*mehmet soner korucu - 040190225*

*EHB420E | CRN:10342*

*CoIn drop predIctIon*

Artificial Neural Networks

# TABLE OF CONTENTS

* Problem view
* Data collection
* Model selection
  + Convolution
  + Dense
* Training and Evaluation
* Conclusion

Github link: <https://github.com/mkorucu/final>

Google colab link: <https://colab.research.google.com/drive/10x75mdFhNcpmkIRSrHoT2NwYDPn5Lx9v?usp=sharing>

Problem View

The problem consists of flipping a coin from a high and predicting the landing location and orientation of the coin. In order to detect its location and orientation, we need to collect data and create model using this model.

Data Collection

I used photo capturing to collect the data. I released the coin from a 0.8m tall desk and spread a white sheet under the coin. I chose the center of the sheet as origin. I dropped the coin in three different orientations: Heads up, Tails up and vertical. I repeated the drop test for 50 times in each orientation and wrote the results in an excel file. Then, I organized the photos as 70% train, 20% validation and 10% test. I also considered heads (tura) as 0 and tails (yazi) as 1.

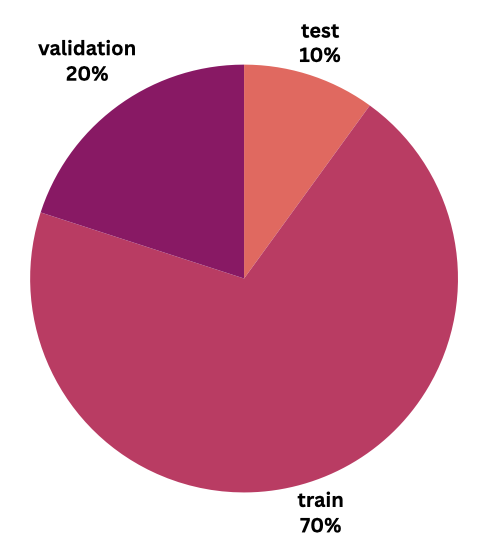
madeni para, kişi, şahıs, maden, para taşıma içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 1: Heads Up Coin Drop

Figure 2: White Sheet and its origin

Table 1: X-Y dIstrIbutIon of heads up coın drops

Data classification:

%70 -> Training data

35 - 35 - 35

%20 -> validation data

10 - 10 - 10

%10 -> test

5 - 5 - 5

Model Selection

Convolutional neural networks (CNN) is selected for deep learning model. CNN is a combination of convolution and dense layers, designed to capture both image features and additional scalar information. The model consists of various inputs:

photo input: the collected photos that is turned into numpy array and extracted from its pixels. Their shape is [320x320x3].

X values: Dropped coin’s X values in the coordinate system.

Y values: Dropped coin’s Y values in the coordinate system.

Bool values: Dropped coin’s orientation.

**Convolution**

Conv2D convolution is used to initialize the convolution layers with 32 filters, 3x3 size, applied.

MaxPooling2D adds a max-pooling layer after the first convolutional layer.

Again, another convolution with 64 filters and max pooling is applied for reducing spatial dimensions. After all, the resulted max-pooling is flattened to 1D and ready to be connected to dense layers.

**Dense**

Lastly, the final dense layer consists of a single neuron with a sigmoid activation function. This architecture is common for binary classification problems. The sigmoid activation ensures the output is between 0 and 1.

Training and Evaluation

A neural network model is created using the Keras Model class and compiled with Adam optimizer. Loss parameter is set to a list containing:

* Mse (mean squared error)
* binary\_crossentropy

After that, model is ready to be trained.

model.fit(photo\_train, [x\_train, y\_train, bool\_train], epochs=50, batch\_size=32, validation\_data=(photo\_valid, [x\_valid, y\_valid, bool\_valid]))

* photo\_train is used for train input
* x\_train, y\_train and bool\_train variables are used as train output
* photo\_valid is used for validation input
* x\_valid, y\_valid and bool\_valid is used for validation output.

After 50 epochs:

* **x\_output\_loss: 0.0455** - Loss for the **x\_output** (regression task).
* **y\_output\_loss: 0.0204** - Loss for the **y\_output** (regression task).
* **output\_bool\_loss: 0.0020** - Loss for the **output\_bool** (binary classification task).
* **x\_output\_mae: 0.1724** - Mean Absolute Error (MAE) for the **x\_output** task.
* **y\_output\_mae: 0.1097** - MAE for the **y\_output** task.
* **output\_bool\_accuracy: 1.0000** - Accuracy for the **output\_bool** task.
* **val\_loss: 473.1600** - Total validation loss.
* **val\_x\_output\_loss: 322.3542** - Validation loss for **x\_output**.
* **val\_y\_output\_loss: 147.9749** - Validation loss for **y\_output**.
* **val\_output\_bool\_loss: 2.8310** - Validation loss for **output\_bool**.
* **val\_x\_output\_mae: 15.0850** - Validation MAE for **x\_output**.
* **val\_y\_output\_mae: 9.9936** - Validation MAE for **y\_output**.
* **val\_output\_bool\_accuracy: 0.3667** - Validation accuracy for **output\_bool**.

**Evaluation results of the test dataset:**

1/1 [==============================] - 0s 55ms/step - loss: 314.7660 - x\_output\_loss: 208.4204 - y\_output\_loss: 104.2841 - output\_bool\_loss: 2.0616 - x\_output\_mae: 9.9265 - y\_output\_mae: 7.1071 - output\_bool\_accuracy: 0.5333

Test Loss: 314.7659912109375

Test MAE for x\_output: 208.4203643798828

Test MAE for y\_output: 104.28407287597656

Test Accuracy for output\_bool: 2.0615718364715576

1/1 [==============================] - 0s 88ms/step

**Prediction for test dataset (15 photos):**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Flip style | Test\_x | Test\_y | Test\_res | Predicted\_x | Predicted\_y | Predicted\_res |
| Vertical | 11 | 1 | 0 | 7.45 | -6.66 | 1 |
| vertical | -16 | -7 | 0 | -14.75 | 3.1 | 1 |
| Vertical | -6 | -3 | 1 | -0.2 | -4.2 | 1 |
| Vertical | -21 | 2 | 1 | -5.6 | -9.9 | 1 |
| Vertical | -11 | -6 | 1 | -6.6 | 1.6 | 0 |
| Heads up | 9 | 6 | 1 | 10.1 | 0.7 | 0 |
| Heads up | -13 | 3 | 1 | -1.4 | 4 | 0 |
| Heads up | -5 | -2 | 1 | 1.2 | 4.7 | 0 |
| Heads up | 13 | -2 | 0 | 10.3 | -3.2 | 0 |
| Heads up | 10 | 0 | 1 | 15.5 | -3.9 | 0 |
| Tails up | 4 | 5 | 0 | 5.2 | -0.5 | 0 |
| Tails up | -32 | -9 | 0 | 13.1 | -14.3 | 0 |
| Tails up | 32 | -16 | 0 | 16.1 | -8.2 | 0 |
| Tails up | 24 | -11 | 1 | 8.8 | -8.7 | 1 |
| Tails up | 14 | 25 | 0 | 10.9 | -4.2 | 0 |

**Model prediction graphs**