House Price Assignment:

This task, different from former 2 tasks in Chapter4, is NOT a binary or multi-class classification task. It is a regression mission.

R1: Standard setting.

2 Hidden layers. Using mae loss function.

```
    model = keras.Sequential([
        # Intermediate Layer 1: 64 units, ReLU
        layers.Dense(64, activation="relu"),
        # Intermediate Layer 2: 64 units, ReLU
        layers.Dense(64, activation="relu"),
        # Output Layer: 1 unit, no activation (linear) for predicting a continuous value
        layers.Dense(1)
```

MAE scores per fold: [1.9036751985549927, 2.5448927879333496, 2.363264322280884, 2.385288715362549]

Mean MAE: 2.30

Standard Deviation: 0.24

R2: Small Capacity

```
model = keras.Sequential([
        # Intermediate Layer 1: 32 units, ReLU
        layers.Dense(32, activation="relu"),
        # Intermediate Layer 2: 32 units, ReLU
        layers.Dense(32, activation="relu"),
        # Output Layer: 1 unit, no activation (linear) for predicting a continuous value
        layers.Dense(1)
    ])
```

Result:

MAE scores per fold: [2.0392792224884033, 2.5804147720336914, 2.4916799068450928, 2.4738986492156982]

Mean MAE: 2.40

Standard Deviation: 0.21

R3: Deeper Network

Add another hidden layer with 64 units

```
model = keras.Sequential([
        # Intermediate Layer 1: 64 units, ReLU
        layers.Dense(64, activation="relu"),
        # Intermediate Layer 2: 64 units, ReLU
        layers.Dense(64, activation="relu"),
        # Intermediate Layer 3: 64 units, ReLU
        layers.Dense(64, activation="relu"),
        # Output Layer: 1 unit, no activation (linear) for predicting a continuous value
        layers.Dense(1)
    ])
```

R

Result:

MAE scores per fold: [2.105470657348633, 2.825239658355713, 2.440906524658203, 2.2629013061523438]

Mean MAE: 2.41

Standard Deviation: 0.27

R4: Fewer Epochs

Turn 100 epochs to 50 epochs

Result:

In this setting, every time when I train the model and do the verification, the loss will be slightly different, all of them resulting in 2.25-2.5, this is acceptable. The random weight can cause this result when the scale of sample is small.

Results:

First time:

```
--- K-Fold Validation Results ---

MAE scores per fold: [2.003215789794922, 2.735074758529663, 2.807130813598633, 2.423522472381592]

Mean MAE: 2.49

Standard Deviation: 0.32
```

Second time:

```
--- K-Fold Validation Results ---

MAE scores per fold: [1.9147340059280396, 2.438098192214966, 2.6078944206237793, 2.5237200260162354]

Mean MAE: 2.37

Standard Deviation: 0.27
```

Third time

Summary:

The **Standard Configuration (R1)** proved to be the **optimal model** for this specific regression task, achieving the lowest Mean MAE of 2.30 (an average error of $2,300 per prediction).

The alternative experiments demonstrate the challenges of training deep learning models on small datasets:

1. **Model Capacity (R2 and R3):** Both decreasing the capacity (R2) and increasing the depth (R3) immediately led to a worse average MAE (2.40 and 2.41, respectively). The standard 2-layer, 64-unit architecture struck the best balance between complexity and regularization for the available data.
2. **Stability (R4):** Reducing the training epochs to 50 (R4) resulted in increased instability (higher Standard Deviation) and volatile Mean MAE results, highlighting the influence of **random weight initialization** on final performance when using limited samples.

In summary, the best strategy was the robust, moderately-sized architecture of R1, which successfully minimized the average prediction error while maintaining high stability across the K-fold validation sets.