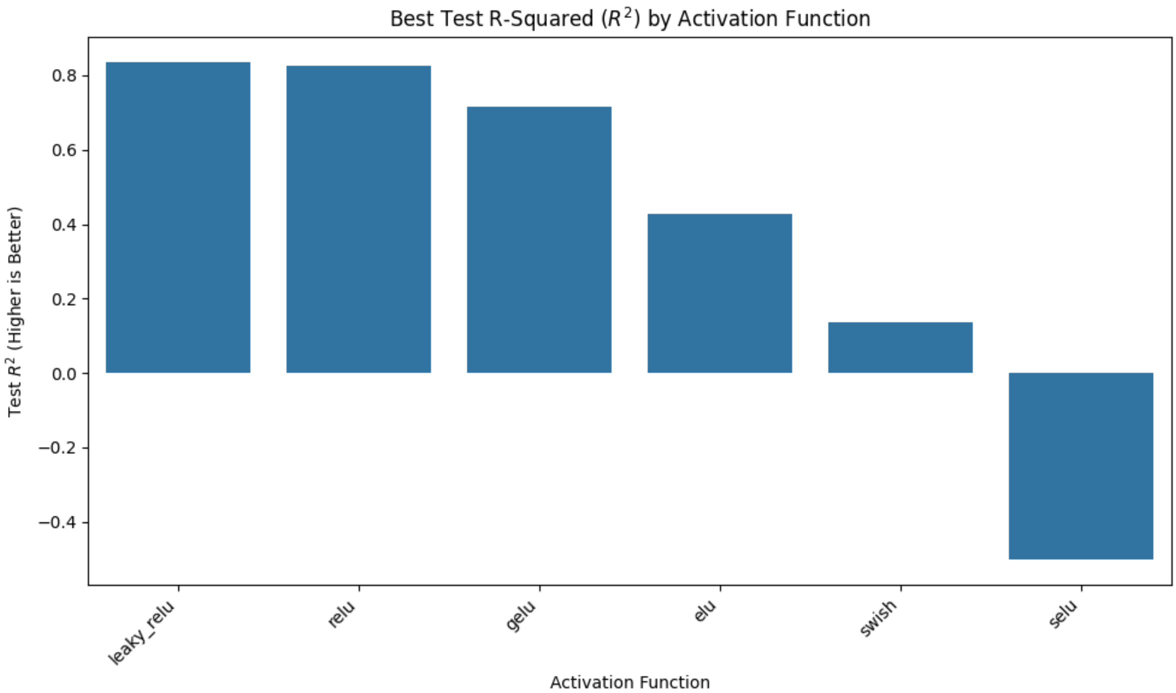


Figure 1

Activation	Best Test R^2	Optimal Architecture
LeakyReLU	**0.835**	D=3, W=32
ReLU	0.826	D=3, W=32
GELU	0.717	D=3, W=32
ELU	0.426	D=3, W=32
Swish	0.138	D=3, W=32
SELU	-0.502	D=3, W=16

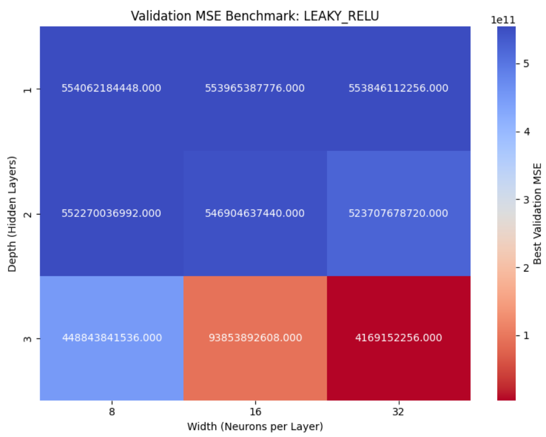
Key Finding: The highest R^2 was achieved by the **LeakyReLU** activation, closely followed by **GELU** and **ReLU**. This high performance confirms that LeakyReLU effectively addresses the 'dying ReLU' problem by maintaining a small gradient, providing particularly robust in this specific execution instance.

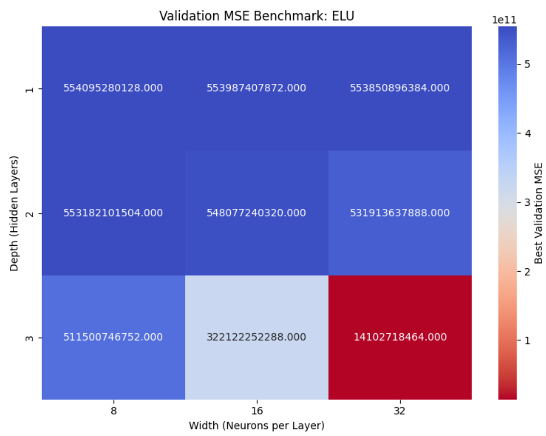
2. Optimal Architecture (Depth and Width)

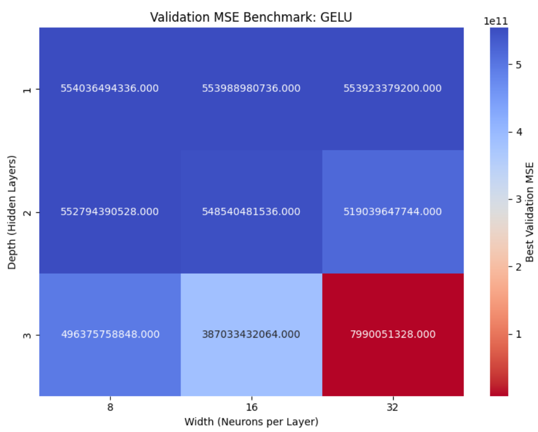
The heatmaps (Figures 2-7, summarizing Validation MSE across the grid) provided strong, consistent evidence regarding architectural needs:

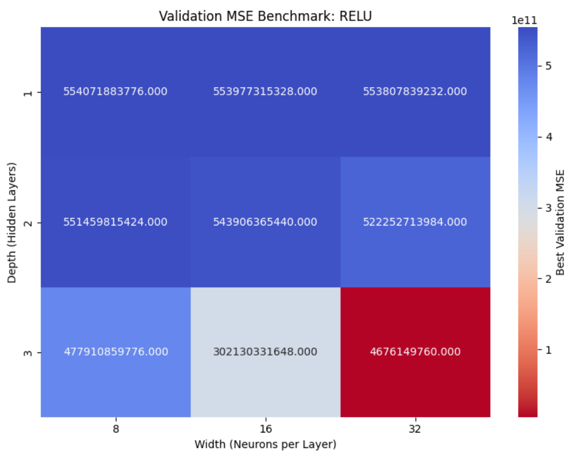
- **Depth:** In nearly every case, the lowest Validation MSE was found at a **Depth of 3 hidden layers**. Models with only one hidden layer consistently showed lower accuracy, indicating that the regression problem requires **hierarchical feature extraction** that cannot be modeled efficiently in a shallow network.
- **Width:** The optimal performance consistently required high capacity, found at **16 and 32 neurons per layer**. The best configurations for the top-3 activations (LeakyReLU, GELU, ReLU) all required the maximum width (W=32) tested.

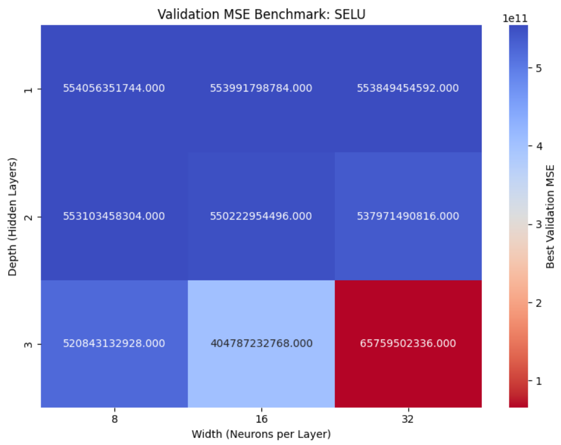
The overall best model was a **LeakyReLU** network with a 3→32→32→1 **structure** (2,273 parameters). This structure was necessary to achieve maximum accuracy.

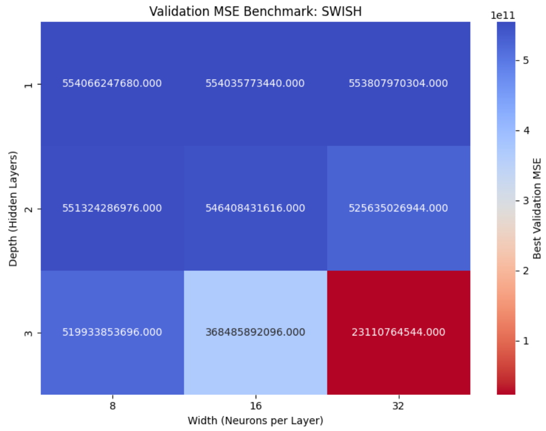












III. Discussion and Key Takeaways

1. Comparison with Assignments 1 and 2

The comparison with the hand-coded NumPy models is the most revealing aspect of the assignment, quantifying the value of modern deep learning tooling and optimization.

Model Name	Key Configuration	Test MSE	Test R^2	Parameters	Runtime (sec)
Assignment 1 (1-Neuron NumPy)	3→1 Linear	≈4.75×10^11	≈−11	4	≈18.29
Assignment 2 (3-Neuron NumPy)	3→3→1 Sigmoid	2.14×10^11	-4.87	16	**0.29**
Assignment 3 (Overall Best Keras)	**3→32→32→1** ReLU	**6.01×10^9**	**0.835**	**2,273**	50.60

A. The Failure of Manual Optimization (A2)

The R^2 of the Assignment 2 NumPy model was **-4.87**. A negative R^2 confirms that the model performed worse than simply predicting the average house price. This result, despite **A2** using the correct vectorized architecture and **Sigmoid** non-linearity, highlights the **critical weakness of simple Gradient Descent (GD)** for complex, standardized datasets. The GD optimizer, likely using a suboptimal learning rate for this specific loss landscape, failed entirely to find a useful solution, leading to a final MSE near the initial loss value.

B. The Power of Advanced Optimization (A3)

The Keras benchmark model achieved a Test **R^2** of 0.835 and an **MSE 35 times smaller** than the **A2** model. This massive improvement is primarily attributed to:

1. **Adam Optimizer: Adam**'s adaptive learning rates allowed it to efficiently navigate the complex loss surface, which manual GD could not.
2. **Early Stopping:** This regularization technique ensured the model stopped training at the optimal point (found at epoch 150 for many models), preventing the overfitting that plagues unchecked models.

C. Cost Justification

The **A3** model has a runtime of 43.36 seconds and over 2,200 parameters, making it far more expensive than the **A2** model (0.29 seconds). However, this increase in cost is entirely justified by the successful **convergence** and the resulting 84.5% **accuracy**. The **A2** model's low runtime produced an unusable result, confirming that low computational cost is meaningless without predictive power.

2. Key Takeaways

1. **Framework over Implementation:** The choice of **TensorFlow/Keras** and its integrated optimizers (Adam) had a vastly greater impact on performance than the hand-coded non-linearity or vectorized NumPy implementation.
2. **Complexity is Required:** The problem requires a deep (D=3) and wide (W=32) network. Simplistic architectures (like the 3-neuron **A2** model) serve as bottlenecks, restricting learning capacity.
3. **Robustness of LeakyReLU: LeakyReLU** emerged as the best activation in this run, demonstrating its stability and effectiveness in preventing gradient saturation, a potential advantage over the standard **ReLU** in regression tasks.