DEEP LEARNING APPROACH TO SOIL TYPE IDENTIFICATION

Mrs. SUNITHA A, M.E Assistant Professor,

Department of Computer Science and Engineering

Mr. ANIL SHEBIN S J, B.E, Student of Computer Science Engineering

Mr. DHARANI S, B.E, Student of Computer science and Engineering

St. Joseph College of Engineering, Sriperumbudur, Chennai.

Abstract

When composition is degraded or decomposed into minute fragments, minerals form, and soil is made up of these minerals. Soils are made up of loose humic substances materials that include 25% oxygen, 25% liquid, 45 percent minerals, and 5% organic compounds. Like different soil classification systems, the various soil system has multiple levels of information varying from the most basic to the most specific. Examples include cinder soil, siliceous soil, marsh soil, and yellow soil. Using TensorFlow and Keras, we demonstrate how to classify type of soil that used a convolutional network. The proposed technique distinguishes soil types from photos using CNN. The soil types were classified using the CNN model. The success of the acquired findings should improve if the CNN methodology is supplemented with additional image processing techniques and appropriately recognizes soil types. We demonstrated how successfully convolutional neural networks CNN analyses various types of images.

1. Keywords: Soil type, deep learning, TensorFlow, Keras, CNN

Introduction

Land cover is a direct result of the interaction between natural environment and human activities. It mainly focuses on describing the natural properties of the earth's surface which has specific time and space characteristics. Changes in land cover will cause the changes in the climate and environmental characteristics, which has an important influence on the social economy and ecosystem. The main form of

land cover is different types of soil, including cultivated lands, woodlands, grasslands, and bare lands. Therefore, it is of great significance to classify different types of soil quickly and accurately for land cover research, soil investigation, and mapping.

The early classification method is the land use topographic map obtained by combining with the actual ground survey. Now the classification technology of the remote sensing image is mostly used to realize the classification of different types of soil Visible and near-infrared spectroscopy technology is a fast, nondestructive measurement method. It has been widely used in medicine, agriculture, oil, and other fields The spectral analysis method indirectly obtains useful information of the substance. Through establishing an effective correction model between the spectrum and the information, the result is obtained. The spectral technology is introduced into the classification of soil, the remote sensing image information is replaced by the spectral information, and different types of soil models are established. It can be fast and nondestructive to realize the classification of soil.

Literature Survey

The paper "Soil Classification Methodology: Critical Analysis" By Mbuya Mukombo Jr. , Mutonkole Ngomba H. , Musimba Kasiya A , Ngoy Biyukaleza in 2018, the soil is a material that has a wide dispersion of characteristics tied, mainly to its granulometry, its mineralogical constitution, and its water percentage. For its identification, several researchers in the field, have proposed different types of classifications for several decades. The objective of the classification is to standardize the naming of soils for their identification

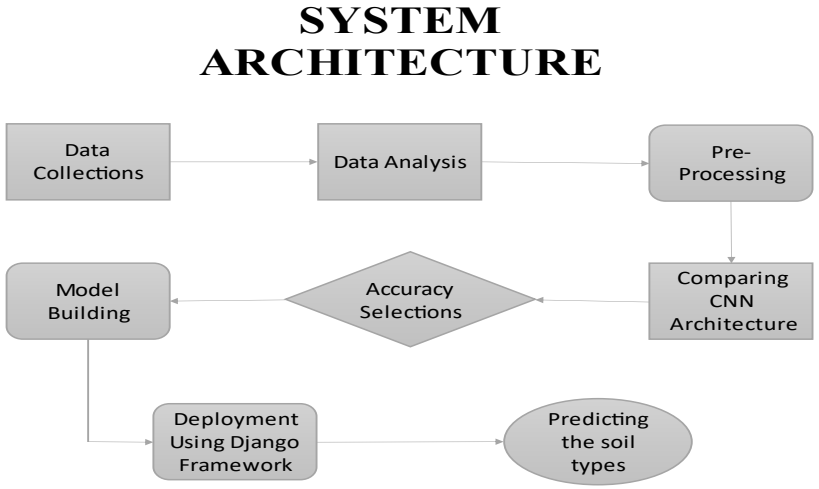
The paper ": soil texture classification with 1d convolutional neural networks based on hyperspectral data" F. M. Riese1 , S. Kelle in 2019, Soil texture is important for many environmental processes. In this paper, we study the classification of soil texture based on hyperspectral data. We develop and implement three 1-dimensional (1D) convolutional neural networks (CNN): the LucasCNN, the LucasResNet which contains an identity block as residual network, and the LucasCoordConv with an additional coordinates layer.

The paper "survey on soil classification using different techniques" By Greema S Raj, Lijin Das S in 2021 Soil has a vital part in successful agriculture. The main role of the soil is to support the growth of agriculture and horticulture crops. There

are several kinds of soil, each type of soil have distinct features and have different kinds of crops that can grow on. Therefore it is needed to know the characteristics and features of soils to know which crop can grow better on a particular soil.

System Design

The early classification method is the land use topographic map obtained by combining with the actual ground survey. Now the classification technology of the remote sensing image is mostly used to realize the classification of different types of soil Visible and near-infrared spectroscopy technology is a fast, nondestructive measurement method. So this project can easily classify the Soil Type. A collection of Soil images we have. To train the machine to classify the types of Soils. This project contains four different Soils like Angry, Happy, Cry and Neutral. We train to teach the machine to achieve the accuracy and get the possible outcome.



The data use is usually split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. It has the test dataset (or subset) in order to test our models and it will do this using Tensor flow library in Python using the Keras method.

IMPLEMENTATION

Module 1:

Manual architecture

```
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import glob
import os
from PIL import Image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Conv2D, Convolution2D, MaxPool2D,
MaxPooling2D, Dense, Dropout, Flatten, Activation
from tensorflow.keras.models import Sequential
from keras.callbacks import ModelCheckpoint
```

In [ ]:

```
dir_name_train_Cinder_Soil='dataset/train/Cinder_Soil'
dir_name_train_Laterite_Soil='dataset/train/Laterite_Soil'
dir_name_train_Peat_Soil='dataset/train/Peat_Soil'
dir_name_train_Yellow_Soil='dataset/train/Yellow_Soil'
```

In [ ]:

```
def plot_images(item_dir, n=6):
    all_item_dir = os.listdir(item_dir)
    item_files = [os.path.join(item_dir, file) for file in all_item_dir][:n]

    plt.figure(figsize=(80, 40))
    for idx, img_path in enumerate(item_files):
        plt.subplot(3, n, idx+1)
        img = plt.imread(img_path)
        plt.imshow(img, cmap='gray')
        plt.axis('off')

    plt.tight_layout()
```

In [ ]:

```
def Images_details_Print_data(data, path):
    print(" ====== Images in: ", path)
    for k, v in data.items():
        print("%s:\t%s" % (k, v))

def Images_details(path):
    files = [f for f in glob.glob(path + "**/*.*", recursive=True)]
    data = {}
    data['images_count'] = len(files)
    data['min_width'] = 10**100  # No image will be bigger than that
    data['max_width'] = 0
    data['min_height'] = 10**100  # No image will be bigger than that
```

```
    data['max_height'] = 0


    for f in files:
        im = Image.open(f)
        width, height = im.size
        data['min_width'] = min(width, data['min_width'])
        data['max_width'] = max(width, data['max_height'])
        data['min_height'] = min(height, data['min_height'])
        data['max_height'] = max(height, data['max_height'])

    Images_details_Print_data(data, path)
```

In [ ]:

```
print("")
print("Trainned data for Cinder_Soil:")
print("")
Images_details(dir_name_train_Cinder_Soil)
print("")
plot_images(dir_name_train_Cinder_Soil, 10)
```

In [ ]:

```
print("")
print("Trainned data for Laterite_Soil:")
print("")
Images_details(dir_name_train_Laterite_Soil)
print("")
plot_images(dir_name_train_Laterite_Soil, 10)
```

In [ ]:

```
print("")
print("Trainned data for Peat_Soil:")
print("")
Images_details(dir_name_train_Peat_Soil)
print("")
plot_images(dir_name_train_Peat_Soil, 10)
```

In [ ]:

```
print("")
print("Trainned data for Yellow_Soil:")
print("")
Images_details(dir_name_train_Yellow_Soil)
print("")
plot_images(dir_name_train_Yellow_Soil, 10)
```

In [ ]:

```
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.
2,horizontal_flip=True)
training_set=train_datagen.flow_from_directory('dataset/train',target_size=(2
75,259),batch_size=32,class_mode='categorical')
```

In [ ]:

```
test_datagen=ImageDataGenerator(rescale=1./255)
test_set=test_datagen.flow_from_directory('dataset/test',target_size=(275,259
),batch_size=32,class_mode='categorical')
```

In [ ]:

```
Classifier=Sequential()
```

```
Classifier.add(Convolution2D(32,(3,3),input_shape=(275,259,3),activation='rel
u'))
Classifier.add(MaxPooling2D(pool_size=(2,2)))
Classifier.add(Flatten())
Classifier.add(Dense(38, activation='relu'))
Classifier.add(Dense(4, activation='softmax'))
Classifier.compile(optimizer='rmsprop',loss='categorical_crossentropy',metric
s=['accuracy'])
```

In [ ]:

```
epochs = 15
batch_size = 32
```

In [ ]:

```
#### Fitting the model
history = Classifier.fit(
           training_set, steps_per_epoch=training_set.samples // batch_size,
           epochs=epochs,
           validation_data=test_set,validation_steps=test_set.samples //
batch_size)
```

In [ ]:

```
plt.figure(figsize=(20, 8))
plt.plot(history.history['acc'])

for i in range(epochs):
    if i%5 == 0:

plt.annotate(np.round(history.history['acc'][i]*100,2),xy=(i,history.history[
'acc'][i]))

plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.show()
```

In [ ]:

```
plt.figure(figsize=(20, 8))
plt.plot(history.history['loss'])

for i in range(epochs):
    if i%5 == 0:

plt.annotate(np.round(history.history['loss'][i]*100,2),xy=(i,history.history
['loss'][i]))

plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.show()
```

In [ ]:

Module 2:

RESNET

In [ ]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:

```
import os
import glob
from PIL import Image
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.layers import Input, ZeroPadding2D, BatchNormalization,
Activation, AveragePooling2D
from tensorflow.keras.models import Model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
```

In [ ]:

```
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.
2,horizontal_flip=True)
training_set=train_datagen.flow_from_directory('dataset/train',target_size=(2
75,259),batch_size=32,class_mode='categorical')
```

In [ ]:

```
test_datagen=ImageDataGenerator(rescale=1./255)
test_set=test_datagen.flow_from_directory('dataset/Test',target_size=(275,259
),batch_size=32,class_mode='categorical')
```

In [ ]:

```
def identity_block(X, f, filters):
    # Retrieve Filters
    F1, F2, F3 = filters

    # First component of main path
    X = Conv2D(filters = F1, kernel_size = (1, 1), strides = (1,1), padding =
'valid')(X)
    X = BatchNormalization(axis = 3)(X)
    X = Activation('relu')(X)

    # Second component of main path
    X = Conv2D(filters = F2, kernel_size = (f, f), strides = (1,1), padding =
'same')(X)
    X = BatchNormalization(axis = 3)(X)
    X = Activation('relu')(X)

    # Third component of main path
    X = Conv2D(filters = F3, kernel_size = (1, 1), strides = (1,1), padding =
'valid')(X)
    X = BatchNormalization(axis = 3)(X)
```

```
    X = Activation('relu')(X)

    return X
```

In [ ]:

```
def convolutional_block(X, f, filters, s):
    # Retrieve Filters
    F1, F2, F3 = filters

    # First component of main path
    X = Conv2D(F1, (1, 1), strides = (s,s))(X)
    X = BatchNormalization(axis = 3)(X)
    X = Activation('relu')(X)

    # Second component of main path
    X = Conv2D(filters=F2, kernel_size=(f, f), strides=(1, 1),
padding='same')(X)
    X = BatchNormalization(axis=3)(X)
    X = Activation('relu')(X)

    # Third component of main path
    X = Conv2D(filters=F3, kernel_size=(1, 1), strides=(1, 1),
padding='valid')(X)
    X = BatchNormalization(axis=3)(X)
    X = Activation('relu')(X)

    return X
```

In [ ]:

```
def ResNet50(input_shape, classes):
    # Define the input as a tensor with shape input_shape
    X_input = Input(input_shape)

    # Zero-Padding
    X = ZeroPadding2D((3, 3))(X_input)

    # Stage 1
    X = Conv2D(64, (7, 7), strides = (2, 2))(X)
    X = BatchNormalization(axis = 3)(X)
    X = Activation('relu')(X)
    X = MaxPooling2D((3, 3), strides=(2, 2))(X)

    # Stage 2
    X = convolutional_block(X, f = 3, filters = [64, 64, 256], s = 1)
    X = identity_block(X, 3, [64, 64, 256])
    X = identity_block(X, 3, [64, 64, 256])

    # Stage 3
    X = convolutional_block(X, f = 3, filters = [128, 128, 512], s = 3)
    X = identity_block(X, 3, [128, 128, 512])
    X = identity_block(X, 3, [128, 128, 512])
    X = identity_block(X, 3, [128, 128, 512])

    # Stage 4
    X = convolutional_block(X, f = 3, filters = [256, 256, 1024], s = 2)
    X = identity_block(X, 3, [256, 256, 1024])
```

```
    X = identity_block(X, 3, [256, 256, 1024])
    X = identity_block(X, 3, [256, 256, 1024])
    X = identity_block(X, 3, [256, 256, 1024])
    X = identity_block(X, 3, [256, 256, 1024])

    # Stage 5
    X = convolutional_block(X, f = 3, filters = [512, 512, 2048], s = 2)
    X = identity_block(X, 3, [512, 512, 2048])
    X = identity_block(X, 3, [512, 512, 2048])

    # AVGPOOL.
    X = AveragePooling2D((2, 2))(X)

    # output layer
    X = Flatten()(X)
    X = Dense(classes, activation='sigmoid')(X)

    # Create model
    model = Model(inputs = X_input, outputs = X)

    return model
```

In [ ]:

```
model = ResNet50(input_shape = (275,259, 3), classes = 4)
model.summary()
```

In [ ]:

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

In [ ]:

```
mc = ModelCheckpoint('resnet.h5', monitor = 'acc', verbose=1, save_best_only
= True)
```

In [ ]:

```
batch_size = 32
epochs = 10

#### Fitting the model
history = model.fit_generator(
           training_set, steps_per_epoch=training_set.samples // batch_size,
           epochs=epochs,
           validation_data=test_set,validation_steps=test_set.samples //
batch_size,
           callbacks=[mc])
```

In [ ]:

```
def graph():
    #Plot training & validation accuracy values
    plt.plot(history.history['acc'])
    plt.plot(history.history['val_acc'])
    plt.title('Model accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
```

```
    plt.show()

    # Plot training & validation loss values
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Model loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()
```

In [ ]:

```
graph()
```

Module 3:

LeNet Architecture

In [ ]:

```
# Import the warnings
import warnings
warnings.filterwarnings("ignore")
```

In [ ]:

```
# Import the neccesary Packages.
import os
import glob
import numpy as np
import pandas as pd
from PIL import Image
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Activation
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import Convolution2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import MaxPool2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
```

In [ ]:

```
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.
2,horizontal_flip=True)
training_set=train_datagen.flow_from_directory('dataset/train',target_size=(2
75,259),batch_size=32,class_mode='categorical')
```

In [ ]:

```
test_datagen=ImageDataGenerator(rescale=1./255)
test_set=test_datagen.flow_from_directory('dataset/test',target_size=(275,259
),batch_size=32,class_mode='categorical')
```

In [ ]:

```
Classifier=Sequential()
Classifier.add(Convolution2D(32,3,3,input_shape=(275,259,3),activation='relu'
))
Classifier.add(MaxPooling2D(pool_size=(2,2)))
Classifier.add(Convolution2D(128,3,3,activation='relu'))
Classifier.add(MaxPooling2D(pool_size=(2,2)))
Classifier.add(Flatten())
Classifier.add(Dense(256, activation='relu'))
Classifier.add(Dense(4, activation='softmax'))

Classifier.compile(optimizer='rmsprop',loss='categorical_crossentropy',metric
s=['accuracy'])
Classifier.summary()
```

In [ ]:

```
mc = ModelCheckpoint('lee.h5', verbose=1, save_best_only = True)
```

In [ ]:

```
epochs = 20
batch_size = 15
```

In [ ]:

```
#### Fitting the model
history = Classifier.fit(
           training_set, steps_per_epoch=training_set.samples // batch_size,
           epochs=epochs,
           validation_data=test_set,validation_steps=test_set.samples //
batch_size)
```

In [ ]:

```
plt.figure(figsize=(20, 8))
plt.plot(history.history['acc'])

for i in range(epochs):
    if i%5 == 0:

plt.annotate(np.round(history.history['acc'][i]*100,2),xy=(i,history.history[
'acc'][i]))

plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.show()
```

In [ ]:

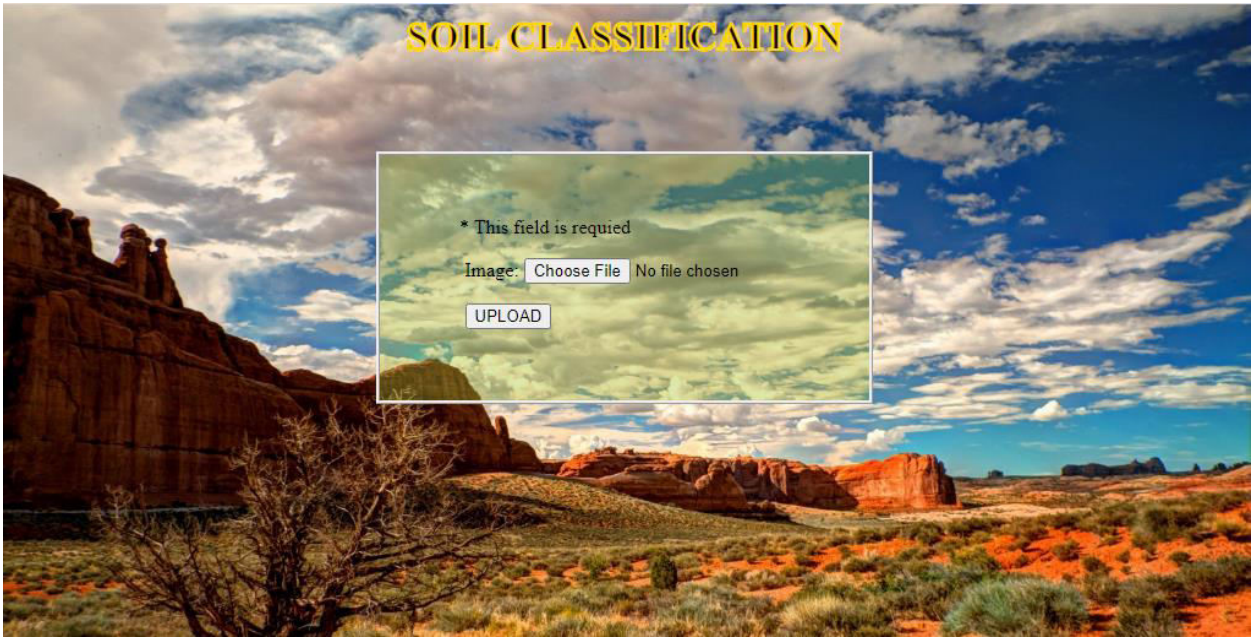
```
plt.figure(figsize=(20, 8))
plt.plot(history.history['loss'])

for i in range(epochs):
    if i%5 == 0:
```

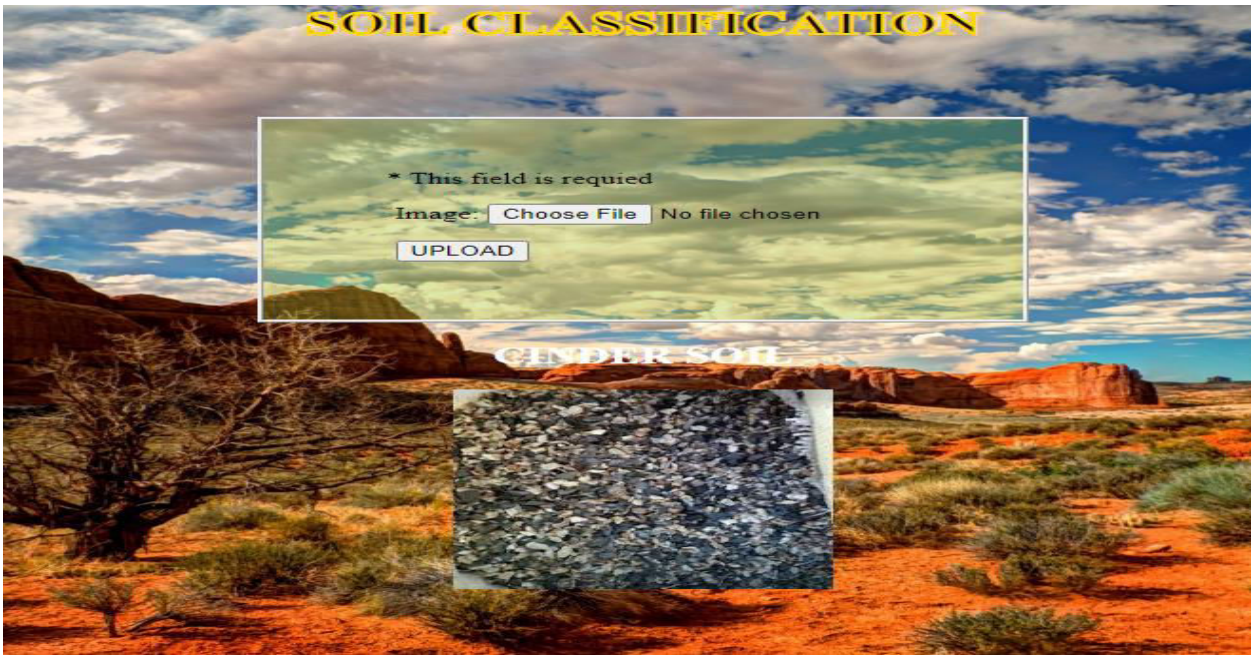
```
plt.annotate(np.round(history.history['loss'][i]*100,2),xy=(i,history.history
['loss'][i]))

plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.show()
```

In [ ]:

```
import h5py
Classifier.save('Lenet.h5')
```

SNAPSHOTS





CONCLUSION

In this project a research to classify Soil type over static soil images using deep learning techniques was developed. This is a complex problem that has already been approached several times with different techniques. While good results have been achieved using feature engineering, this project focused on feature learning, which is one of DL promises. While feature engineering is not necessary, image pre-processing boosts classification accuracy. Hence, it reduces noise on the input data. Nowadays, we find Soil type to easily software includes the use of feature engineering. A solution totally based on feature learning does not seem close yet because of a major limitation. Thus, Soil classification could be achieved by means of deep learning techniques.

FUTURE ENHANCEMENTS

Further improvement on the network's accuracy and generalization can be achieved through the following practices. The first one is to use the whole dataset during the optimization. Using batch optimization is more suitable for larger

datasets. Another technique is to evaluate soil one by one. This can lead to detect which soil are more difficult to classify. Finally, using a larger dataset for training seems beneficial. However, such a dataset might not exist nowadays. Using several datasets might be a solution, but a careful procedure to normalize them is required. Finally, using full dataset for training, pre-training on each Soil, and using a larger dataset seem to have the possibility to improve the network's performance. Thus, they should be addressed in future research on this topic.

REFERENCES:

[1] Mukombo Jr. , Mutonkole Ngomba H. , Musimba Kasiya A , Ngoy Biyukaleza "Soil Classification Methodology: Critical AnalysisMbuya" 2018

[2] Suleiman Usman1 and Basiru Usman2 "New method of soil classification in defining the dynamic condition of agricultural surface soils" 2018

[3] Greema S Raj, Lijin Das S "Survey on soil Classification using different techniques" 2021

[4] F. M. Riese1 , S. Kelle "Soil texture Classification with 1d convolutional neural networks based on hyperspectral data" 2019

[5] Joe G. Lagarteja "Android-based Soil Series Classifier Using Convolutional Neural Network" 2018

AUTHOR 1

Mrs. A Sunitha, M.E., is an Assistant Professor in the Department of Computer Science and Engineering at St. Joseph College of Engineering, Sriperumbudur, Chennai, Tamil Nadu. She has completed her M.E, CSE under Anna University Affiliation College in the year 2014. she has done his B.E, CSE under Anna University Affiliation College in the year 2012. Mrs. A Sunitha has 08 years of teaching experience and 8 publications in International Journals and Conferences. Her area of interests includes Network Security, Computer Networks, Data Science and Machine Learning. She is an active member of CSI and IEANG. She has organized various International Conferences, workshops and Seminars in the area of Computer Networks, Cloud Computing & Machine Learning respectively.

AUTHOR 2

Mr. S J Anil Shebin B.E., Student of Computer Science and Engineering at St. Joseph College of Engineering, Sriperumbudur, Chennai, Tamil Nadu. I had Attended Many Workshops and Seminars in the area of Python and Machine Learning.

AUTHOR 3

Mr. S Dharani B.E., Student of Computer Science and Engineering at St. Joseph College of Engineering, Sriperumbudur, Chennai, Tamil Nadu. I had Attended Many Workshops and Seminars in the area of Python and Machine Learning.