King Saud University

College of Computer and Information Sciences

Computer Science Department

"AI experiment"

CSC 361

Group14

Prepared by:

Mohammed Aljalajil 442101136
Abdulaziz Alshaalan 441170125
Abdulelah Alhadyah 441102210

Abdulaziz Alhajri 441102528

Abdullah Ashalan 439101546

Instructed by:

Dr.Ahmad Alshibli

Spring 2024

I.Abstract

In this assignment, we will propose a deep learning experiment that make a model, that can identify a hidden mathematical pattern between two integer datasets using TensorFlow's Keras library. A normal or even a smart human might face some difficulties and challenges or taking a lot of time finding some hard or even easy mathematical pattern between numbers, but we can use deep learning models to do that for us and it is easy to implement, and efficient if we applied it the right way. We are going to make a neural network using Keras by just writing a few codes to make the layers of this neural network, and we can also identify the number of training iterations, it is named epochs in Keras, and identify the optimizer and the loss function, neural network will take a value of X and a corresponding value of Y and then try to guess the pattern between them over and over again making in the beginning wrong guesses and then enhance those guesses by using the optimizer and the loss function, until it will provide the actual pattern between them, of course after doing enough iterations and training on enough data. After the model accomplish finding the actual pattern by specifying the right weights of the neural network, we can give it any number as an input, and it will take that number, and apply the pattern the model has detected, and it will provide a new number after applying that pattern as an output. In conclusion we are going to use an API tool (Keras) provided by Google that can make us program a deep learning model with just a few lines of code, we want to make a model that can specify a mathematical pattern by giving it two datasets, it will do that by learning by guessing the pattern in the beginning until it will find the right pattern after enough training.

II. Introduction

Finding mathematical patterns can be very difficult and it can take a lot of time if the mathematical pattern was complicated, to preserve our valuable time and effort we can use a deep learning model to do that for us [1], we can build this model using TensorFlow's Keras library [2], this library is a widely used deep-learning packages to build deep neural networks and optimization models [2], Keras is very effective in building deep learning models by an easily implemented code [4]. At first, we ~~implemented~~will implement some libraries ~~like~~, one is TensorFlow to import Keras [5] to build the model, NumPy to deal with integer arrays in a better way~~,~~ that's it for libraries. And then used sequential which allows easy creation of a linear stack of layers [3] to make neural network layers, and then use compile this to specify how the training is going to be made by identifying the optimizer and the loss function [4], then provided a quite enough data for X and Y values, then we did the training by using fit and giving it X and Y values [3], and the number of iterations (epochs) that is going to be made [6]. In the end after the model finished training and it is supposed to accomplish a quite well results by bringing the loss to nearly to 0.4 after iterating for 100 epochs, then we will give the model a new number for example number 10 as an X value as an input to "predict" this is the function that will predict the output from the input value and after applying the pattern it detected before [3], it is going to generate a value near to 18 but not exactly 18, not just because of the loss value it accomplished because it was quite will but because the model is not 100% sure of the pattern it detected even if the loss was lower than that it will never give the value 18, to get the specific value we need to do some extra work that is out of this course.

Related Works:

- Davies *et al.*, “Advancing mathematics by guiding human intuition with AI,” *Nature*, vol. 600, no. 7887, pp. 70–74, Dec. 2021, doi: 10.1038/s41586-021-04086-x.

- I. Nica, C. Delcea, and N. Chiriță, “Mathematical Patterns in Fuzzy Logic and Artificial Intelligence for Financial Analysis: A Bibliometric Study,” *Mathematics*, vol. 12, no. 5, p. 782, Mar. 2024, doi: 10.3390/math12050782.

- P. Sokkhey and T. Okazaki, “Comparative Study of Prediction Models on High School Student Performance in Mathematics,” in *2019 34th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*, IEEE, Jun. 2019, pp. 1–4. doi: 10.1109/ITC-CSCC.2019.8793331.

Applications:

- Voice recognition.

- Image recognition.

- Natural Language Processing

- Sentiment analysis.

- Deep neural network.

The code:

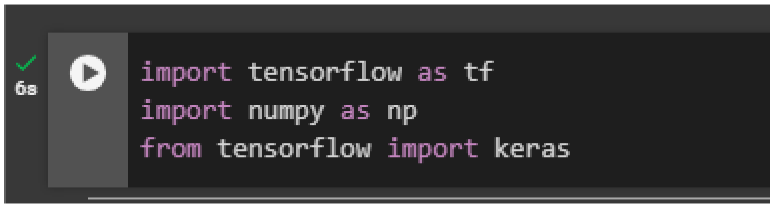
```
import tensorflow as tf
import numpy as np
from tensorflow import keras

model = tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])

model.compile(optimizer='sgd', loss='mean_squared_error')

x = np.array([-1.0,  0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
y = np.array([-4.0, -2.0, 0.0, 2.0, 4.0, 6.0], dtype=float)

model.fit(x, y, epochs=100)

print(model.predict([10.0]))
```

Experiment:

In our experiment we will be providing two datasets to the model in order to make it learn the rule or the relation between the two datasets we will be using google collab and TensorFlow and NumPy libraries, first of all we will import our libraries in our case it's TensorFlow and NumPy and we will be using Keras from TensorFlow it used to build a neural network as a sequence of layers we will import it from TensorFlow.

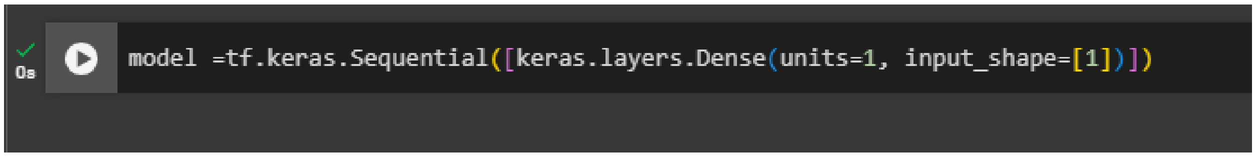
Our rule will be (2X-2)

Step 1: importing the libraries.



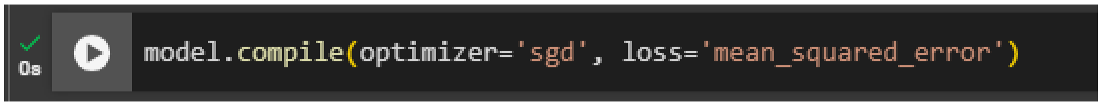
```
import tensorflow as tf
import numpy as np
from tensorflow import keras
```

Step 2: In this step we will build this model using Keras sequential class to define the network as sequence of layers.



```
model =tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
```

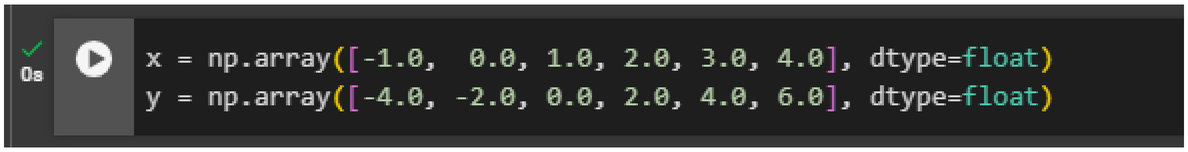
Step 3: Now we will be compiling the neural network, we will have to specify new functions, loss function and optimizer function to make another guess based on how the loss function went, optimizer function will try to minimize the loss function by giving improved guesses.



```
model.compile(optimizer='sgd', loss='mean_squared_error')
```

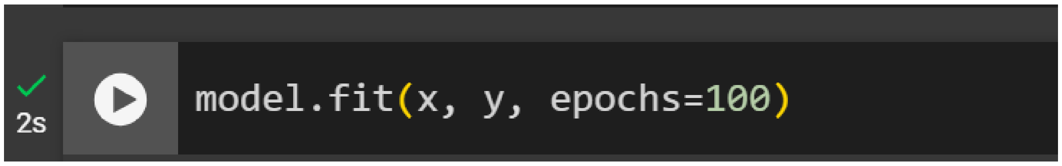
Step 4: Using our datasets we will provide 2 datasets X and Y that the relation between them is

Y = (2X-2) and make the model guess the relation and learn it between X and Y.



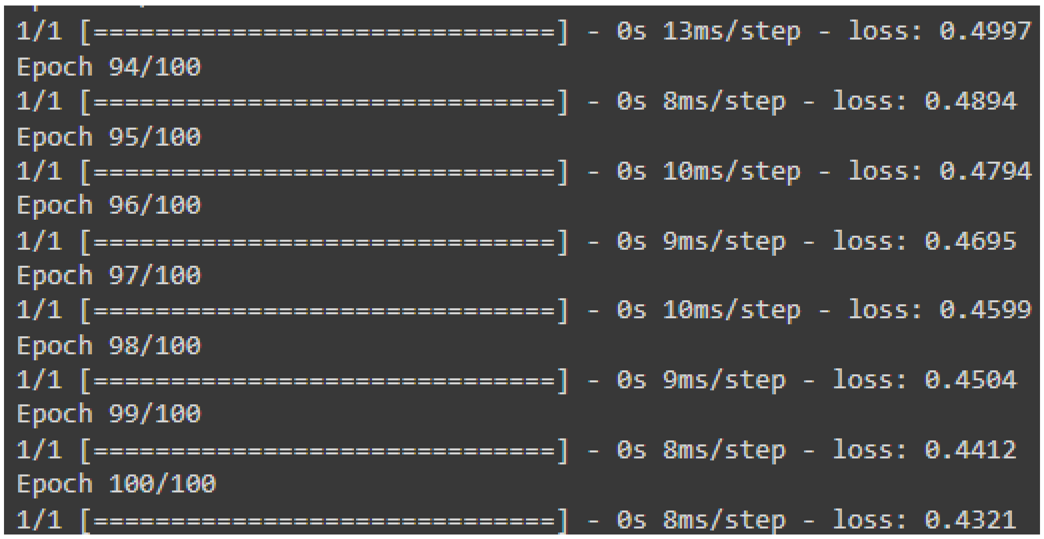
```
x = np.array([-1.0,  0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
y = np.array([-4.0, -2.0, 0.0, 2.0, 4.0, 6.0], dtype=float)
```

Step 5: Now we will train our model with **model.fit** as we will see the loss function of our guess measure weather is it good or bad by the value and use optimizer to do improved guesses, **epochs** is the number of times the we will make our model train.



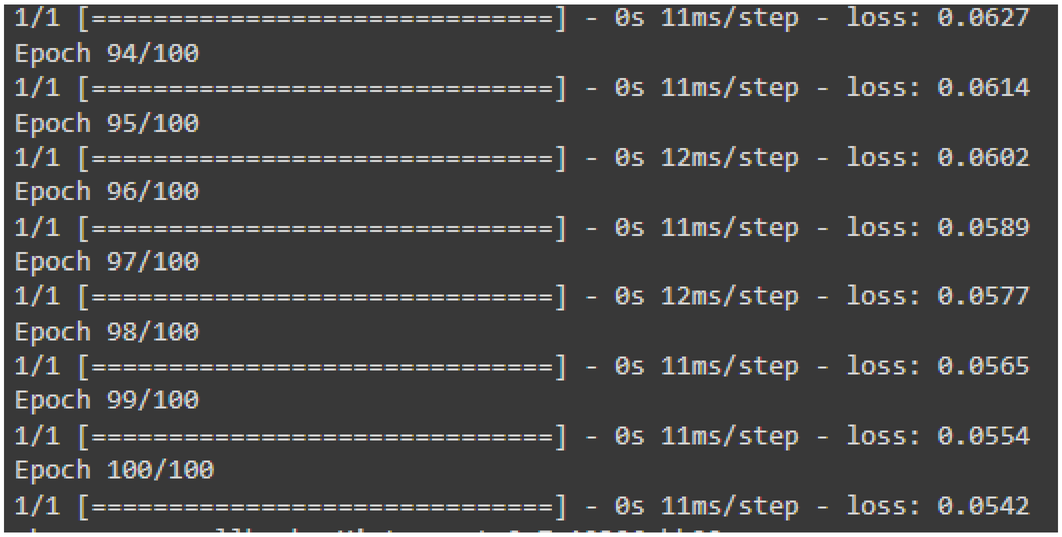
```
model.fit(x, y, epochs=100)
```

NOTING: high epochs doesn't necessary means better loss function, but we can make very high number of epochs and use callback function in order to stop the model.fit in desired loss function value.



```
1/1 [==============================] - 0s 13ms/step - loss: 0.4997
Epoch 94/100
1/1 [==============================] - 0s 8ms/step - loss: 0.4894
Epoch 95/100
1/1 [==============================] - 0s 10ms/step - loss: 0.4794
Epoch 96/100
1/1 [==============================] - 0s 9ms/step - loss: 0.4695
Epoch 97/100
1/1 [==============================] - 0s 10ms/step - loss: 0.4599
Epoch 98/100
1/1 [==============================] - 0s 9ms/step - loss: 0.4504
Epoch 99/100
1/1 [==============================] - 0s 8ms/step - loss: 0.4412
Epoch 100/100
1/1 [==============================] - 0s 8ms/step - loss: 0.4321
```

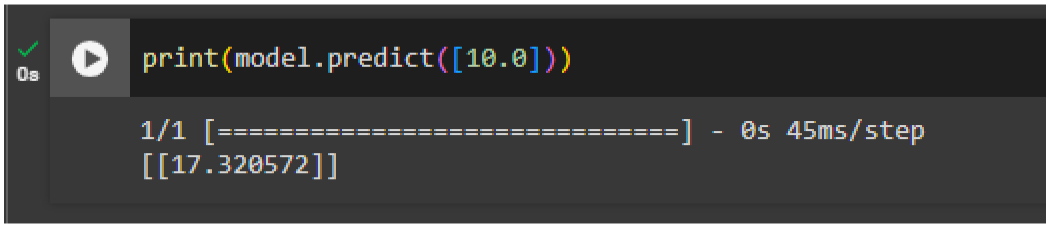
As you can see we trained our model using 100 epochs our loss function is 0.4321 it is accepted value but what if we retrained or fit our model again using 100 epochs will it give the same loss function?



```
1/1 [==============================] - 0s 11ms/step - loss: 0.0627
Epoch 94/100
1/1 [==============================] - 0s 11ms/step - loss: 0.0614
Epoch 95/100
1/1 [==============================] - 0s 12ms/step - loss: 0.0602
Epoch 96/100
1/1 [==============================] - 0s 11ms/step - loss: 0.0589
Epoch 97/100
1/1 [==============================] - 0s 12ms/step - loss: 0.0577
Epoch 98/100
1/1 [==============================] - 0s 11ms/step - loss: 0.0565
Epoch 99/100
1/1 [==============================] - 0s 11ms/step - loss: 0.0554
Epoch 100/100
1/1 [==============================] - 0s 11ms/step - loss: 0.0542
```

As you can see it is different than our first fit.

Step 6: we will be making our model to predict Y as X is provided to the model and the relationship between them is (2X-2) and we used model.fit to make the model learn it, lets give the model a value like 10 we will give it as 10.0 for float but will it give us (2(10)-2) as 18? [5]



```
print(model.predict([10.0]))
```

```
1/1 [==============================] - 0s 45ms/step
[[17.320572]]
```

The model predicted 17.3 not 18, very close that is because we had loss function of 0.0542.

Thank you very much as from here we finished conducting our experiment.

Comparison:

Features	TensorFlow	PyTorch	CNTK
Developed By	Google	Facebook	Microsoft
Open Source	✓	✓	✓
Community	Largest community.	Growing day-by-day.	small community.
Performance	Provides optimized-performance.	similar to TensorFlow.	renowned for being quick and scalable.
Strengths	1-Comprehensive ecosystem. 2-Powerful visualization tools.	1-Usability. 2- Dynamic diagrams. 3- Strong support.	1-Great performance. 2- Scalability. 3-Memory efficiency.
Weaknesses	1-Slow learning curve. 2-Can be complicated for simple tasks.	1-Visualization tools is not as Resourceful as TensorFlow. 2-Much effort for deployment.	1-Much smaller community. 2-Less intuitive than PyTorch. 3-Documentation can be challenging.
Developments	Focus on enhancing ease of use, efficiency, and mobile deployment options. TensorFlow 2.0 added aggressive	Growing popularity, and seeing greater application in research and industry. Focused on improving efficiency and expanding the ecological system.	Development has halted as Microsoft redirects its focus to other plans. However, it is still an options for some use

	execution and a more simplified API.		circumstances like (HPC) Scenarios.

III.Conclusion

To summarize, we got two datasets that had a pattern between them, X and Y. We trained the model to predict that pattern using Keras. Of course, AI is built on statistics, which means nothing is definite. We had a bit of loss, which is ~~numerized~~numerated as a loss function. The loss function manifested when it didn't predict Y from X perfectly. We did face issues like expected; one of them revolves around the time and effort taken to find the pattern between numbers, also we faced struggles choosing the right AI tool. At first, we used Pytorch and started to write the code after that we realized that it's not the optimal choice,

So, we changed it to TensorFlow which made it so much better and easier to write the code. Keras sometimes would just run in circles or make no significant progress. We had to fine-tune the code and epochs to get the results in a reasonable timeframe. We had fun using Keras and TensorFlow; they're powerful and make life easier for everyone in the AI field. There's no doubt that they will be one of the key factors for growth in AI and will allow everyone, no matter their depth or skill, to try their luck in it. Fresh minds and talent are always a plus in every growing field.

References

[1] R. O. Duda, P. E. Hart, and D. G. Stork, “Pattern Classification and Scene Analysis 2nd ed. Part 1: Pattern Classification,” 1995.

[2] Developers, T., “TensorFlow”, <i>Zenodo</i>, Zenodo, 2021.

[3] E. Haghighat and R. Juanes, “SciANN: A Keras/TensorFlow wrapper for scientific computations and physics-informed deep learning using artificial neural networks,” Comput Methods Appl Mech Eng, vol. 373, p. 113552, Jan. 2021, doi: 10.1016/j.cma.2020.113552.

[4] J. Moolayil, Learn Keras for Deep Neural Networks. Berkeley, CA: Apress, 2019. doi: 10.1007/978-1-4842-4240-7.

[5] Ketkar, N. (1970) Introduction to keras, SpringerLink.

[6] Sinha, S., Singh, T.N., Singh, V.K. et al. Epoch determination for neural network by self-organized map (SOM). Comput Geosci 14, 199–206 (2010).