**DAY 01**

**19.06.2023**

**Weather Classifier using ResNet(CNN)**

**Abstract:**

Weather has a great impact on our daily lives. How we dress, how we plan our activities for the day, our schedule and even our mood are determined by the weather of the day.

Evaluating weather conditions from a single image is called weather classification and is done by human vision or expensive sensors. It plays a key role in many weather systems that have a significant impact on our daily lives.

This weather classifier uses a ResNet (convolutional neural network) model to achieve this feat. The ResNet model and dataset are models that were tested earlier and is available in Kaggle to learn about image processing.

This model provides 85-87% accuracy and the datasets contain 6862 images for classifying different types of weather.

Images are classified into 11 classes, they are: Dew, fog/smog, frost, glaze, hail, lightning, rain, rainbow, hoarfrost, sandstorm, and snow.

The training accuracy achieved by ResNet150V2 is 91% and the validation accuracy is 87%, while Xception achieves 90% and 86% accuracy respectively.

Title: Weather Classification using ResNet

Algorithm related to CNN:

* ResNet

Source Code:

# Common

import os

import keras

import numpy as np

import tensorflow as tf

# Data

from keras.preprocessing.image import ImageDataGenerator as IDG

# Data Viz

import seaborn as sns

import plotly.express as px

import matplotlib.pyplot as plt

# Pre-Trained Models

from tensorflow.keras.applications import InceptionV3, Xception, ResNet152V2, ResNet50V2, ResNet50

# Model

from keras.models import Sequential, load\_model

from keras.layers import GlobalAvgPool2D as GAP, Dense, Dropout, Conv2D, MaxPool2D, BatchNormalization, Layer, ReLU, Add, InputLayer, Multiply

# Callbacks

from keras.callbacks import EarlyStopping, ModelCheckpoint

# Model viz

from tensorflow.keras.utils import plot\_model

root\_path = '/content/drive/MyDrive/Weather/dataset'

class\_names = sorted(os.listdir(root\_path))

n\_classes = len(class\_names)

print(f"Total No. of Classes : {n\_classes}")

class\_dis = [len(os.listdir(root\_path + "/" + name)) for name in class\_names]

class\_dis

fig = px.pie(names=class\_names, values=class\_dis, title="Class Distribution")

fig.update\_layout({'title':{'x':0.5}})

fig.show()

plt.figure(figsize=(10,8))

sns.barplot(

    x=class\_names,

    y=class\_dis

)

plt.axhline(np.mean(class\_dis), alpha=0.5, linestyle='--', color='k', label="Mean")

plt.title("Class Distribution")

plt.legend(fontsize=15)

plt.show()

# Initialize DataGenerator

train\_gen = IDG(rescale=1./255, horizontal\_flip=True, rotation\_range=20, validation\_split=0.2)

# Load Data

train\_ds = train\_gen.flow\_from\_directory(root\_path, target\_size=(256,256), class\_mode="binary", subset='training', shuffle=True, batch\_size=32)

valid\_ds = train\_gen.flow\_from\_directory(root\_path, target\_size=(256,256), class\_mode="binary", subset='validation', shuffle=True, batch\_size=32)

i=1

plt.figure(figsize=(15,20))

for images, labels in train\_ds:

    id = np.random.randint(len(images))

    image, label = images[id], int(labels[id])

    plt.subplot(5, 4, i)

    plt.imshow(image)

    plt.title(class\_names[label])

    plt.axis('off')

    i+=1

    if i>=21: break

plt.show()

name = 'resnet'

# TL Model

base\_model = ResNet152V2(include\_top=False, input\_shape=(256,256,3))

base\_model.trainable = False

# Model

model = Sequential([

    base\_model,

    GAP(),

    Dense(256, activation='relu'),

    Dropout(0.4),

    Dense(128, activation='relu'),

    Dropout(0.2),

    Dense(n\_classes, activation="softmax")

])

# Compile

model.compile(

    loss='sparse\_categorical\_crossentropy',

    optimizer='adam',

    metrics=['accuracy']

)

# Callbacks

cbs = [

    EarlyStopping(patience=5, restore\_best\_weights=True),

    ModelCheckpoint(name + ".h5", save\_best\_only=True)

]

model = load\_model('/content/resnet.h5')

i=1

plt.figure(figsize=(15,20))

for images, labels in train\_ds:

    id = np.random.randint(len(images))

    image, label = images[id], int(labels[id])

    pred\_label = class\_names[np.argmax(model.predict(image[np.newaxis,...]))]

    plt.subplot(5, 4, i)

    plt.imshow(image)

    plt.title(f"Org: {class\_names[label]},Pred: {pred\_label}")

    plt.axis('off')

    i+=1

    if i>=21: break

plt.show()

model.evaluate(valid\_ds)

Images:







