LOVELY PROFESSIONAL UNIVERSITY

Project Report

Classifying Corn Image Data Set

Course Code: INT248

Course Title: Advanced Machine Learning

Submitted By:

Ch.Hemanthsai

11712304

Section: KM040

Roll.No:20

Submitted To:

Sanjay Kumar Singh

Professor at Lovely Professional University

ACKNOWLEDGMENT

Place: - Lovely Professional University

Date: - 31st October 2020

I would like to thank Sanjay Kumar Singh for assigning us with this project. Through the project, I can grasp more technical and have a hands-on practical experience with python and some machine learning algorithms. Through it, I can learn how a project is created and how necessary and crucial technical knowledge is. I am grateful to the faculty that has provided me with the necessary guidelines.

Reg.No: 11712304

Name: Ch.Hemanth sai

DECLARATION

Place: - Lovely Professional University

Date: - 31st October 2020

I hereby declare that this report has been written by me. No part of the report is copied from the other sources. All information included from other sources has been duly acknowledged. I aver that if any part of the report is found to be copied, I will take full responsibility for it.

Name: Ch.Hemanth sai

Reg No: 11712304.

TABLE OF CONTENTS

INTRODUCTION

Name: Classifying Corn Image Data Set.
Dataset: It consists of four classes
Technique: Multilayer sequential model

Collection of Database/necessary requirements for project:

Database of all the corn image dataset from the Kaggle platform. Link for the dataset is https://www.kaRu1e.com/nahomy/corn1mage.
It will be attached with this report in the zip folder.
Anaconda Application and TensorFlowV.2.2.0 must be installed in your System.

ABOUT DATASET

1. *Cercospora* - Cercospora is a genus of ascomycete fungi. Most species have no known sexual stage, and when the sexual stage is identified, it is in the genus Mycosphaerella. Most species of this genus cause plant diseases, and form leaf spots. It is a relatively well-studied genus of fungus, but there are countless species not yet described, and there is still much to learn about the best-known of the species.

2. *Common rust* - Common rust is caused by the fungus *Puccfnia sorghi* and occurs every growing season. It is seldom a concern in hybrid com. Rust pustules usually first appear in late June. Early symptoms of common rust are chlorotic flecks on the leaf surface. These soon develop into powdery, brick-red pustules as the spores break through the leaf surface. Pustules are oval or elongated, about 1/8 inch long, and scattered sparsely or clustered together. The leaf tissue around the pustules may become yellow or die, leaving lesions of dead tissue. The lesions sometimes form a band across the leaf and entire leaves will die if severely infected. As the pustules age, the red spores turn black, so the pustules appear black, and continue to erupt through the leaf surface. Husks, leaf sheaths, and stalks also may be infected.

3. *Healthy* — in this corn is best to others

4. *Northern_b1ight* - Northern leaf blight (NLB) can cause severe yield loss in maize; however, scouting large areas to accurately diagnose the disease is time consuming and difficult. We demonstrate a system capable of automatically identifying NLB lesions in field-acquired images of maize plants with high reliability. This approach uses a computational pipeline of convolution neural networks (CNNs) that addresses the challenges of limited data and the myriad irregularities that appear in images of field-grown plants. Several CNNs were trained to classify small regions of images as containing NLB lesions or not; their predictions were combined into separate heat maps, then fed into a final CNN trained to classify the entire image as containing diseased plants or not. The system achieved 96.7% accuracy on test set images not used in training. We suggest that such systems mounted on aerial- or ground-based vehicles can help in automated high-throughput plant phenoltyping, Precision breeding for disease resistance and reduced pesticide use through targeted application across a variety of plant and disease categories.

IMPLEMENTATION OF CODE

- ❖ Importing Images and Augmentation
- ❖ Choosing appropriate model Optimizer
- ❖ Training (Retraining for optimization)
- ❖ Evaluating Accuracy with visualization

1. Importing Images and Augmentation:

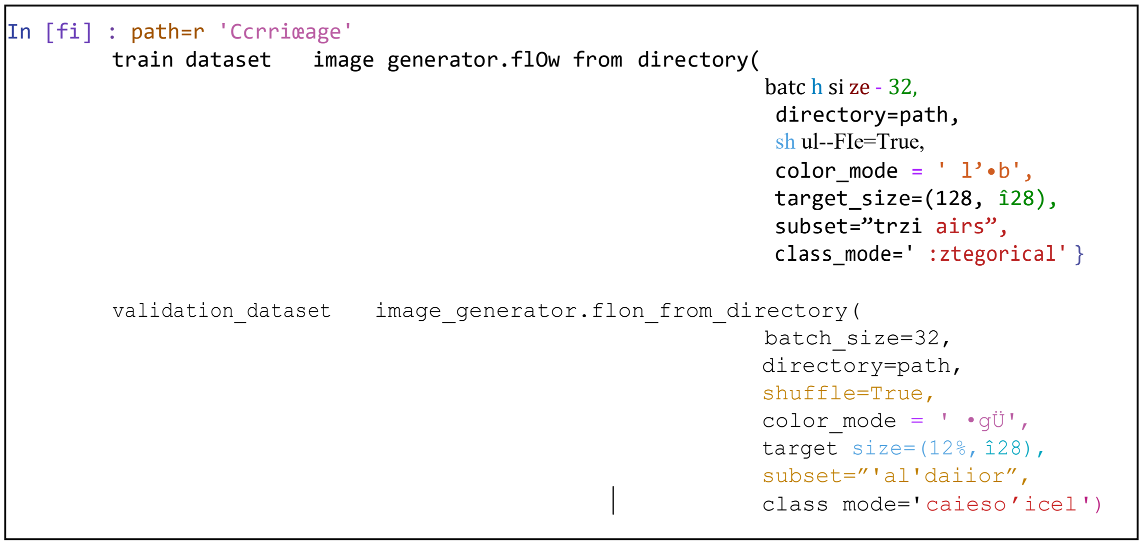
- S The total dataset contains 3839 images belonging to 4 classes.
- S And out of those 3839 images we use 3073 images for training and remaining 766 for validation purpose, you can see them in Fig 1.
- S The images are of 256x256 pixel size. But we are import them as the 128x128 pixels, you can see them in Fig 2.
- S We are rescaling the image from 1 to 255, you can see them in Fig 2.
- S We are putting a random application of shearing of with 0.2, you can see them in Fig 2.
- S We are putting a random zoom range with 0.2, you can see them in Fig 2.
- S We are putting a random horizontal flipping of images; you can see them in Fig2.
- S I am using flow from directory to load the images, you can see them in Fig 3.

```
Found 3073 images belonging to 4 classes.
Found 766 in ages belonging to 4 c las ses .
```

Fig 1

```
In [3]:  image generator=tf. keras.preprocessing.image.ImageDataGenerator(
             rescale = 1./255,
             shear_range = 0.2,   rcru r coe'icut i» c  sñecrl. q
             zoom_range = 0.2,
             horizontal_flip = True,
             validation split=0.2
         )
```

Fig 2



```
In [fi] : path=r 'Ccrriœage'
       train dataset    image generator.flOw from directory(
                                                             batc h si ze - 32,
                                                              directory=path,
                                                              sh ul--FIe=True,
                                                              color_mode = ' l’•b',
                                                              target_size=(128, î28),
                                                              subset=”trzi airs”,
                                                              class_mode=' :ztegorical' }

       validation_dataset    image_generator.flon_from_directory(
                                                             batch_size=32,
                                                             directory=path,
                                                             shuffle=True,
                                                             color_mode = ' •gÜ',
                                                             target size=(12%, î28),
                                                             subset=”'al'daiior”,
                                                             class mode='caieso’icel')
```

Fig 3

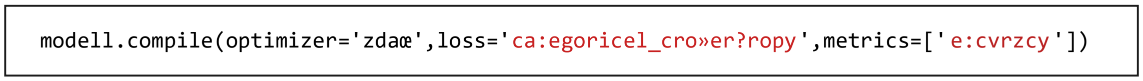
2.Choosing appropriate optimizer for Sequential model:

H As we have to perform classification, I am choosing three famous optimizers like Adam, SGD and RMSprop.

R The implementation of Adam optimizer can be seen in fig4.

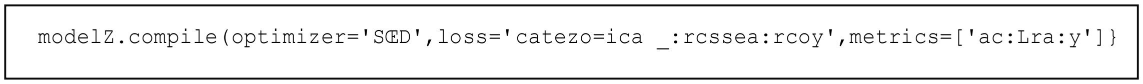
R The implementation of Adam optimizer can be seen in fig 5

H The implementation of Adam optimizer can be seen in fig 6



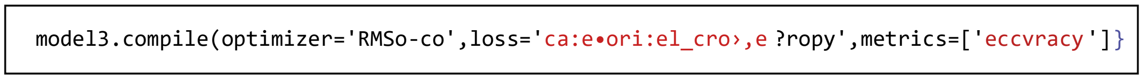
```
modell.compile(optimizer='zdaœ',loss='ca:egoricel_cro»er?ropy',metrics=['e:cvrzcy'])
```

Fig 4



```
modelZ.compile(optimizer='SŒD',loss='catezo=ica _:rcssea:rcoy',metrics=['ac:Lra:y']}
```

Fig 5



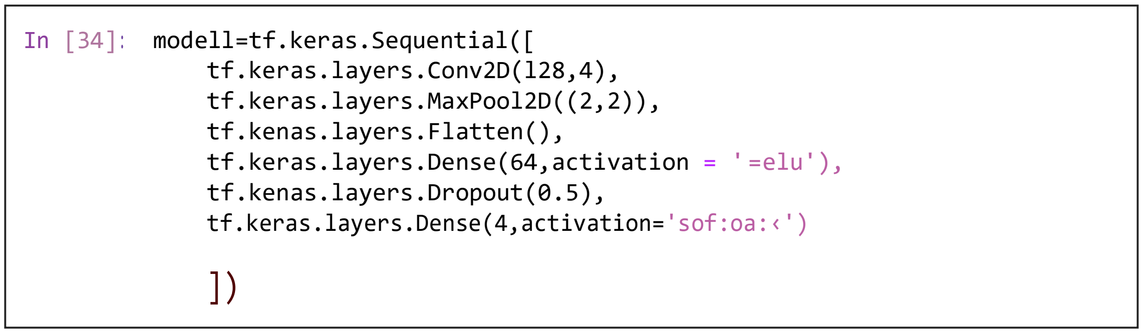
```
model3.compile(optimizer='RMSo-co',loss='ca:e•ori:el_cro›,e ?ropy',metrics=['eccvracy']}
```

Fig 6

2 Training (Retraining for optimization):

S Sequential Model is built as shown in the fig 7

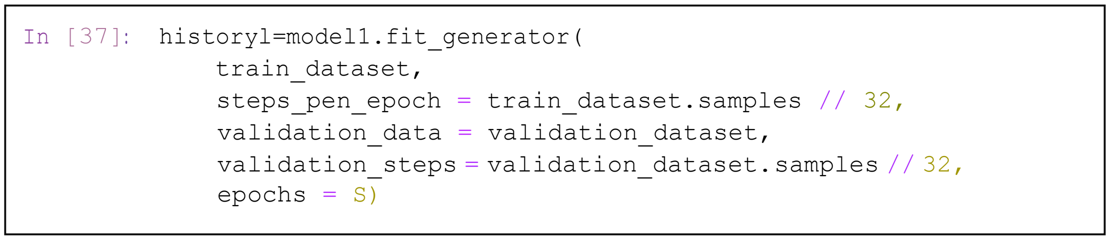
S Trained every model on Five Epochs with categorical crossentropy loss function as shown in fig 8



```
In [34]:  modell=tf.keras.Sequential([
              tf.keras.layers.Conv2D(128,4),
              tf.keras.layers.MaxPool2D((2,2)),
              tf.kenas.layers.Flatten(),
              tf.keras.layers.Dense(64,activation = '=elu'),
              tf.kenas.layers.Dropout(0.5),
              tf.keras.layers.Dense(4,activation='sof:oa:‹')

              ])
```

Fig 7



```
In [37]:  historyl=model1.fit_generator(
              train_dataset,
              steps_pen_epoch = train_dataset.samples // 32,
              validation_data = validation_dataset,
              validation_steps = validation_dataset.samples // 32,
              epochs = S)
```

Fig 8

3 Evaluating Accuracy with visualization:

S The fig 9 shows the graph of loss and accuracy for the Adam Optimizer.
S The fig 10 shows the graph of loss and accuracy for the SGD Optimizer.
S The fig 11 shows the graph of loss and accuracy for the RMSprop Optimizer.

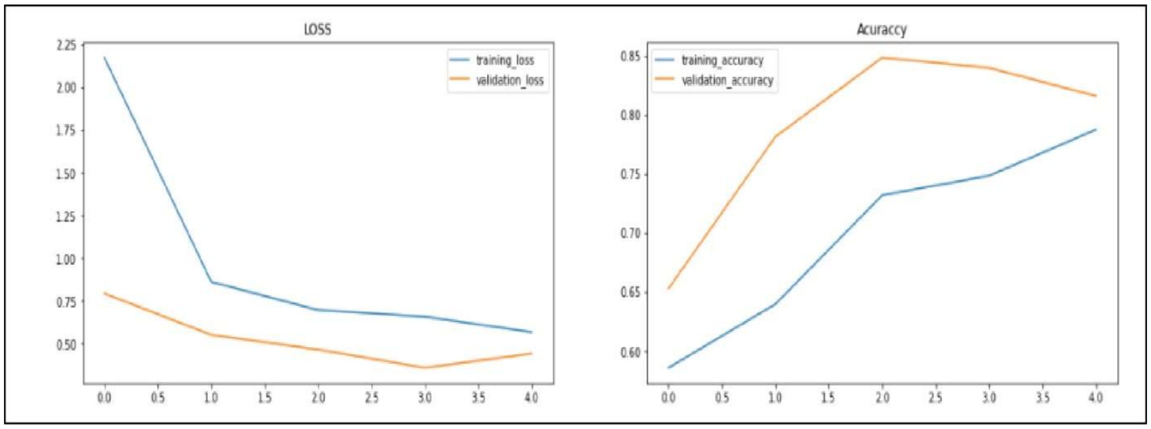


Fig 9

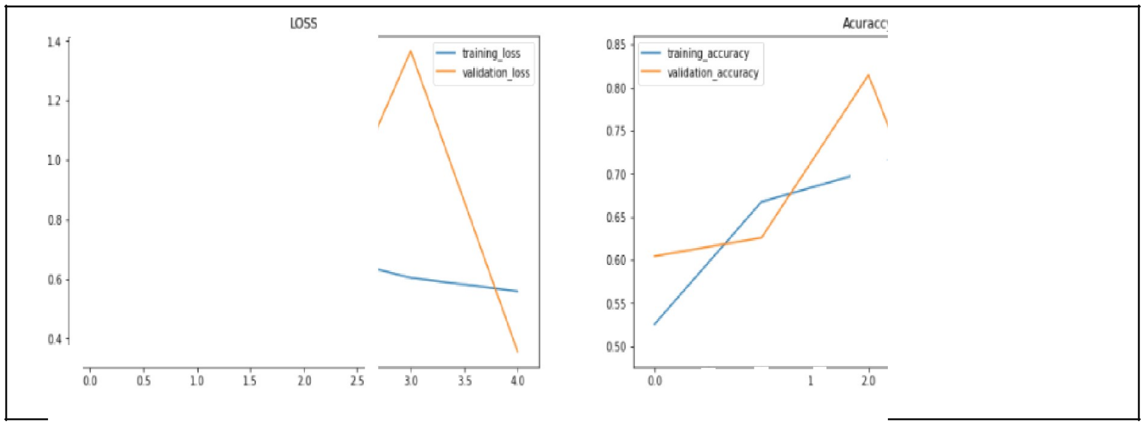


Fig 10

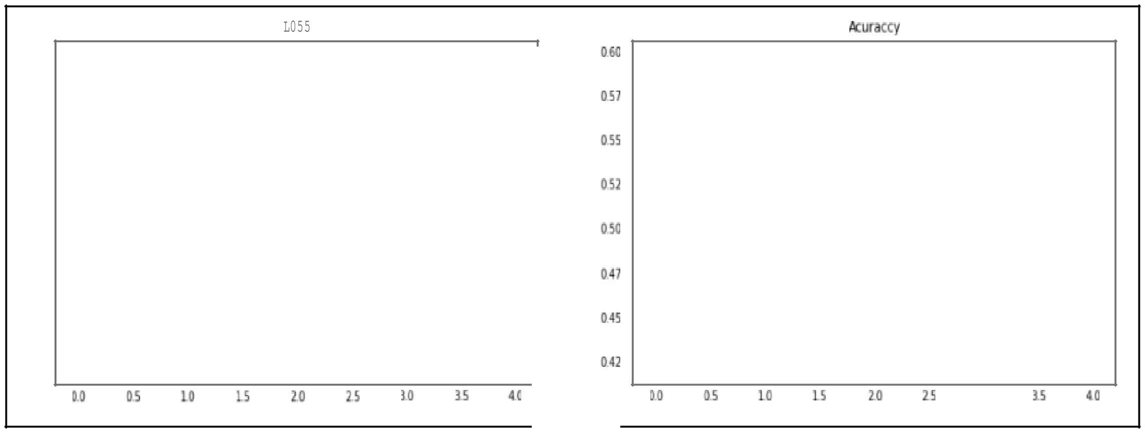


Fig 11

TECHNOLOGIES USED

• Python

- Python is an interpreter, object-oriented, high-level programming language with dynamic semantics. It's high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components.

• Machine learning

- Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning.

• Classification

- Classification is one of the most widely used techniques in machine learning, with a broad array of applications, including sentiment analysis, ad targeting, spam detection, risk assessment, medical diagnosis and image classification. The core goal of classification is to predict a category or class Y from some inputs X.

• Pandas

- Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

• Numpy

- NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

• Matplotlib

- Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+.

• SKlearn

- Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines and others.

• TensorFIow

- TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

• Seaborn

- Seaborn is a Python data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

• Keras

- Keras is an open-source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks; it focuses on being user-friendly, modular, and extensible.

REFERENCES

I went through different websites and textbooks to learn those concepts that will be used in making this project.

Some of the websites are:

- https://www.zoozle.com/
- https://www.python.ore/
- https://numpy.orm/
- https://pandas.pydata.org/
- https://matp1ot1ib.org/
- https://scikit-learn.orm/
- https://www.tensorflow.org/
- https://keras.io/

CODE

```
# Suppressing Warnings
import warnings
warnings.filterwarnings('ignore')
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as pit

dir_name=path=r'Cornimage'
sample=tf.keras.preprocessing.image_dataset_from directory(dir_name,seed=123,image size=(1
80,180),batch_size=32)

plt.figure(figsize=(10,10))
for image,label in sample.take(9):
    for i in range(9):
        plt.subplot(3,3,i+1)
        plt.imshow(image[i].numpy().astype('uint8'))
        pit.axis('off)
        plt.title(class_names[label[i]])


image_generator=tf.keras.preprocessing.image.ImageDataGenerator(
    rescale = 1./255,
    shear_range = 0.2, # random application of shearing
    zoom_range = 0.2,
    horizontal_flip = True,
     validation_split=0.25
)

path=r'Cornimage'
train_dataset = image_generator.flow_from_directory(
                                            batch size=32,
                                             directory=path,
                                             shuffle=True,
                                             color_mode = 'rgb',
                                             target_size=(128, 128),
                                             subset="training",
                                             class  mode='categorical')

validation_dataset = image_generator.flow_from_directory(
                                            batch_size=32,
                                            directory=path,
                                            shuffle=True,
                                            color_mode = 'rgb',
                                            target_size=(128, 128),
                                            subset="validation",
                                            class_mode='categorical')

model1=tf.keras.Sequential([
   tf.keras.layers.Conv2D(128,4),
   tf.keras.layers.MaxPool2D((2,2)),
   tf.keras.layers.Flatten(),
   tf.keras.layers.Dense(64,activation = 'relu'),
   tf.keras.layers.Dropout(0.5),
   tf.keras.layers.Dense(4,activation='softmax')


modell.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])


model2=tf.keras.Sequential([
   tf.keras.layers.Conv2D(128,4),
   tf.keras.layers.MaxPool2D((2,2)),
   tf.keras.layers.Flatten(),
   tf.keras.layers.Dense(64,activation = 'relu'),
   tf.keras.layers.Dropout(0.5),
   tf.keras.layers.Dense(4,activation='softmax')


model2.compile(optimizer='SGD',loss='categorical_crossentropy',metrics=['accuracy'])

model3=tf.keras.Sequential([
   tf.keras.layers.Conv2D(128,4),
   tf.keras.layers.MaxPool2D((2,2)),
   tf.keras.layers.Flatten(),
   tf.keras.layers.Dense(64,activation = 'relu'),
   tf.keras.layers.Dropout(0.5),
   tf.keras.layers.Dense(4,activation='softmax')

   ])
model3.compile(optimizer='RMSprop',loss='categorical_crossentropy',metrics=['accuracy'])

history1=model1.fit_generator(
   train_dataset,
   steps_per_epoch = train_dataset.samples // 32,
   validation_data = validation_dataset,
   validation_steps = validation_dataset.samples // 32,
   epochs = 5)

history2=model2.fit_generator(
    train  dataset,
    steps_per_epoch = train_dataset.samples // 32,
    validation_data = validation_dataset,
    validation_steps = validation_dataset.samples // 32,
    epochs = 5)

history3=model3.fit_generator(
    train_dataset,
    steps_per_epoch = train_dataset.samples // 32,
    validation_data = validation_dataset,
    validation_steps = validation_dataset.samples fl 32,
    epochs = 5)

model1.summary()

model_hist1=history1.history
model_hist1=pd.DataFrame(model_hist1)
model_hist1['epoch']=history1.epoch
model_hist1.head()

model_hist2=history2.history
model_hist2=pd.DataFrame(model_hist2)
model_hist2['epoch']=history2.epoch
model_hist2.head()

model hist3=history3.history
model_hist3=pd.DataFrame(model_hist3)
model_hist3['epoch']=history3.epoch
model_hist3.head()

plt.figure(figsize=(20,5))
plt.subplot(1,2,1)
plt.plot(model hist1['epoch'],model hist1['loss'],label='training loss')
plt.subplot(1,2,1)
pit.plot(model_hist1['epoch'],model_hist1['val_loss'],label='validation_loss')
pit.legend()
plt.title('LOSS')
plt.subplot(1,2,2)
plt.plot(model_hist1['epoch'],model hist1['accuracy'],label='training accuracy')
plt.subplot(1,2,2)
pit.plot(model_hist1['epoch'],model_hist1['val_accuracy'],label='validation_accuracy')
plt.legend()
pit.title('Acuraccy')
plt.show()


pit.figure(figsize=(20,5))
plt.subplot(1,2,1)
pit.plot(model_hist2['epoch'],model_hist2['loss'],label='training_loss')
plt.subplot(1,2,1)
plt.plot(model_hist2['epoch'],model_hist2['val_loss'],label='validation_loss')
plt.legend()
plt.title('LOSS')
pit.subplot(1,2,2)
pit.plot(model_hist2['epoch'],model_hist2['accuracy'],label='training_accuracy')
plt.subplot(1,2,2)
plt.plot(model_hist2['epoch'],model_hist2['val_accuracy'],label='validation_accuracy')
plt.legend()
plt.title('Acuraccy')
plt.show()

plt.figure(figsize=(20,5))
plt.subplot(1,2,1)
plt.plot(model_hist3['epoch'],model_hist3['loss'],label='training_loss')
plt.subplot(1,2,1)
plt.plot(model_hist3['epoch'],model_hist3['val_loss'],label='validation_loss')
plt.legend()
plt.title('LOSS')
plt.subplot(1,2,2)
plt.plot(model_hist3['epoch'],model_hist3['accuracy'],label='training_accuracy')
plt.subplot(1,2,2)
plt.plot(model_hist3['epoch'],model_hist3['val_accuracy'],label='validation_accuracy')
plt.legend()
plt.title('Acuraccy')
plt.show()
```

CONCLUSION

In the first stage, it is desired to optimize the topology of the network to find the number of layers, the number of neurons in each layer, and the type of activation function for each layer. In the second stage, the optimization of the learning rates for each of the optimization algorithms is desired. In the third stage, the performance of the model with the optimized topology and learning rates is tested with different number of epochs and batch sizes for different optimization algorithms. In order to optimize the hyper-parameters, seven optimization algorithms including modern algorithms with adaptive learning rates were employed. The pseudo codes, pros and cons of each algorithm are discussed briefly. The developed deep model can automatically select the most significant predictor input variables, formulate the model structure, and solve the unknown parameters of the regression equation.

- The sequential minimal optimization (SMO) method has better predictions than the multilayer perceptron neural Network (MLPNN) method.
- The predictive system using the SMO method can predict the price of crude oil more accurately than the MLPNN method.
- Predictive system with good accuracy can support leaders in making sale and
- Purchase decisions in financial products better.
- The results of this study can be used as a reference in developing predictive systems in other fields.