Nama : Rifqi Alfinnur Charisma

NIM : 19104031

1. **Land Use**

```
# IMPORT LIBRARY
import pandas as pd
import cv2
import numpy as np
import os
import re
import glob
```

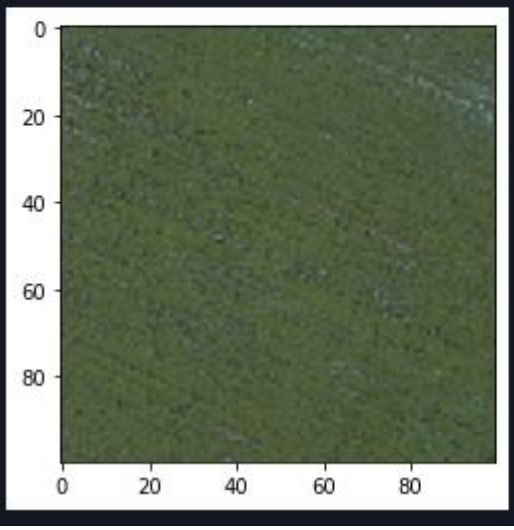
Import library yang dibutuhkan

```
# READ IMAGE
path = os.path.abspath('landuse_msib.ipynb') #absolute path of program
path = re.sub('[a-zA-Z\s._]+$', '', path) #remove unintended file
dirs = os.listdir(path+'UCMerced_LandUse/Images') #list directory in Land Use Images folder
label = 0 #label dari data
im_arr = [] #array untuk gambar
lb_arr = [] #array untuk label
X = [] #array untuk X
y = [] #array untuk y
for i in dirs: #loop all directory
    count = 0 #count data
    for pic in glob.glob(path+'UCMerced_LandUse/Images/'+i+'/*.tif'): #loop jpg difolder
        im = cv2.imread(pic) #open image pakai opencv
        im = cv2.resize(im,(100,100)) #resize gambar ke 32 x 32
        im = np.array(im) #change into array
        count = count + 1 #count data + 1
        X.append(im) #masukan gambar yg sudah dibaca kedalam X
        y.append(label) #masukan label yang sudah dibaca kedalam y
        if(count == 3): #SAmple data no.3 tampilkan nanti di plot
            im_arr.append({str(i):im}) #sample data no.3 tampilkan
    print("Jumlah "+str(i)+" : "+str(count)) #print jumlah data
    label = label + 1 #label saat ini + 1
    lb_arr.append(i) #labelarray
X = np.array(X)
y = np.array(y);
```

Membaca folder dataset lalu menghitung label dan jumah dataset tiap label menggunakan for loop. Untuk fitur dari dataset akan dimasukkan ke dalam variable X dan untuk labelnya akan dimasukkan ke dalam variable y.

```
import matplotlib.pyplot as plt

plt.imshow(X[32])
plt.show()
```



Melihat salah satu isi dari dataset, pada kasus ini saya melihat dataset dengan indeks 32

```
import os
import re
import glob
import numpy as np
import cv2
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

from tensorflow.keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.optimizers import SGD
from keras.models import load_model

from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
```

Mengimport library neural network untuk proses pembuatan model klasifikasi

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

X_train = X_train.astype('float32') #set x_train data type as float32
X_test = X_test.astype('float32') #set x_test data type as float32
X_train /= 255 #change x_train value between 0 - 1
X_test /= 255 #change x_test value between 0 - 1
y_train = to_categorical(y_train, 21) #change label to binary / categorical: [1 0 0 0] = 0, [0 1 0 0] = 1, so on
y_test = to_categorical(y_test, 21) #change label to binary / categorical
```

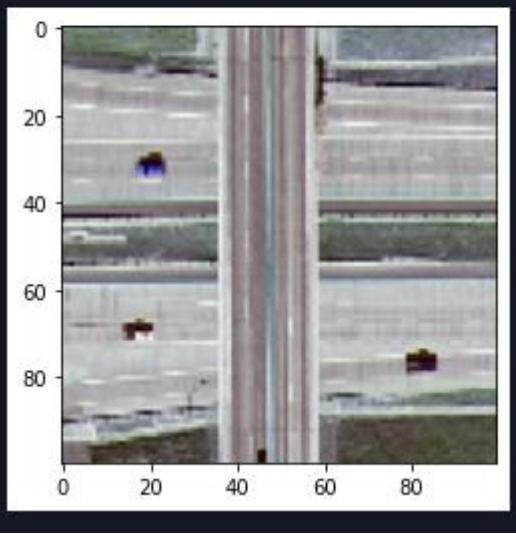
Memisahkan antara data train dan data testing. Tipe data pada data training diubah menjadi float. Encoding pada data testing.

```
print(y_train[30])
print(y_train[31])
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
```

Melihat hasil dari encoding

```
plt.imshow(X_train[35])
plt.show()
```



Melihat dataset yang sudah kita preprocessing sebelumnya.

```
model = Sequential() #model = sequential
model.add(Conv2D(32, kernel_size=(3, 3),activation='relu',input_shape=(100,100,3))) #layer convolutional 2D
model.add(MaxPooling2D(pool_size=(2,2))) #max pooling with stride (2,2)
model.add(Conv2D(32, (3, 3), activation='relu')) #layer convolutional 2D
model.add(MaxPooling2D(pool_size=(2,2))) #max pooling with stride (2,2)
model.add(Dropout(0.25)) #delete neuron randomly while training and remain 75%
model.add(Flatten()) #make layer flatten
model.add(Dense(128, activation='relu')) #fully connected layer
model.add(Dropout(0.5)) #delete neuron randomly and remain 50%
model.add(Dense(21, activation='softmax')) #softmax works
```

Membuat model konvolusi dengan input (100,100,3) dan dengan aktivasi softmax. Menambahkan layer dropout agar proses training tidak overfitting

```
epochs = 5
lrate = 0.01
decay = lrate/epochs
sgd = SGD(lr=lrate, momentum=0.9, decay=decay, nesterov=False)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
print(model.summary())
```

Menentukan iterasi untuk proses training, pada kasus kali ini kita set nilai iterasi menjadi 5. Menggunakan SGD sebagai optimizer yang cocok untuk kasus ini. Menampilkan model konvolusi menggunakan model.summary()

```
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=32)
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
Epoch 1/5
44/44 [==============================] - 15s 292ms/step - loss: 3.0566 - accuracy: 0.0519 - val_loss: 3.0356 - val_accuracy: 0.0491
Epoch 2/5
44/44 [==============================] - 11s 252ms/step - loss: 3.0294 - accuracy: 0.0569 - val_loss: 3.0099 - val_accuracy: 0.1948
Epoch 3/5
44/44 [==============================] - 11s 256ms/step - loss: 2.9553 - accuracy: 0.0988 - val_loss: 2.8480 - val_accuracy: 0.1847
Epoch 4/5
44/44 [==============================] - 11s 258ms/step - loss: 2.8898 - accuracy: 0.1251 - val_loss: 2.7889 - val_accuracy: 0.1429
Epoch 5/5
44/44 [==============================] - 11s 258ms/step - loss: 2.7842 - accuracy: 0.1421 - val_loss: 2.6233 - val_accuracy: 0.1948
Accuracy: 19.48%
```

Menampilkan proses training dan hasil akurasi yang diperoleh.

2. **Defect Detection**

```
# IMPORT LIBRARY
import pandas as pd
import cv2
import numpy as np
import os
import re
import glob
```

Mengimport library yang dibutuhkan

```
# READ IMAGE
path = os.path.abspath('defect.ipynb') #absolute path of program
path = re.sub('[a-zA-Z\s._]+$', '', path) #remove unintended file
dirs = os.listdir(path+'NEU-CLS-64/') #list directory in Land Use Images folder
label = 0 #label dari data
im_arr = [] #array untuk gambar
lb_arr = [] #array untuk label
X = [] #array untuk X
y = [] #array untuk y
for i in dirs: #loop all directory
    count = 0 #count data
    for pic in glob.glob(path+'NEU-CLS-64/'+i+'/*.jpg'): #loop jpg difolder
        im = cv2.imread(pic) #open image pakai opencv
        im = cv2.resize(im,(32,32)) #resize gambar ke 32 x 32
        im = np.array(im) #change into array
        count = count + 1 #count data + 1
        X.append(im) #masukan gambar yg sudah dibaca kedalam X
        y.append(label) #masukan label yang sudah dibaca kedalam y
        if(count == 3): #SAmple data no.3 tampilkan nanti di plot
            im_arr.append({str(i):im}) #sample data no.3 tampilkan
    print("Jumlah "+str(i)+" : "+str(count)) #print jumlah data
    label = label + 1 #label saat ini + 1
    lb_arr.append(i) #labelarray
X = np.array(X)
y = np.array(y);
```

Membaca dataset yang diperlukan. Menghitung jumlah dataset dari tiap label dan menyimpannya ke dalam variable X dan y.

```
import os
import re
import glob
import numpy as np
import cv2
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

from tensorflow.keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.optimizers import SGD
from keras.models import load_model

from sklearn.metrics import confusion_matrix
```

import beberapa library yang dibutuhkan untuk membuat model konvolusi.

```
mapping = {
    "rs":"Role in Scale",
    "pa":"Patches",
    "cr":"Crazing",
    "ps":"Pitt Surface",
    "in":"Inclusion",
    "sc":"Scratces",
    "sp":"Scratch Patches",
    "rp":"Role Patches",
    "gg":"Ground Ground"
}
```

Lakukan proses mapping agar label lebih mudah dibaca

```
fig, axs = plt.subplots(3, 3, figsize=(20, 10))
cnt = 0
row = 0
col = 0
for i in example_data:
    for key, value in i.items():
        if(cnt==3):
            row = row + 1
            col = 0
            cnt = 0
        axs[row, col].imshow(value)
        axs[row, col].set_title(mapping.get(key))
        cnt = cnt + 1
        col = col + 1
plt.show()
```

Menampilkan citra dari tiap label.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

X_train = X_train.astype('float32') #set x_train data type as float32
X_test = X_test.astype('float32') #set x_test data type as float32
X_train /= 255 #change x_train value between 0 - 1
X_test /= 255 #change x_test value between 0 - 1
y_train = to_categorical(y_train, 9) #change label to binary / categorical: [1 0 0 0] = 0, [0 1 0 0] = 1, so on
y_test = to_categorical(y_test, 9) #change label to binary / categorical
```

Memisahkan data antara data training dan data testin. Kita menggunakan data testing sebanyak 33% dan melakukan 42 kali pengacakan agar data tersebar secara merata.

```
model = Sequential() #model = sequential
model.add(Conv2D(32, kernel_size=(3, 3),activation='relu',input_shape=(32,32,3))) #layer convolutional 2D
model.add(MaxPooling2D(pool_size=(2,2))) #max pooling with stride (2,2)
model.add(Conv2D(32, (3, 3), activation='relu')) #layer convolutional 2D
model.add(MaxPooling2D(pool_size=(2,2))) #max pooling with stride (2,2)
model.add(Dropout(0.25)) #delete neuron randomly while training and remain 75%
model.add(Flatten()) #make layer flatten
model.add(Dense(128, activation='relu')) #fully connected layer
model.add(Dropout(0.5)) #delete neuron randomly and remain 50%
model.add(Dense(9, activation='softmax')) #softmax works
```

Membuat model konvolusi dengan aktifasi relu, softmax. Input (32,32,3). Dan dengan layer sebanyak 9 layer.

```
epochs = 25
lrate = 0.01
decay = lrate/epochs
sgd = SGD(lr=lrate, momentum=0.9, decay=decay, nesterov=False)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
print(model.summary())
```

Menentukan jumlah iterasi sebanyak 25 dengan SGD sebagai optimizernya.

```
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=32)
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

Melatih model menggunakan model.fit() lalu menyetak akurasinya. Diapatkan akuasi sebesar 84%.