Projectile Motion

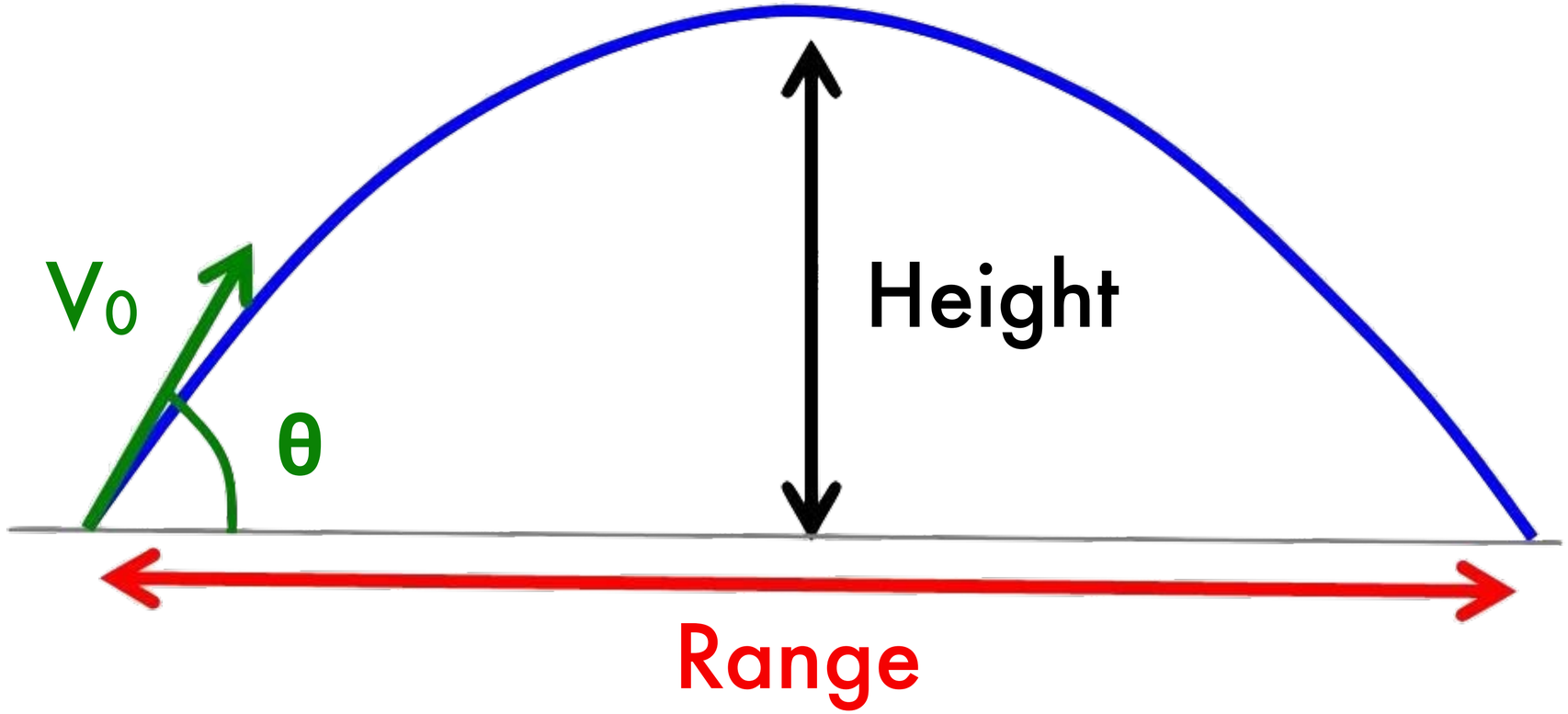
- Machine Learning Report -

0611514 材料系 張郁琦
0611527 材料系 曹心譯
0853436 資管碩 高楚晴

- Outline -

- Preparation -
- Generating Data -
- Building Model -
- Data Preprocessing -
- Training Model -
- Result & Discussion -
- Conclusion -

- Projectile Motion -



- input -

V_0, θ

- output -

H, R

- Preparation -

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.layers import Dropout
from tensorflow.keras.optimizers import Adam, SGD
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, KFold
from sklearn.preprocessing import MinMaxScaler
import math
import random
from datetime import datetime
import tensorboard
import seaborn as sns
%load_ext tensorboard
!rm -rf ./logs/
```

Import the module and the package for the following use.

- Generating Data -

```
def data_generator():
    vel_data = np.random.uniform(1,100,size=20000)
    ang_data = np.random.uniform(1,89,size=20000)
    rad_data = np.radians(ang_data)
    H_data = (vel_data**2)*(np.sin(rad_data)**2)/(2*9.8)
    R_data = (vel_data**2)*(np.sin(rad_data*2))/9.8
    sns.distplot(vel_data, hist=True)
    plt.show()
    sns.distplot(ang_data, hist=True)
    plt.show()
    return vel_data, rad_data, H_data, R_data
```

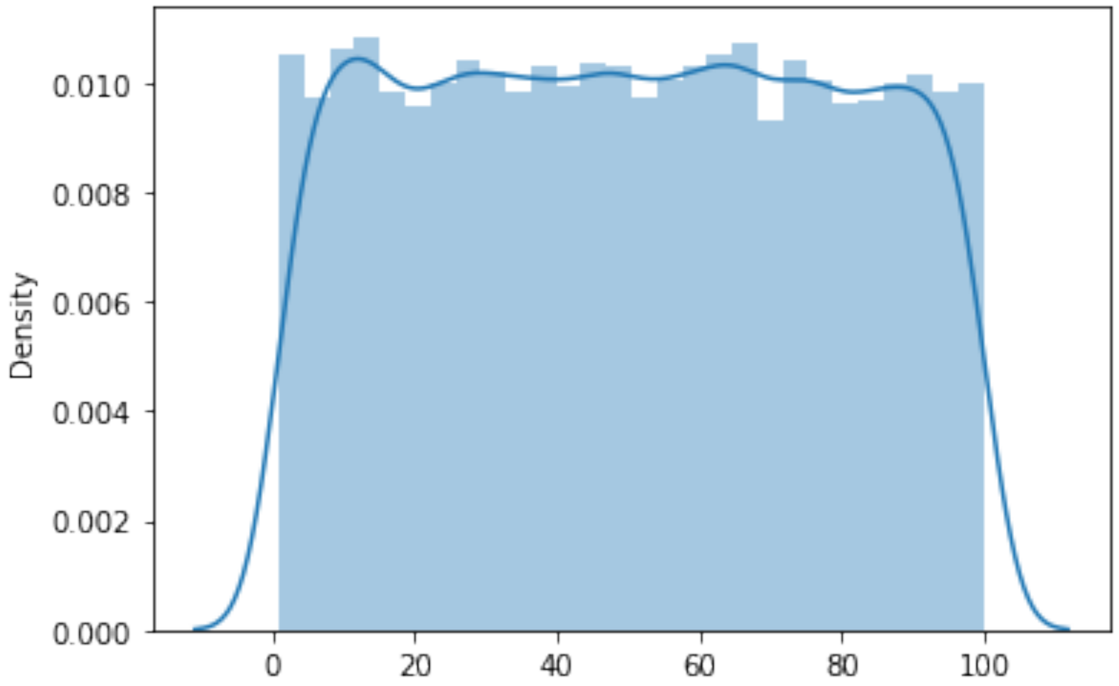
Use these equations:

$$H = \frac{v_0^2 \sin^2\theta}{2g} \qquad R = \frac{v_0^2 \sin 2\theta}{g}$$

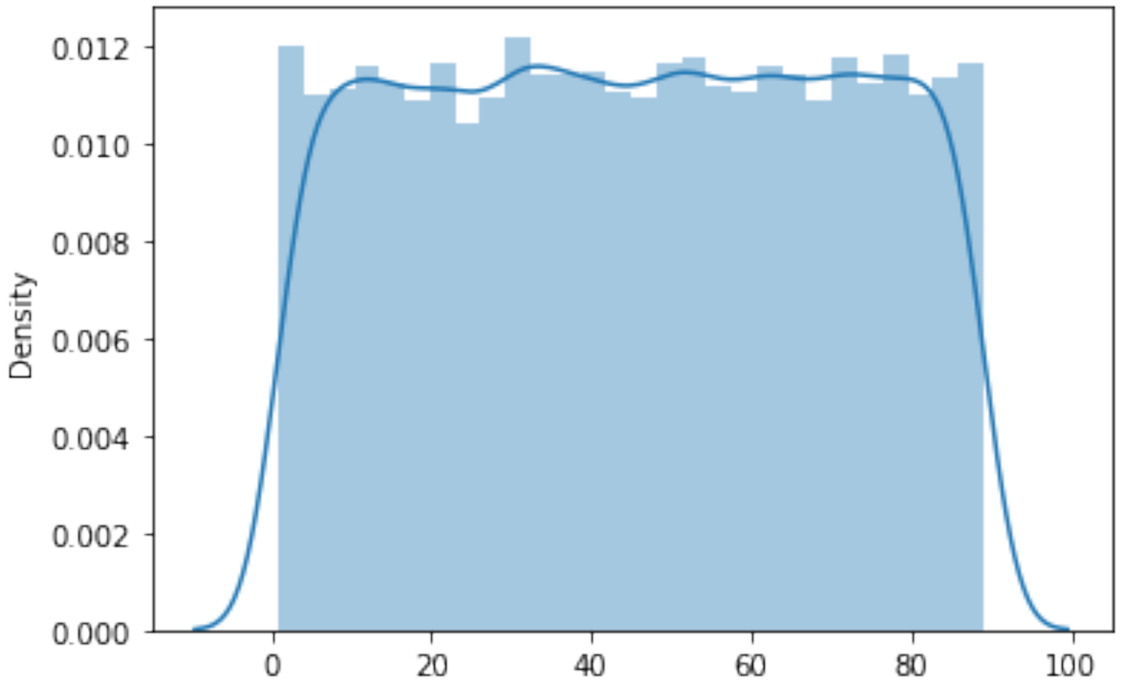
to generate the output data.

- Generating Data -

Velocity



Angle



- Building Model -

```
epochs = 150
model = Sequential([
Dense(128, activation = 'relu', input_shape=(2, )),
Dense(64, activation = 'relu'),
Dense(64, activation = 'relu'),
Dense(32, activation = 'relu'),
Dense(2, activation = 'relu')])
model.compile(loss='mae', optimizer= Adam(learning_rate = 0.03/epochs), metrics= ['accuracy'])
print(model.summary())
logdir="logs/fit/" + datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=logdir)
```

- Data Preprocessing -

Rescale/Normalize data (MinMaxScaler)

```
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X_val = scaler.transform(X_val)
```

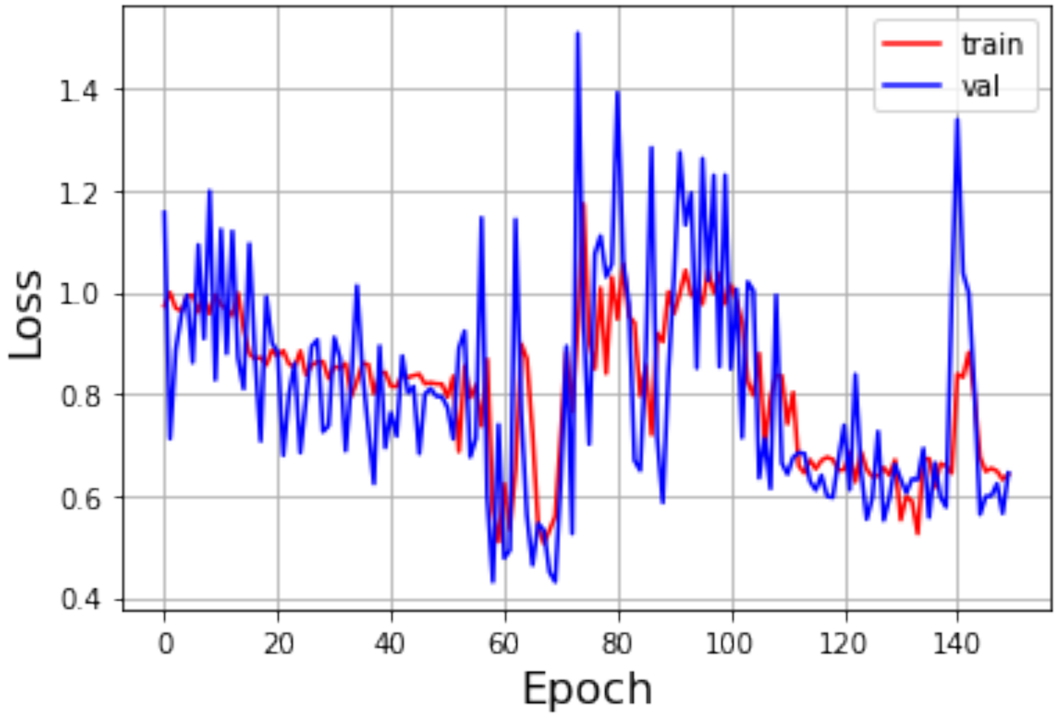
- Training Model -

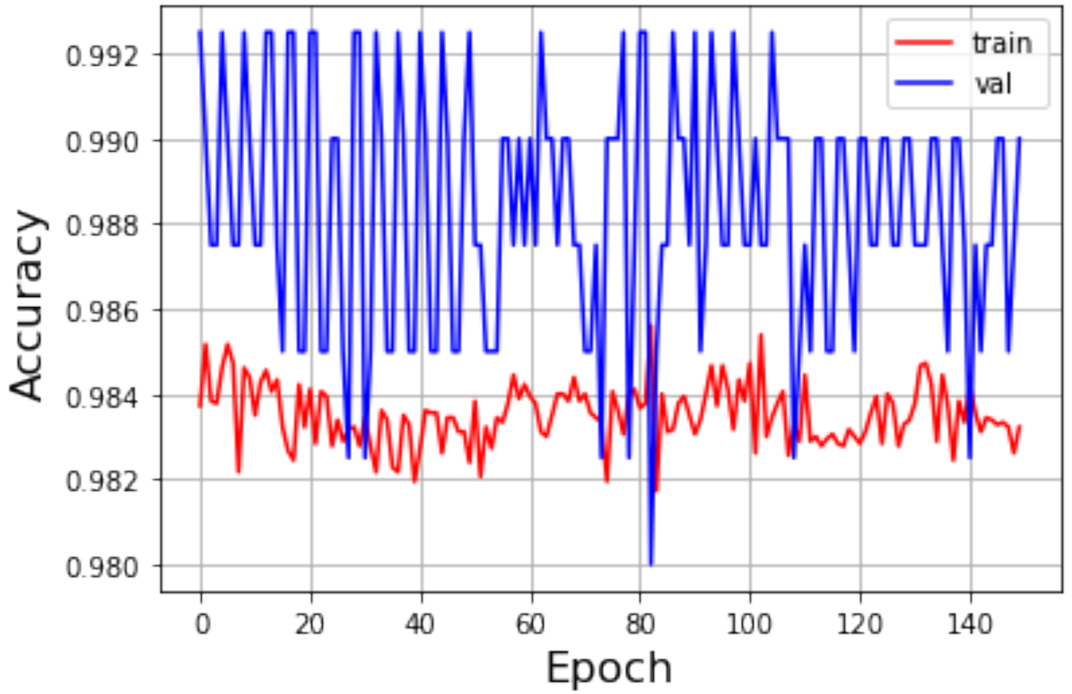
Cross-validation (K-Fold)

```
k = 10
kf = KFold(n_splits = k, shuffle = True, random_state = 42)
for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index,:], X[test_index,:]
    Y_train, Y_test = Y[train_index,:], Y[test_index,:]
    X_test, X_val, Y_test, Y_val = train_test_split(X_test, Y_test, test_size = 0.2)
    results = model.fit(X_train,Y_train,batch_size=2048,
                        epochs=epochs,validation_data=(X_val,Y_val),
                        callbacks=[tensorboard_callback])
    preds = model.predict(X_test)
```

- Result & Discussion -

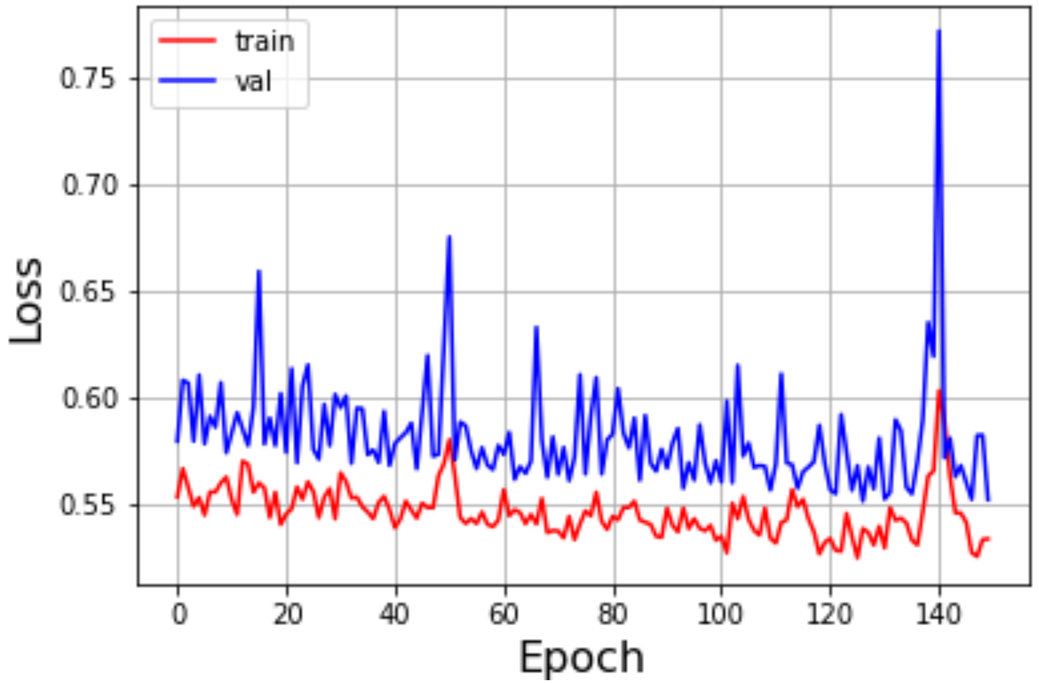
Compare the result of the loss & accuracy with or **without** “optimizer = Adam(learning_rate = 0.03/epochs)”.

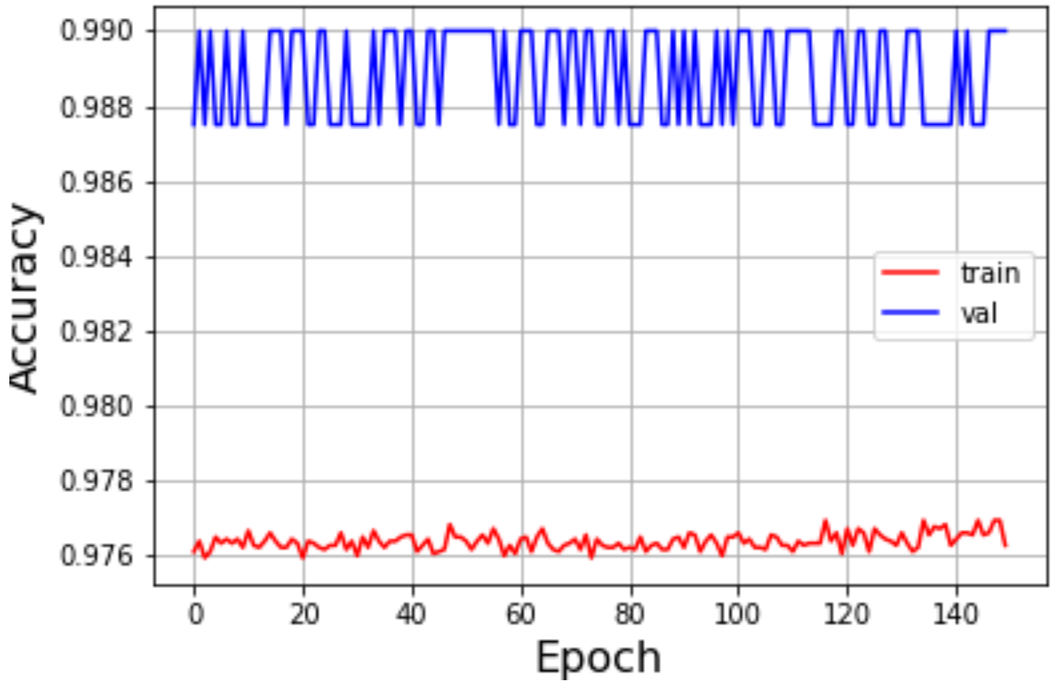




- Result & Discussion -

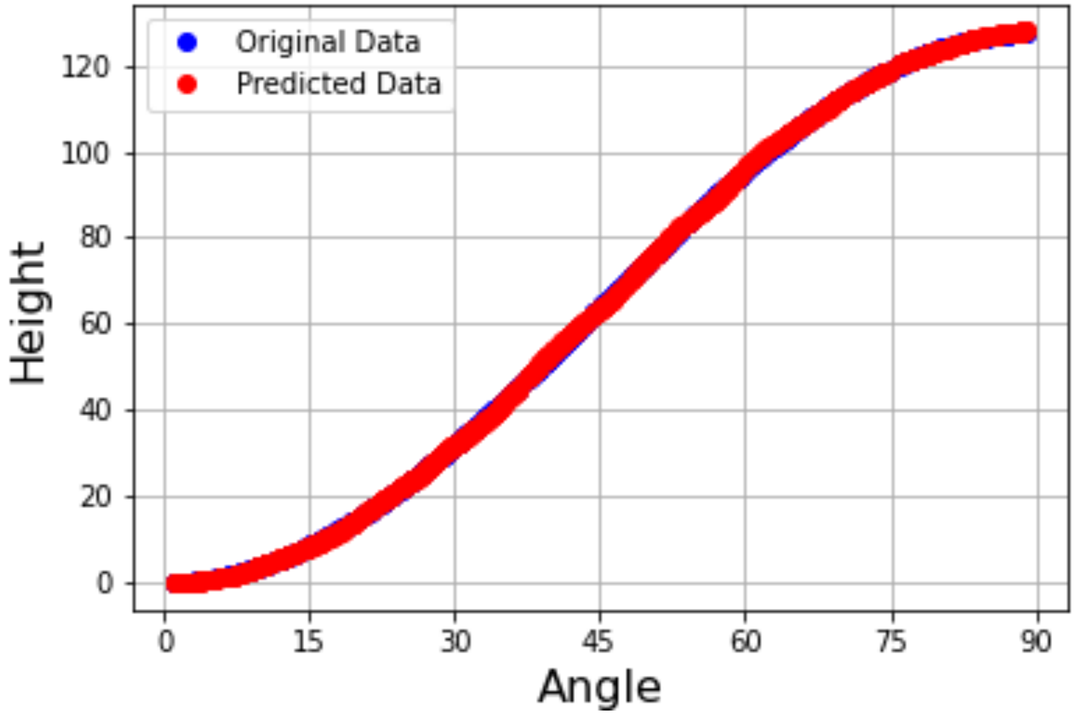
Compare the result of the loss & accuracy **with** or without "optimizer = Adam(learning_rate = 0.03/epochs)".

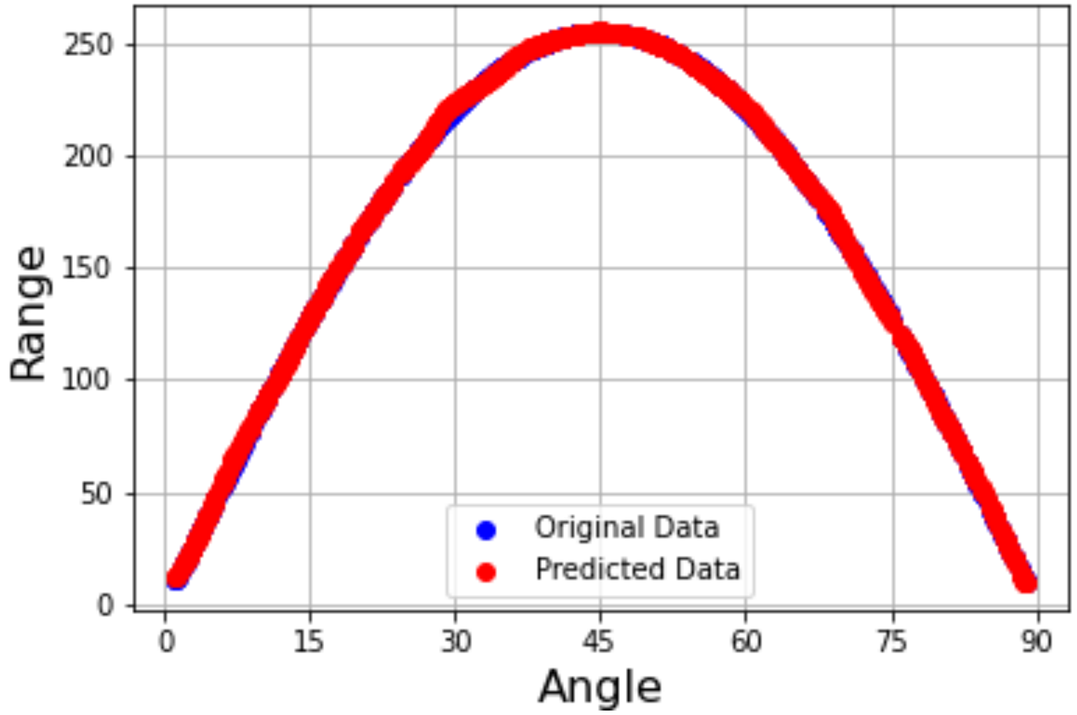




- Result & Discussion -

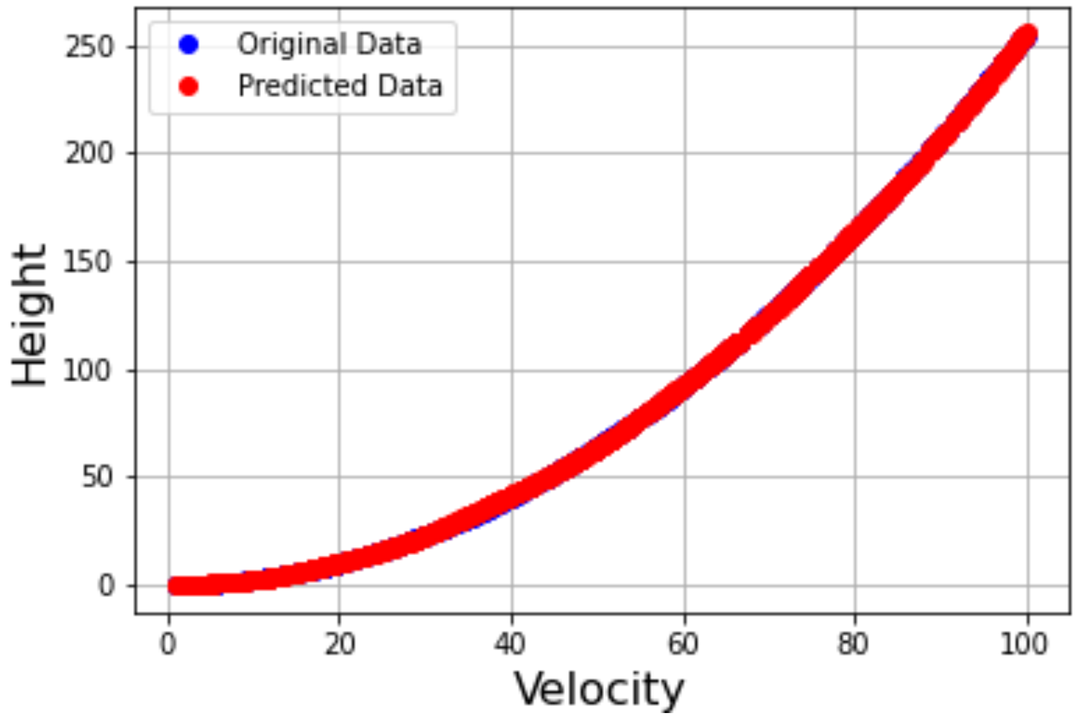
Compare the original and the predicated data of the relationship between angle and height/range.

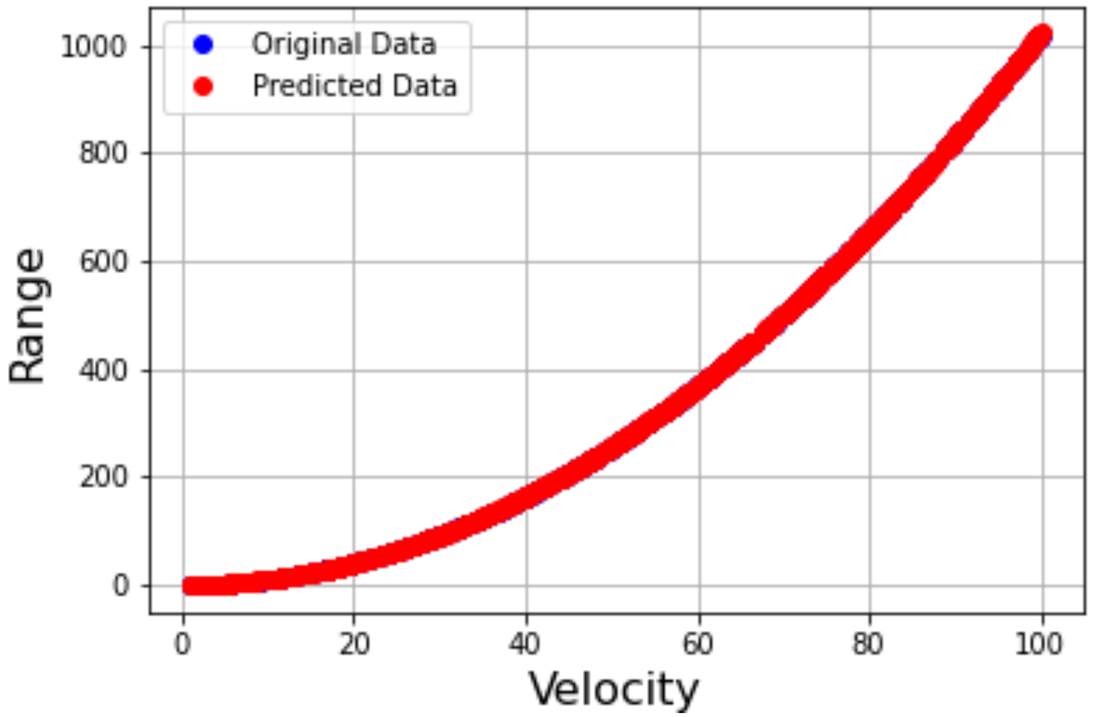




- Result & Discussion -

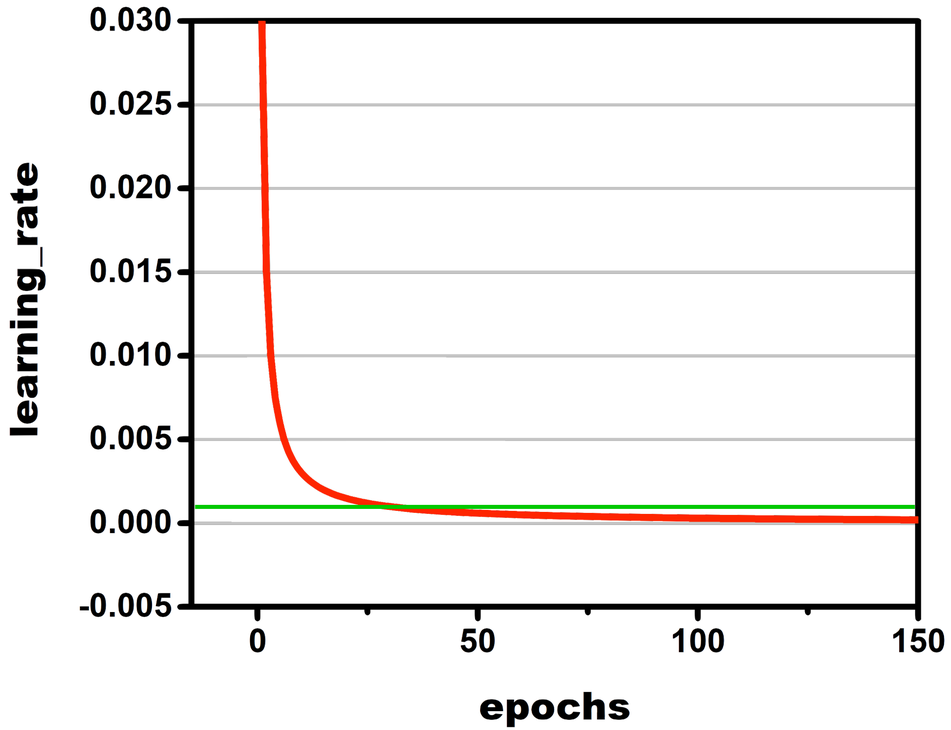
Compare the original and the predicated data of the relationship between velocity and height/range.

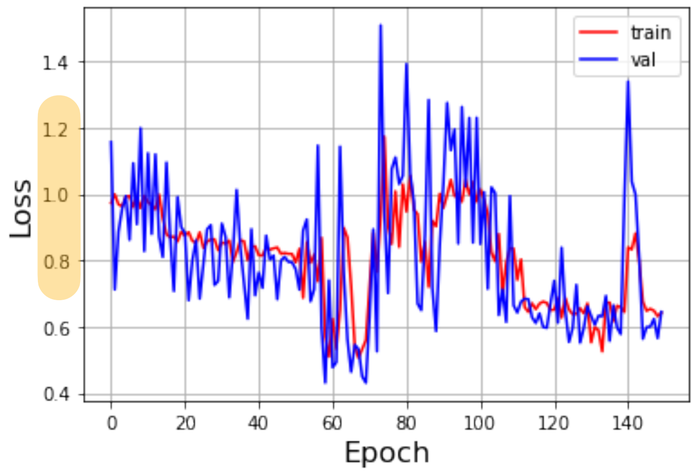


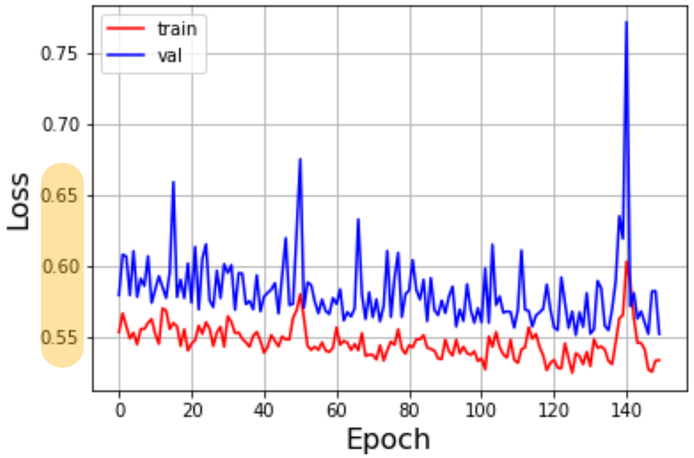


- Conclusion -

The loss decreasing as the learning rate decay while the epoch increase.





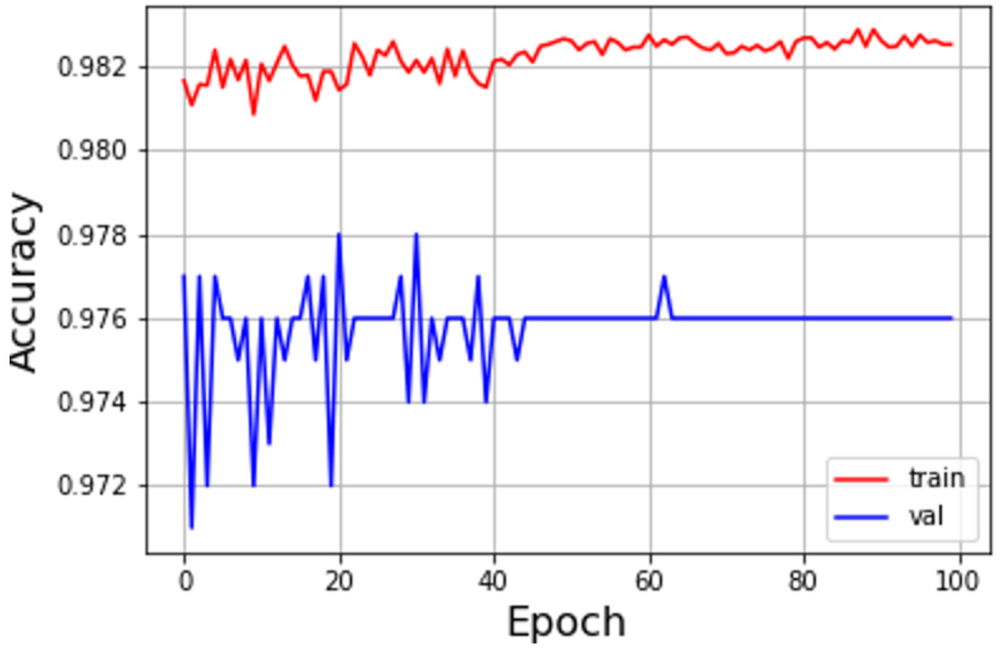


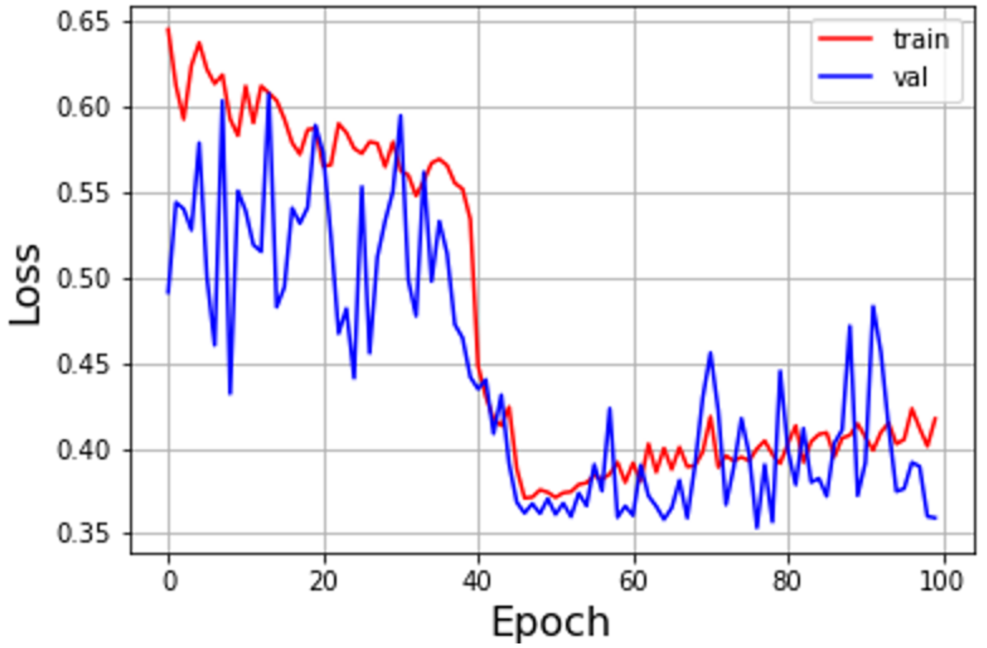
- Modified -

with learning-decay

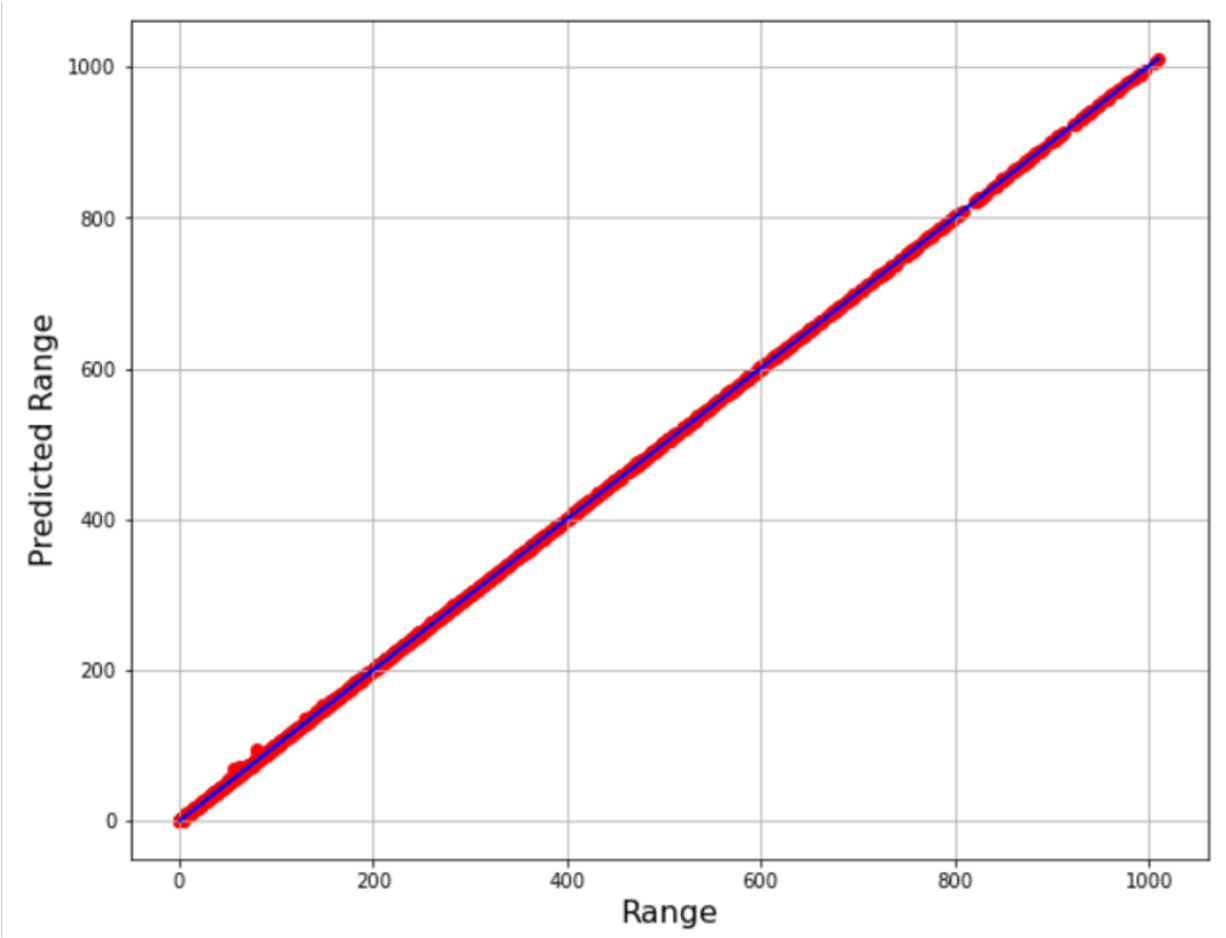
```python
#Gernate Data
def data_generator():
    vel_data = np.random.uniform(1,100,size=50000).astype(np.float32)
    ang_data = np.random.uniform(1,89,size=50000).astype(np.float32)
    rad_data = np.radians(ang_data)
    H_data = (vel_data**2)*(np.sin(rad_data)**2)/(2*9.8)
    R_data = (vel_data**2)*(np.sin(rad_data*2))/9.8
    return vel_data, rad_data, H_data, R_data
```

```python
#Build Model
epochs = 100
model = Sequential([
Dense(252, activation = 'relu', input_shape=(2, )),
Dense(126, activation ='relu'),
Dense(64, activation = 'relu'),
Dense(32, activation = 'relu'),
Dense(2, activation = 'relu')])
model.compile(loss='mae', optimizer= Adam(learning_rate = 0.03/epochs),
print(model.summary())
```



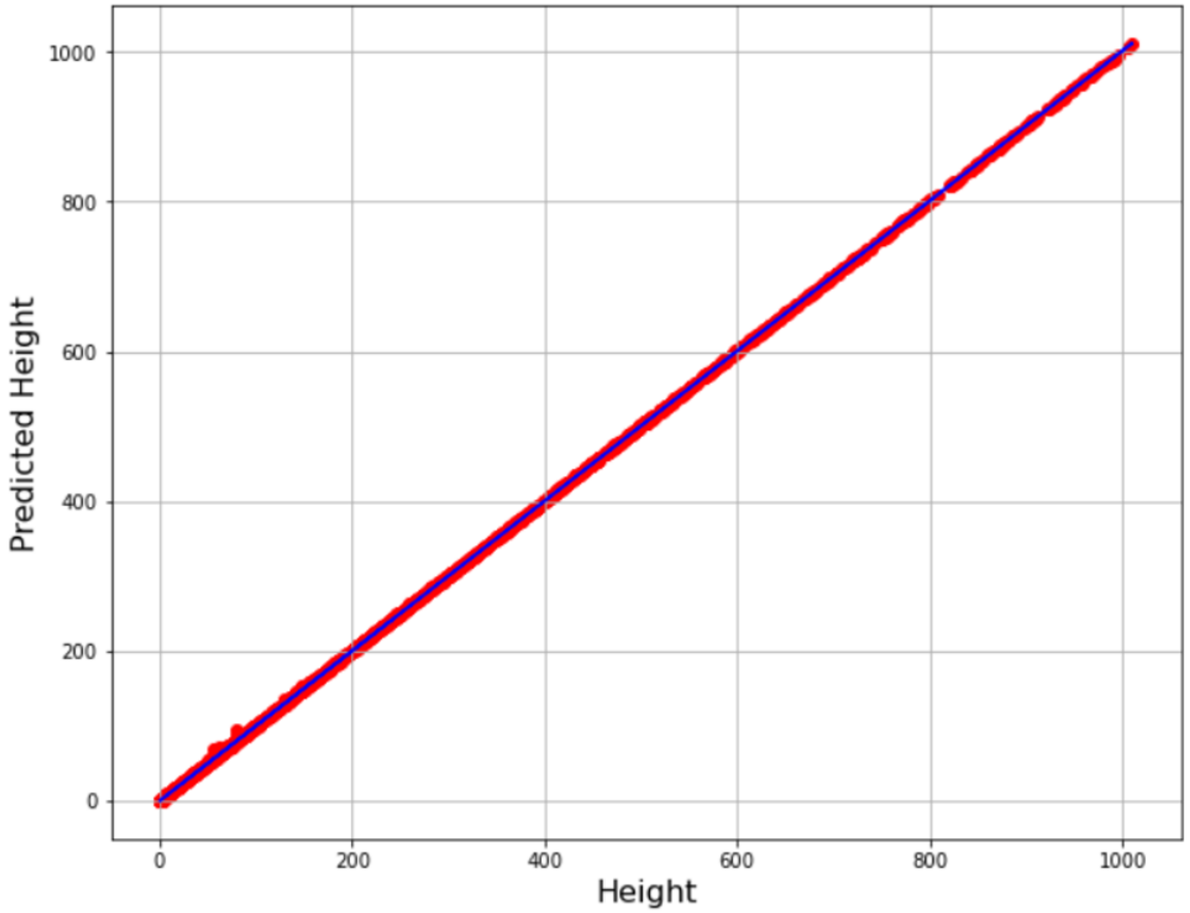


With learning-decay



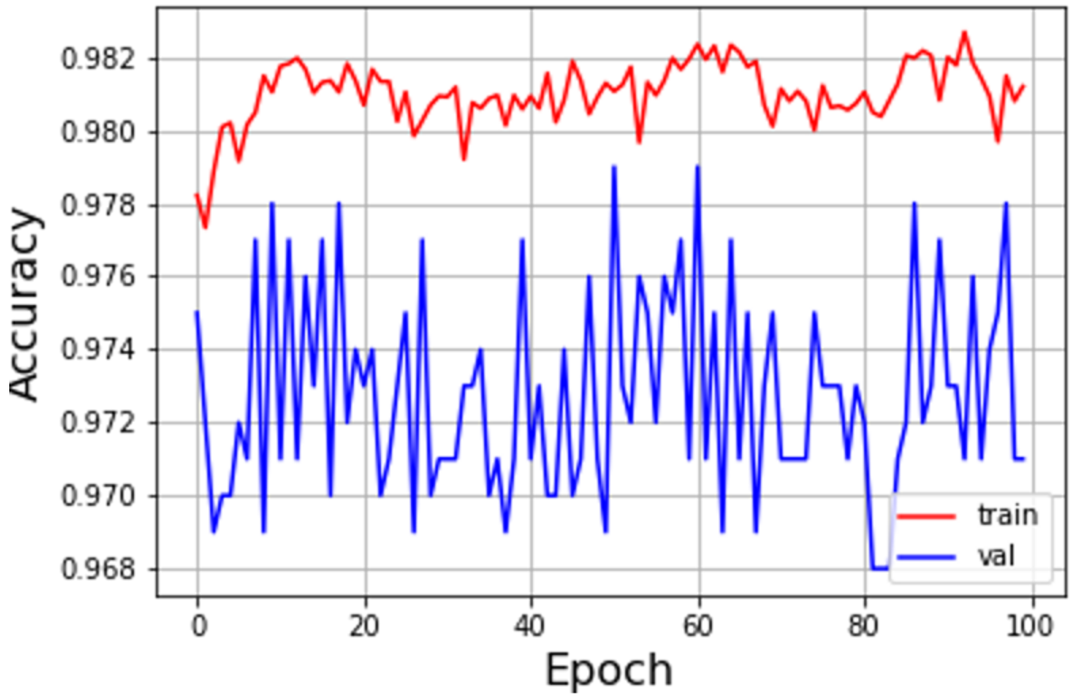
隨機產生速度與角度，比較公式與預測的距離
距離共變異數： 56761.84046276871
公式距離標準差： 238.1263986958496
預測距離標準差： 238.3699977769658
Pearson Correlation Coefficient: 0.9999938574427241
R-squared from Linear Regression: 0.9999877149236484
迴歸直線：y = 1.00102 x + 0.0729

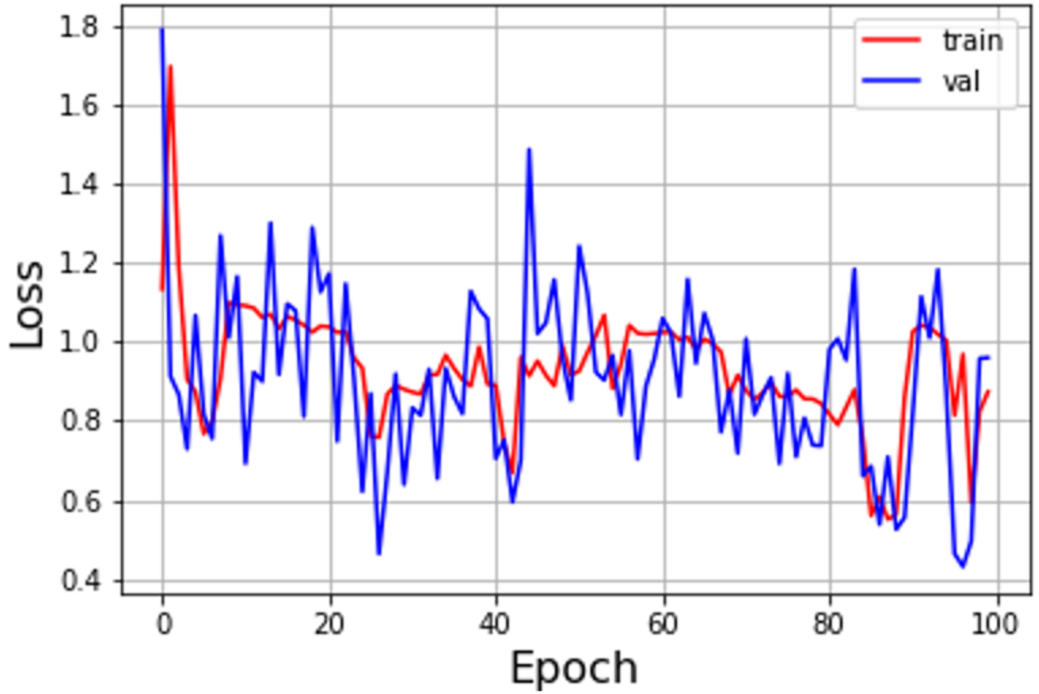
With learning-decay



隨機產生速度與角度，比較公式與預測的高度
高度共變異數： 11271.556858179085
公式高度標準差： 106.09929777303628
預測高度標準差： 106.2364222638473
Pearson Correlation Coefficient: 0.9999953030724262
迴歸直線：y = 1.00129 x + 0.04104
線性迴歸下的R-squared： 0.9999906061669013

Without learning-decay





- Thank you for your attention -