

Lab 3

Daniel Mehta

Question:

Apply all the techniques you learned this week including network tweak and train a regression model network on the Boston housing dataset. Please download dataset from online

```
In [17]: # Imports
import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, BatchNormalization, Dropout
```

```
In [4]: # load data
boston = fetch_openml(name="boston", version=1, as_frame=True)
X = boston.data
y = boston.target

#split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=55)
```

```
In [5]: # standardize the inputs
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
In [6]: # initialize the model
model = Sequential()






















#input and hidden layer
model.add(Dense(64, input_shape=(X_train.shape[1],), activation='relu', kernel_initializer='he_normal'))
model.add(BatchNormalization())
model.add(Dropout(0.2))




#second hidden layer
model.add(Dense(32, activation='relu', kernel_initializer='he_normal'))
model.add(BatchNormalization())
model.add(Dropout(0.2))






















# output layer for regression. No activation
model.add(Dense(1, activation='linear'))
```







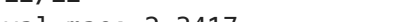
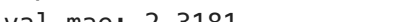
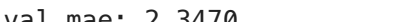
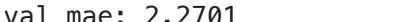


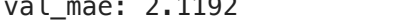
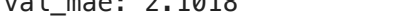

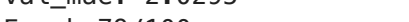





```
In [7]: # compile the model
model.compile(optimizer='adam', loss='mean_squared_error', metrics=['mae'])
```

```
In [8]: #train model  
history = model.fit(  
    X_train, y_train,  
    validation_split=0.1,  
    batch_size=32,  
    epochs=100,  
    verbose=1  
)
```








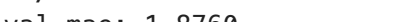

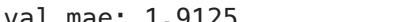




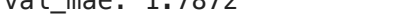


Epoch 1/100
12/12  2s 18ms/step - loss: 572.2478 - mae: 22.3856 - val_loss: 645.6
323 - val_mae: 23.4121
Epoch 2/100
12/12  0s 6ms/step - loss: 554.3553 - mae: 22.0161 - val_loss: 625.88
23 - val_mae: 23.2083
Epoch 3/100
12/12  0s 6ms/step - loss: 575.0591 - mae: 22.6570 - val_loss: 608.37
55 - val_mae: 23.0231
Epoch 4/100
12/12  0s 6ms/step - loss: 531.4358 - mae: 22.0117 - val_loss: 592.58
82 - val_mae: 22.8572
Epoch 5/100
12/12  0s 8ms/step - loss: 504.0962 - mae: 21.4640 - val_loss: 576.71
23 - val_mae: 22.6848
Epoch 6/100
12/12  0s 8ms/step - loss: 521.6538 - mae: 21.8792 - val_loss: 561.52
01 - val_mae: 22.5068
Epoch 7/100
12/12  0s 8ms/step - loss: 473.1009 - mae: 21.0427 - val_loss: 548.73
70 - val_mae: 22.3550
Epoch 8/100
12/12  0s 8ms/step - loss: 509.3601 - mae: 21.6957 - val_loss: 535.12
48 - val_mae: 22.1833
Epoch 9/100
12/12  0s 9ms/step - loss: 507.0719 - mae: 21.6975 - val_loss: 520.95
07 - val_mae: 21.9871
Epoch 10/100
12/12  0s 7ms/step - loss: 490.2766 - mae: 21.4664 - val_loss: 506.75
08 - val_mae: 21.7578
Epoch 11/100
12/12  0s 7ms/step - loss: 471.3423 - mae: 21.0147 - val_loss: 493.25
95 - val_mae: 21.5395
Epoch 12/100
12/12  0s 7ms/step - loss: 444.0689 - mae: 20.5931 - val_loss: 479.29
32 - val_mae: 21.2898
Epoch 13/100
12/12  0s 7ms/step - loss: 440.3524 - mae: 20.4432 - val_loss: 467.94
87 - val_mae: 21.0901
Epoch 14/100
12/12  0s 7ms/step - loss: 450.0169 - mae: 20.6902 - val_loss: 455.83
32 - val_mae: 20.8577
Epoch 15/100
12/12  0s 8ms/step - loss: 426.2607 - mae: 20.1650 - val_loss: 444.47
15 - val_mae: 20.6391
Epoch 16/100
12/12  0s 7ms/step - loss: 412.9320 - mae: 19.8164 - val_loss: 430.68
46 - val_mae: 20.3487
Epoch 17/100
12/12  0s 8ms/step - loss: 395.4991 - mae: 19.4343 - val_loss: 419.51
20 - val_mae: 20.1055
Epoch 18/100
12/12  0s 7ms/step - loss: 394.4098 - mae: 19.4118 - val_loss: 407.11
78 - val_mae: 19.8230
Epoch 19/100
12/12  0s 7ms/step - loss: 406.3126 - mae: 19.5625 - val_loss: 392.83
40 - val_mae: 19.4904
Epoch 20/100
12/12  0s 7ms/step - loss: 374.5716 - mae: 18.8664 - val_loss: 380.69
35 - val_mae: 19.1948
Epoch 21/100
12/12  0s 7ms/step - loss: 357.5237 - mae: 18.4437 - val_loss: 366.03

55 - val_mae: 18.8218
Epoch 22/100
12/12  0s 6ms/step - loss: 347.6257 - mae: 18.1276 - val_loss: 348.78
32 - val_mae: 18.3708
Epoch 23/100
12/12  0s 7ms/step - loss: 326.3197 - mae: 17.5411 - val_loss: 336.47
29 - val_mae: 18.0426
Epoch 24/100
12/12  0s 8ms/step - loss: 321.6734 - mae: 17.3424 - val_loss: 319.51
11 - val_mae: 17.5848
Epoch 25/100
12/12  0s 7ms/step - loss: 320.4146 - mae: 17.3824 - val_loss: 305.70
02 - val_mae: 17.1979
Epoch 26/100
12/12  0s 7ms/step - loss: 324.5984 - mae: 17.3948 - val_loss: 292.00
24 - val_mae: 16.8111
Epoch 27/100
12/12  0s 8ms/step - loss: 264.8664 - mae: 15.7745 - val_loss: 279.29
25 - val_mae: 16.4368
Epoch 28/100
12/12  0s 7ms/step - loss: 278.2612 - mae: 16.0766 - val_loss: 264.78
31 - val_mae: 15.9944
Epoch 29/100
12/12  0s 15ms/step - loss: 266.7070 - mae: 15.7028 - val_loss: 251.0
413 - val_mae: 15.5602
Epoch 30/100
12/12  0s 6ms/step - loss: 255.3811 - mae: 15.3551 - val_loss: 237.37
02 - val_mae: 15.1143
Epoch 31/100
12/12  0s 7ms/step - loss: 239.1788 - mae: 14.8167 - val_loss: 224.37
83 - val_mae: 14.6685
Epoch 32/100
12/12  0s 7ms/step - loss: 226.8656 - mae: 14.4853 - val_loss: 209.94
12 - val_mae: 14.1728
Epoch 33/100
12/12  0s 7ms/step - loss: 240.7354 - mae: 14.9262 - val_loss: 194.13
54 - val_mae: 13.6133
Epoch 34/100
12/12  0s 8ms/step - loss: 203.8279 - mae: 13.5942 - val_loss: 182.74
96 - val_mae: 13.1983
Epoch 35/100
12/12  0s 7ms/step - loss: 174.3922 - mae: 12.6656 - val_loss: 172.69
49 - val_mae: 12.8243
Epoch 36/100
12/12  0s 8ms/step - loss: 182.8582 - mae: 12.9454 - val_loss: 165.19
09 - val_mae: 12.5218
Epoch 37/100
12/12  0s 7ms/step - loss: 163.2609 - mae: 12.0394 - val_loss: 152.98
71 - val_mae: 12.0270
Epoch 38/100
12/12  0s 7ms/step - loss: 162.2883 - mae: 12.0248 - val_loss: 136.93
52 - val_mae: 11.3695
Epoch 39/100
12/12  0s 7ms/step - loss: 143.5146 - mae: 11.2054 - val_loss: 124.51
32 - val_mae: 10.8171
Epoch 40/100
12/12  0s 7ms/step - loss: 126.4199 - mae: 10.5156 - val_loss: 112.92
49 - val_mae: 10.2595
Epoch 41/100
12/12  0s 8ms/step - loss: 117.7734 - mae: 10.1040 - val_loss: 104.89
13 - val_mae: 9.8498
Epoch 42/100

12/12  0s 8ms/step - loss: 130.4870 - mae: 10.4952 - val_loss: 94.260
9 - val_mae: 9.3054
Epoch 43/100
12/12  0s 7ms/step - loss: 98.0411 - mae: 8.8990 - val_loss: 84.7680
- val_mae: 8.7788
Epoch 44/100
12/12  0s 7ms/step - loss: 87.1437 - mae: 8.3817 - val_loss: 77.7519
- val_mae: 8.3725
Epoch 45/100
12/12  0s 7ms/step - loss: 84.6622 - mae: 8.2526 - val_loss: 73.1158
- val_mae: 8.1037
Epoch 46/100
12/12  0s 8ms/step - loss: 94.3173 - mae: 8.3505 - val_loss: 65.1629
- val_mae: 7.6443
Epoch 47/100
12/12  0s 8ms/step - loss: 74.0808 - mae: 7.7197 - val_loss: 59.2572
- val_mae: 7.2700
Epoch 48/100
12/12  0s 8ms/step - loss: 69.4617 - mae: 7.2791 - val_loss: 53.0911
- val_mae: 6.8860
Epoch 49/100
12/12  0s 8ms/step - loss: 75.0175 - mae: 7.2955 - val_loss: 47.6128
- val_mae: 6.4874
Epoch 50/100
12/12  0s 8ms/step - loss: 65.8916 - mae: 6.8053 - val_loss: 43.4744
- val_mae: 6.1715
Epoch 51/100
12/12  0s 7ms/step - loss: 55.6018 - mae: 6.2455 - val_loss: 38.4007
- val_mae: 5.7586
Epoch 52/100
12/12  0s 7ms/step - loss: 50.3141 - mae: 5.8234 - val_loss: 33.3311
- val_mae: 5.2968
Epoch 53/100
12/12  0s 8ms/step - loss: 51.6321 - mae: 5.9203 - val_loss: 28.9376
- val_mae: 4.9123
Epoch 54/100
12/12  0s 9ms/step - loss: 44.9687 - mae: 5.5295 - val_loss: 27.5366
- val_mae: 4.7504
Epoch 55/100
12/12  0s 6ms/step - loss: 43.7434 - mae: 5.2319 - val_loss: 24.5874
- val_mae: 4.4693
Epoch 56/100
12/12  0s 6ms/step - loss: 32.3503 - mae: 4.5870 - val_loss: 22.4457
- val_mae: 4.2873
Epoch 57/100
12/12  0s 14ms/step - loss: 33.0716 - mae: 4.5280 - val_loss: 20.7366
- val_mae: 4.0956
Epoch 58/100
12/12  0s 6ms/step - loss: 32.7472 - mae: 4.4010 - val_loss: 18.8280
- val_mae: 3.8955
Epoch 59/100
12/12  0s 6ms/step - loss: 34.0623 - mae: 4.6503 - val_loss: 16.4207
- val_mae: 3.6333
Epoch 60/100
12/12  0s 6ms/step - loss: 30.7667 - mae: 4.3748 - val_loss: 14.6530
- val_mae: 3.4259
Epoch 61/100
12/12  0s 6ms/step - loss: 25.4985 - mae: 3.9847 - val_loss: 13.3231
- val_mae: 3.2613
Epoch 62/100
12/12  0s 6ms/step - loss: 29.3105 - mae: 4.0378 - val_loss: 12.3079
- val_mae: 3.1393

Epoch 63/100
12/12  0s 6ms/step - loss: 26.8766 - mae: 4.0114 - val_loss: 11.4332
- val_mae: 2.9890
Epoch 64/100
12/12  0s 6ms/step - loss: 26.2433 - mae: 4.0240 - val_loss: 10.4955
- val_mae: 2.8531
Epoch 65/100
12/12  0s 6ms/step - loss: 26.1982 - mae: 4.1168 - val_loss: 9.1822 -
val_mae: 2.6835
Epoch 66/100
12/12  0s 6ms/step - loss: 21.3726 - mae: 3.5599 - val_loss: 8.3546 -
val_mae: 2.5269
Epoch 67/100
12/12  0s 6ms/step - loss: 22.0592 - mae: 3.6790 - val_loss: 7.8186 -
val_mae: 2.4710
Epoch 68/100
12/12  0s 6ms/step - loss: 32.0911 - mae: 4.1849 - val_loss: 7.0055 -
val_mae: 2.3117
Epoch 69/100
12/12  0s 6ms/step - loss: 25.6657 - mae: 3.9450 - val_loss: 7.3582 -
val_mae: 2.3417
Epoch 70/100
12/12  0s 6ms/step - loss: 18.9462 - mae: 3.2407 - val_loss: 7.4862 -
val_mae: 2.3181
Epoch 71/100
12/12  0s 6ms/step - loss: 24.2831 - mae: 3.6093 - val_loss: 8.0556 -
val_mae: 2.3470
Epoch 72/100
12/12  0s 6ms/step - loss: 23.0589 - mae: 3.6845 - val_loss: 7.0925 -
val_mae: 2.2701
Epoch 73/100
12/12  0s 6ms/step - loss: 28.0088 - mae: 3.9640 - val_loss: 7.0504 -
val_mae: 2.2538
Epoch 74/100
12/12  0s 6ms/step - loss: 22.3398 - mae: 3.4084 - val_loss: 6.9526 -
val_mae: 2.2348
Epoch 75/100
12/12  0s 6ms/step - loss: 20.3651 - mae: 3.3797 - val_loss: 6.3266 -
val_mae: 2.1192
Epoch 76/100
12/12  0s 6ms/step - loss: 17.3271 - mae: 3.2014 - val_loss: 6.4038 -
val_mae: 2.1018
Epoch 77/100
12/12  0s 6ms/step - loss: 22.5333 - mae: 3.6461 - val_loss: 6.6288 -
val_mae: 2.1347
Epoch 78/100
12/12  0s 6ms/step - loss: 18.3403 - mae: 3.1201 - val_loss: 6.0968 -
val_mae: 2.0295
Epoch 79/100
12/12  0s 6ms/step - loss: 21.7993 - mae: 3.6248 - val_loss: 5.6963 -
val_mae: 1.9233
Epoch 80/100
12/12  0s 6ms/step - loss: 23.0804 - mae: 3.5837 - val_loss: 5.4617 -
val_mae: 1.8826
Epoch 81/100
12/12  0s 14ms/step - loss: 24.5505 - mae: 3.8342 - val_loss: 5.6293
- val_mae: 1.9434
Epoch 82/100
12/12  0s 7ms/step - loss: 19.7901 - mae: 3.3639 - val_loss: 6.1507 -
val_mae: 2.0208
Epoch 83/100
12/12  0s 6ms/step - loss: 22.6920 - mae: 3.6214 - val_loss: 7.0608 -

```

val_mae: 2.1297
Epoch 84/100
12/12  0s 6ms/step - loss: 23.2871 - mae: 3.6297 - val_loss: 6.5907 -
val_mae: 2.0787
Epoch 85/100
12/12  0s 6ms/step - loss: 20.3778 - mae: 3.3319 - val_loss: 5.9528 -
val_mae: 2.0105
Epoch 86/100
12/12  0s 6ms/step - loss: 22.7217 - mae: 3.5694 - val_loss: 5.4382 -
val_mae: 1.8921
Epoch 87/100
12/12  0s 6ms/step - loss: 20.8176 - mae: 3.5130 - val_loss: 5.2851 -
val_mae: 1.8659
Epoch 88/100
12/12  0s 6ms/step - loss: 20.3383 - mae: 3.6779 - val_loss: 5.0191 -
val_mae: 1.8021
Epoch 89/100
12/12  0s 6ms/step - loss: 17.7908 - mae: 3.2143 - val_loss: 5.3917 -
val_mae: 1.8884
Epoch 90/100
12/12  0s 6ms/step - loss: 21.9207 - mae: 3.5945 - val_loss: 5.6272 -
val_mae: 1.8979
Epoch 91/100
12/12  0s 6ms/step - loss: 26.0262 - mae: 3.7425 - val_loss: 5.5692 -
val_mae: 1.8760
Epoch 92/100
12/12  0s 6ms/step - loss: 23.0110 - mae: 3.7211 - val_loss: 5.5682 -
val_mae: 1.8921
Epoch 93/100
12/12  0s 6ms/step - loss: 21.4021 - mae: 3.5792 - val_loss: 5.8731 -
val_mae: 1.9125
Epoch 94/100
12/12  0s 6ms/step - loss: 19.9595 - mae: 3.4217 - val_loss: 6.0088 -
val_mae: 1.9232
Epoch 95/100
12/12  0s 6ms/step - loss: 19.8390 - mae: 3.3956 - val_loss: 5.5730 -
val_mae: 1.8267
Epoch 96/100
12/12  0s 13ms/step - loss: 21.6693 - mae: 3.4570 - val_loss: 5.6258
- val_mae: 1.8457
Epoch 97/100
12/12  0s 6ms/step - loss: 17.5230 - mae: 3.1809 - val_loss: 5.3001 -
val_mae: 1.7907
Epoch 98/100
12/12  0s 6ms/step - loss: 19.8882 - mae: 3.4345 - val_loss: 5.2366 -
val_mae: 1.7872
Epoch 99/100
12/12  0s 6ms/step - loss: 16.5068 - mae: 3.0728 - val_loss: 5.4624 -
val_mae: 1.8145
Epoch 100/100
12/12  0s 6ms/step - loss: 20.5564 - mae: 3.3527 - val_loss: 5.5975 -
val_mae: 1.8336

```

```

In [9]: #Evaluate model
test_loss, test_mae = model.evaluate(X_test,y_test)
print(f"Test Loss (MSE): {test_loss:.2f}")
print(f"Test MAE: {test_mae:.2f}")

#print some vals
y_pred = model.predict(X_test)

```

```
np.set_printoptions(precision=2)
print(np.concatenate([y_pred[:10], np.array(y_test[:10]).reshape(-1, 1)], axis=1))
```

4/4 — 0s 6ms/step — loss: 16.6971 — mae: 2.7305

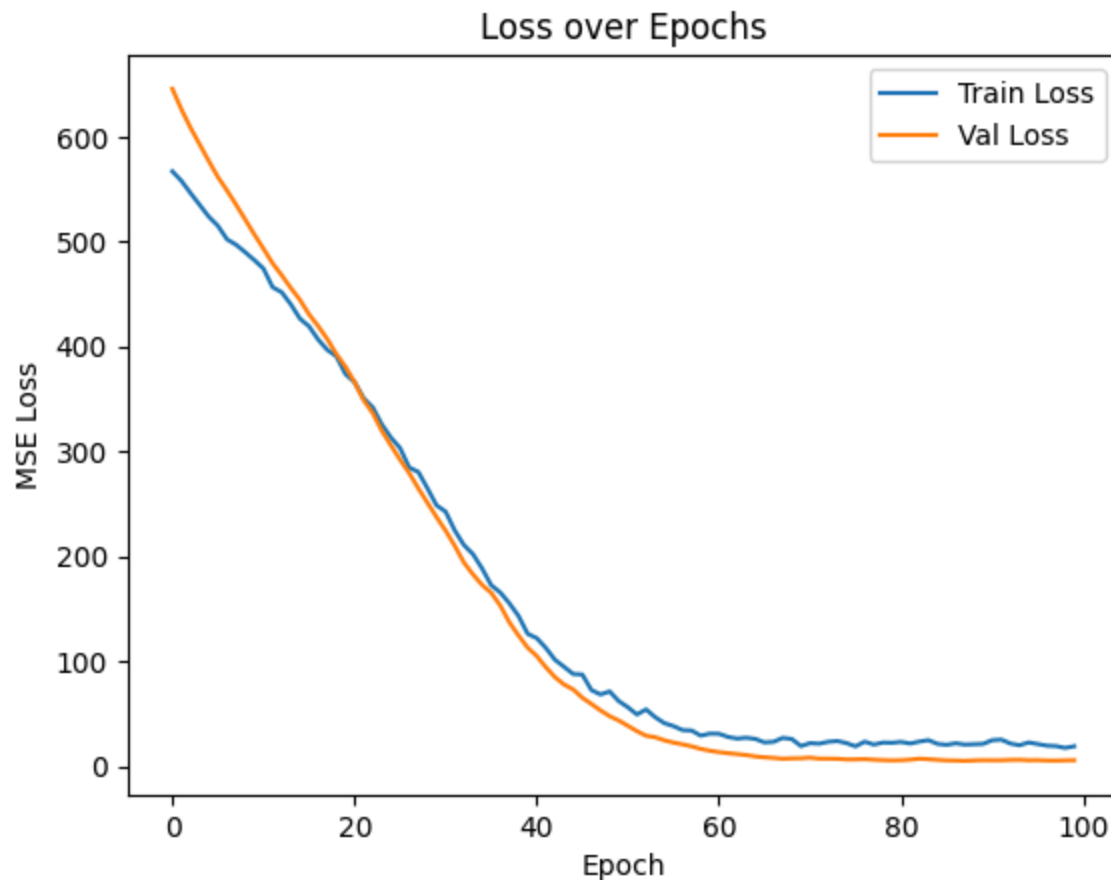
Test Loss (MSE): 16.38

Test MAE: 2.70

4/4 — 0s 22ms/step

```
[[23.42 22.2 ]
 [26.94 24.5 ]
 [20.84 21.1 ]
 [20.86 23.1 ]
 [16.19 17.4 ]
 [14.92 13.6 ]
 [15.11 13.2 ]
 [20.71 20.1 ]
 [20.67 29.8 ]
 [37.18 33.1 ]]
```

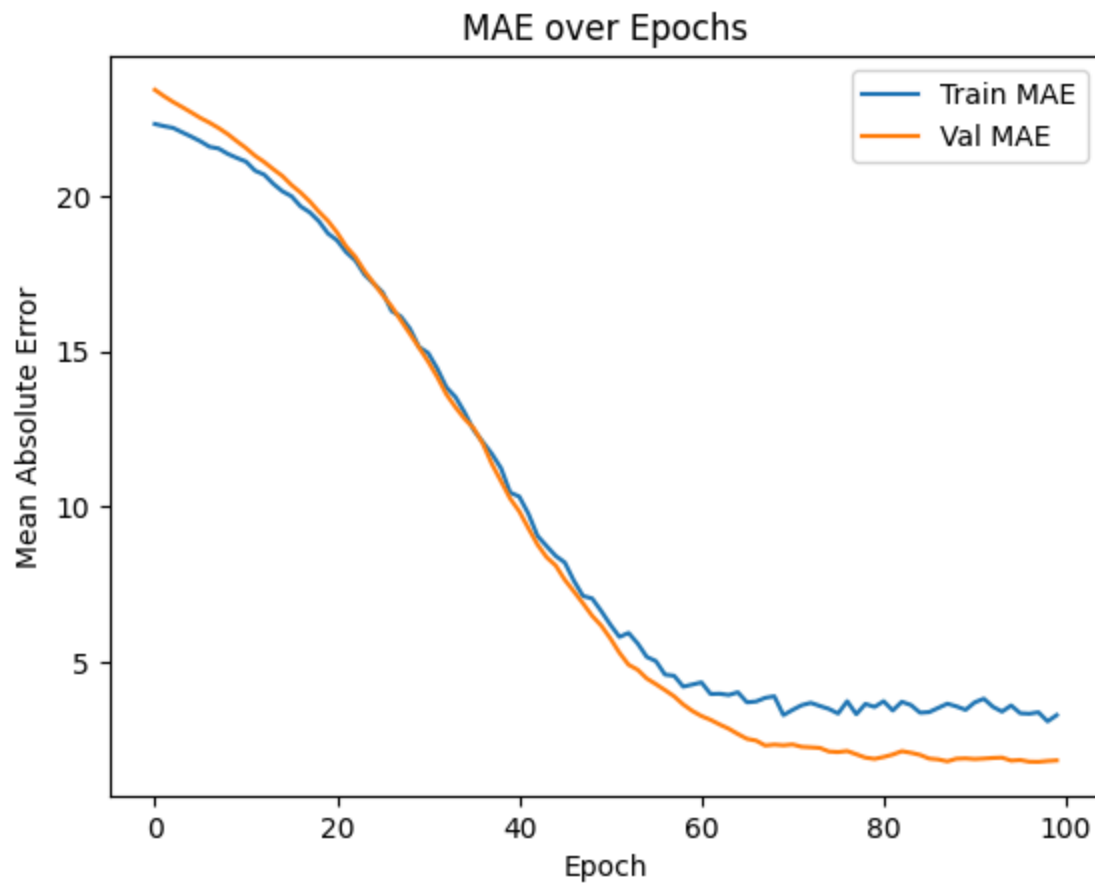
```
In [21]: #plot loss
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title('Loss over Epochs')
plt.xlabel('Epoch')
plt.ylabel('MSE Loss')
plt.legend()
plt.show()
```



```
In [23]: #plot MAE
plt.plot(history.history['mae'], label='Train MAE')
plt.plot(history.history.get('val_mae', []), label='Val MAE')
plt.title('MAE over Epochs')
plt.xlabel('Epoch')
plt.ylabel('Mean Absolute Error')
```



```
plt.legend()  
plt.show()
```



Conclusion

- Model trained well with no overfitting.
- Loss and MAE dropped consistently.
- ReLU and He init, batch norm, and dropout made training smooth.
- Final test MAE was 2.7, and predictions were close to actual values.