Natural-Disasters-Intensity-Analysis-And-Classification-Using-Ai

Natural disasters not only disturb the human ecological system but also destroy the properties and critical infrastructures of human societies and even lead to permanent change in the ecosystem. Disaster can be caused by naturally occurring events such as earthquakes, cyclones, floods, and wildfires. Many deep learning techniques have been applied by various researchers to detect and classify natural disasters to overcome losses in ecosystems, but detection of natural disasters still faces issues due to the complex and imbalanced structures of images. To tackle this problem, we developed a multilayered deep convolutional neural network model that classifies the natural disaster and tells the intensity of disaster of natural The model uses an integrated webcam to capture the video frame and the video frame is compared with the Pre-trained model and the type of disaster is identified and showcased on the OpenCV window.

Dataset Link

https://drive.google.com/file/d/11-FdbTaJVrpwQmaCLV5gYYDQlfTeD0uz/view?usp=sharing

```
import tensorflow
```

```
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense, Flatten
from keras.layers import Conv2D, MaxPool2D
from tensorflow.keras import layers
from keras.layers import Dropout
from keras.models import load_model
import numpy as np
```

```
train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1./255)
```

```
!unzip /content/drive/MyDrive/dataset.zip

     Archive:  /content/drive/MyDrive/dataset.zip
     replace dataset/readme.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename:
```

```
train_ds = train_datagen.flow_from_directory('/content/dataset/train_set', target_size=(64, 64), class_mode='categorical', batch_size=5,

test_ds = train_datagen.flow_from_directory('/content/dataset/test_set', target_size=(64, 64), class_mode='categorical', batch_size=5,

     Found 742 images belonging to 4 classes.
     Found 198 images belonging to 4 classes.
```

```
model = Sequential()
```

```
model.add(Conv2D(32,(3,3), input_shape=(64, 64, 3), activation='relu'))
model.add(MaxPool2D(2,2))
model.add(Dropout(0.2))
model.add(Conv2D(32,(3,3), activation='relu'))
model.add(MaxPool2D(2,2))
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(4, activation='softmax'))
```

```
model.summary()

     Model: "sequential"
     _________________________________________________________________
      Layer (type)                Output Shape              Param #
     =================================================================
      conv2d (Conv2D)             (None, 62, 62, 32)        896

      max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
      )

      dropout (Dropout)           (None, 31, 31, 32)        0

      conv2d_1 (Conv2D)           (None, 29, 29, 32)        9248

      max_pooling2d_1 (MaxPooling  (None, 14, 14, 32)       0
      2D)

      dropout_1 (Dropout)         (None, 14, 14, 32)        0

      flatten (Flatten)           (None, 6272)              0

      dense (Dense)               (None, 128)               802944

      dropout_2 (Dropout)         (None, 128)               0

      dense_1 (Dense)             (None, 4)                 516

     =================================================================
     Total params: 813,604
     Trainable params: 813,604
     Non-trainable params: 0
     _________________________________________________________________
```

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

```
history = model.fit(train_ds,
                    steps_per_epoch = len(train_ds),
                    epochs = 20,
                    validation_data = test_ds,
                    validation_steps = len(test_ds))
```

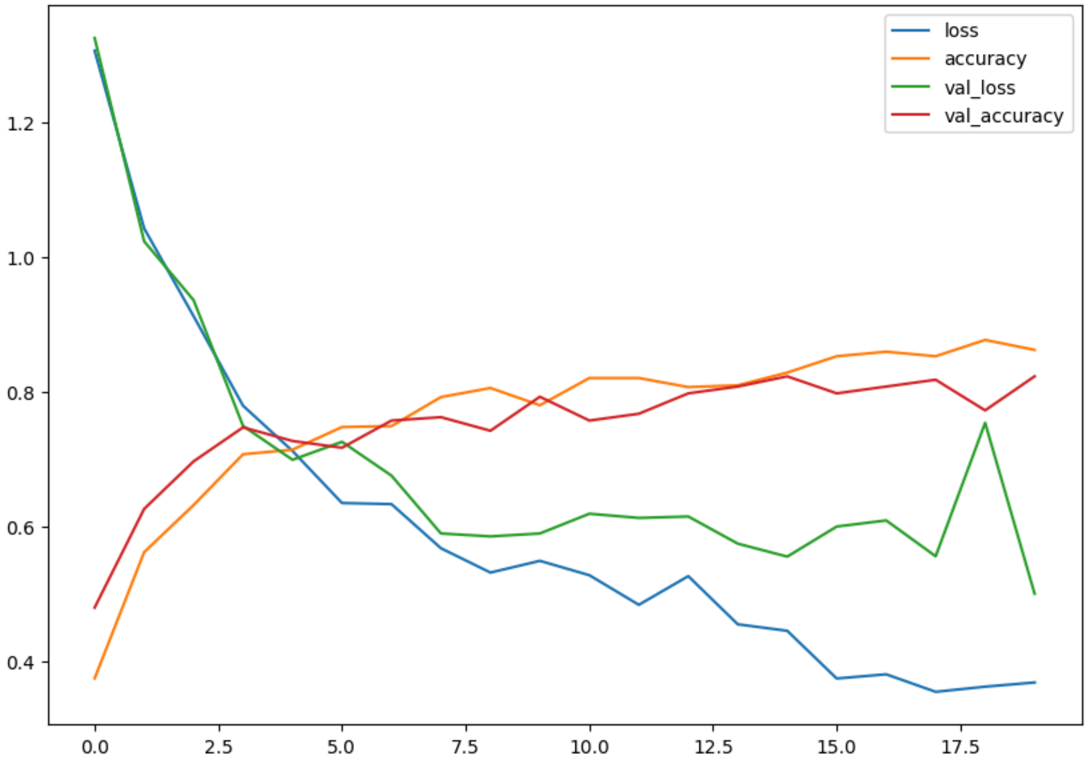
```
Epoch 1/20
149/149 [==============================] - 32s 202ms/step - loss: 1.3070 - accuracy: 0.3747 - val_loss: 1.3257 - val_accuracy: 0.479
Epoch 2/20
149/149 [==============================] - 33s 222ms/step - loss: 1.0431 - accuracy: 0.5620 - val_loss: 1.0239 - val_accuracy: 0.626
Epoch 3/20
149/149 [==============================] - 30s 197ms/step - loss: 0.9126 - accuracy: 0.6321 - val_loss: 0.9364 - val_accuracy: 0.697
Epoch 4/20
149/149 [==============================] - 28s 189ms/step - loss: 0.7796 - accuracy: 0.7075 - val_loss: 0.7495 - val_accuracy: 0.747
Epoch 5/20
149/149 [==============================] - 28s 188ms/step - loss: 0.7124 - accuracy: 0.7143 - val_loss: 0.6993 - val_accuracy: 0.727
Epoch 6/20
149/149 [==============================] - 27s 184ms/step - loss: 0.6351 - accuracy: 0.7480 - val_loss: 0.7260 - val_accuracy: 0.717
Epoch 7/20
149/149 [==============================] - 27s 184ms/step - loss: 0.6335 - accuracy: 0.7493 - val_loss: 0.6758 - val_accuracy: 0.757
Epoch 8/20
149/149 [==============================] - 33s 220ms/step - loss: 0.5679 - accuracy: 0.7925 - val_loss: 0.5899 - val_accuracy: 0.762
Epoch 9/20
149/149 [==============================] - 29s 193ms/step - loss: 0.5318 - accuracy: 0.8059 - val_loss: 0.5855 - val_accuracy: 0.742
Epoch 10/20
149/149 [==============================] - 28s 188ms/step - loss: 0.5492 - accuracy: 0.7803 - val_loss: 0.5898 - val_accuracy: 0.792
Epoch 11/20
149/149 [==============================] - 28s 187ms/step - loss: 0.5279 - accuracy: 0.8208 - val_loss: 0.6192 - val_accuracy: 0.757
Epoch 12/20
149/149 [==============================] - 28s 190ms/step - loss: 0.4839 - accuracy: 0.8208 - val_loss: 0.6130 - val_accuracy: 0.767
Epoch 13/20
149/149 [==============================] - 29s 195ms/step - loss: 0.5265 - accuracy: 0.8073 - val_loss: 0.6151 - val_accuracy: 0.798
Epoch 14/20
149/149 [==============================] - 28s 179ms/step - loss: 0.4549 - accuracy: 0.8100 - val_loss: 0.5748 - val_accuracy: 0.808
Epoch 15/20
149/149 [==============================] - 27s 181ms/step - loss: 0.4453 - accuracy: 0.8288 - val_loss: 0.5555 - val_accuracy: 0.823
Epoch 16/20
149/149 [==============================] - 26s 171ms/step - loss: 0.3745 - accuracy: 0.8531 - val_loss: 0.6002 - val_accuracy: 0.798
Epoch 17/20
149/149 [==============================] - 28s 187ms/step - loss: 0.3807 - accuracy: 0.8598 - val_loss: 0.6093 - val_accuracy: 0.808
Epoch 18/20
149/149 [==============================] - 28s 187ms/step - loss: 0.3547 - accuracy: 0.8531 - val_loss: 0.5560 - val_accuracy: 0.818
Epoch 19/20
149/149 [==============================] - 28s 187ms/step - loss: 0.3623 - accuracy: 0.8774 - val_loss: 0.7542 - val_accuracy: 0.772
Epoch 20/20
149/149 [==============================] - 28s 189ms/step - loss: 0.3686 - accuracy: 0.8625 - val_loss: 0.5004 - val_accuracy: 0.823
```

```
model.save('model.h5')
```

```
result = model.evaluate(test_ds)
```

```
40/40 [==============================] - 4s 110ms/step - loss: 0.5093 - accuracy: 0.8030
```

```
import pandas as pd
pd.DataFrame(history.history).plot(figsize=(10, 7));
```



```
import keras.utils as image
```

```
model = load_model('model.h5')
```

```
img = image.load_img("/content/dataset/test_set/Flood/1015.jpg", target_size = (64, 64))
img
```

```
x = image.img_to_array(img)
x = np.expand_dims(x, axis = 0)
pred = np.argmax(model.predict(x), axis=-1)
pred
```

```
1/1 [==============================] - 0s 140ms/step
array([2])
```

```
index = ['Cyclone', 'Earthquake', 'Flood', 'WildFire']
result = np.array(index[pred[0]])
result
```

```
array('Flood', dtype='<U5')
```

```
img = image.load_img("/content/dataset/test_set/Earthquake/1327.jpg", target_size = (64, 64))
img
```

```
x = image.img_to_array(img)
x = np.expand_dims(x, axis = 0)
pred = np.argmax(model.predict(x), axis=-1)
pred
```

```
1/1 [==============================] - 0s 29ms/step
array([1])
```

```
index = ['Cyclone', 'Earthquake', 'Flood', 'WildFire']
result = np.array(index[pred[0]])
result
```

```
array('Earthquake', dtype='<U10')
```