Assignment 1

Lisa Ewen, 0655603
Department of Computer Science
Lakehead University
955 Oliver Road, P7B 5E1
lewen@lakeheadu.ca

***Abstract*— This document is a report outlining a model implementation for predicting median housing prices. The model is a non-linear regression model built in Python using the PyTorch, NumPy, pandas, and Scikit-learn libraries. Utilizing two one-dimensional convolution layers, two ReLu layers, one max pooling layer, and two fully connected layers, the model is able to achieve accuracy.**

I. Introduction

In this report we will discuss a non-linear regression model built in python to predict median housing prices. This model is a neural network that utilizes one dimensional convolution layers implemented using Python, and libraries supported by Python. Section II features an overview of the libraries and their applications, while in section III an in-depth description of the model and its structure is provided. Section IV will discuss the simulation results.

II. Background

To build the model, the PyTorch, Scikit-learn, pandas, and NumPy Python libraries are utilized. Each of these libraries play a significant role to the model, as it would not be possible to approach a complex task without these libraries. Neural networks and data manipulation are a significant aspect of this model, and each of these libraries serve an important role.

PyTorch is based on the Torch library[2], which provides a set of algorithms used for deep learning that is no longer in development as of 2018[1]. Much like Torch, PyTorch is an open source library and is often used for computer vision[7] and natural language processing [6] tasks. An essential aspect of the library is the ability to use the Tensor class, which is similar to NumPy arrays, but have the ability to be operated on a CUDA-capable GPU [2]. This ability makes it especially useful for completing complex tasks requiring deep learning, much like natural language processing does. The ability to use the GPU alongside deep learning tasks makes the PyTorch library essential for this model.

The Scikit-learn library, often referred to as sklearn, provides many different methods for machine learning applications [5]. Most notably, this library features classification, regression, and clustering algorithms. It is also interoperable with NumPy and SciPy libraries, making support vector machines, random forests, k-means clustering, and many other applications, significantly more sophisticated. Another aspect of the library that increases ease of use for machine learning models is its wide array of pre-processing capabilities. In order to effectively pre-process and split the data, as well as test the model, it is important to have the capabilities of the Scikit-learn library.

As was mentioned previously, NumPy plays a significant role in other libraries used for machine learning. The NumPy library primarily adds support for large, multi-dimensional arrays and matrices [4]. Also provided with the library is a large array of functions capable of operating on these arrays and matrices. As Python was not often used for numerical computing when it was first designed, NumPy made these processes much simpler and more accessible to users. Because large amounts of data are being used to train this model, NumPy allows for much easier storage and calculations of this data before and after the model is trained.

Finally, the pandas library is intended for data analysis and manipulation [3]. Similar to NumPy, this library was created due to the fact that Python was not initially designed to perform the tasks that pandas can now provide. The initial goal was to achieve high performance and flexibility when attempting to perform quantitative analysis. Without the pandas library, it would be significantly more difficult to import and transform the data so that it is ready to train the model with.

III. Proposed Model

The model is trained using a modified version of the California Housing dataset. We can see from the first ten data points in tables I and II that the features are longitude, latitude, housing median age, total rooms, total bedrooms, population, households, median income, median house value, and ocean proximity. Due to the fact that ocean proximity is a string value, while all other features are floats, the ocean proximity feature will be excluded from the model training. We can also see via figure 1 the distribution of the datapoints, and how they appear relatively stable. The data is then split 70:30 for training and testing, which involves 114424 training values and 49040 testing values.

The definition of the model's structure is given by listing 1. In this model, ReLu is used primarily to avoid overfitting or underfitting of the model, but is also addressing the vanishing/exploding gradient problem as well. In order to test the model, the average L1Loss, R^2Score, and mean squared error is calculated during both training and testing. The model also utilizes a stochastic gradient descent optimizer, and is trained over 50 epochs with a batch size of 32.

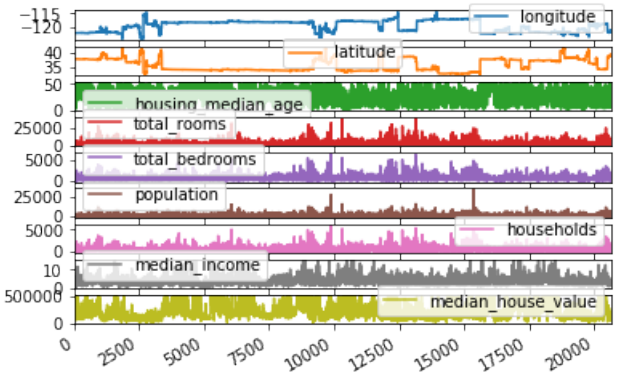

Fig. 1. Subplots of the 9 features being considered by the model

TABLE I

THE FIRST 10 DATA POINTS PRECEDING 5 FEATURES.

Features 1-5				
(1)[a]	(2)[b]	(3)[c]	(4)[d]	(5)[e]
-122.23	37.88	41.0	880.0	129.0
-122.22	37.86	21.0	7099.0	1106.0
-122.24	37.85	52.0	1467.0	190.0
-122.25	37.85	52.0	1274.0	235.0
-122.25	37.85	52.0	1627.0	280.0
-122.25	37.85	52.0	919.0	213.0
-122.25	37.84	52.0	2535.0	489.0
-122.25	37.84	52.0	3104.0	687.0
-122.26	37.84	42.0	2555.0	665.0
-122.25	37.84	52.0	2549.0	707.0

[a] 1: Longitude, [b] 2: Latitude, [c] 3: Housing Median Age, [d] (4): Total Rooms, [e] (5): Total Bedrooms

TABLE II

THE FIRST 10 DATA POINTS' REMAINING FEATURES.

Features 6-9			
(1)[a]	(2)[b]	(3)[c]	(4)[d]
322.0	126.0	8.3252	452600.0
2401.0	1138.0	8.3014	358500.0
496.0	177.0	7.2574	352100.0
558.0	219.0	5.6431	341300.0
565.0	259.0	3.8462	342200.0
413.0	193.0	4.0368	269700.0
1094.0	514.0	3.6591	299200.0
1157.0	647.0	3.1200	241400.0
1206.0	595.0	2.0804	226700.0
1551.0	714.0	3.6912	261100.0

[a] 1: Population, [b] 2: Households, [c] 3: Median Income, [d] (4): Median House Value

Listing 1. Model definition

```
class CnnRegressor(torch.nn.Module):
  def __init__(self, batch_size, inputs, outputs):
    #layer definitions
    super(CnnRegressor, self).__init__()
    self.batch_size = batch_size
    self.inputs = inputs
    self.outputs = outputs
    self.input = Conv1d(inputs, batch_size, 1)
    self.max_pooling = MaxPool1d(1)
    self.conv = Conv1d(batch_size, 64, 1)
    self.flatten = Flatten()
    self.linear = Linear(64, batch_size)
    self.output = Linear(batch_size, outputs)

  def feed(self, input):
  #layer sequence for feedforward computations
    input = input.reshape((self.batch_size, self.
        inputs, 1))
    output = relu(self.input(input))
    output = self.max_pooling(output)
    output = relu(self.conv(output))
    output = self.flatten(output)
    output = self.linear(output)
    output = relu(self.output(output))

    return output
```

IV. Experimental Analysis

In order to track progress reliably, the seed of the training and testing split is set to 2003. When run with the outlined framework, the outputs obtained are given by tables III and IV.

TABLE III

THE RESULTS OF SIMULATION TESTS.

	Simulation Results	
Simulation Time	R^2Score	L1Loss
56.44s	0.31	73802.05
56.21s	0.34	71815.06
56.71s	0.36	70093.16

TABLE IV

THE RESULTS OF SIMULATION TESTS.

Simulation Results
MSE
9139782200.25
8753749637.16
8558615209.32

V. Conclusion

As we can see by the results given in the previous section, this non-linear regression model is capable of predicting the mean value price of a house with % testing accuracy. In order to achieve this, the PyTorch, NumPy, Scikit-learn, and pandas library were heavily relied upon. While we can see that the overall results are not particularly fantastic, the model successfully runs and makes predictions as intended.

References

[1] P.W.D. Charles. *Torch*. `https://github.com/torch/torch7`. 2002.

[2] Nikhil Ketkar. "Introduction to pytorch". In: *Deep learning with python*. Springer, 2017, pp. 195–208.

[3] Wes McKinney et al. "pandas: a foundational Python library for data analysis and statistics". In: *Python for High Performance and Scientific Computing* 14.9 (2011).

[4] Travis E Oliphant. “Python for scientific computing”. In: *Computing in Science & Engineering* 9.3 (2007), pp. 10–20.

[5] Fabian Pedregosa et al. “Scikit-learn: Machine learning in Python”. In: *Journal of machine learning research* 12.Oct (2011), pp. 2825–2830.

[6] Delip Rao and Brian McMahan. *Natural language processing with PyTorch: build intelligent language applications using deep learning*. " O’Reilly Media, Inc.", 2019.

[7] Ivan Vasilev. *Python deep learning: exploring deep learning techniques and neural network architectures with PyTorch, Keras, and TensorFlow*. Packt Publishing, 2019.