EXTERNSHIP PROJECT: VIT VELLORE CAMPUS

TITLE: AUTOMATED WEATHER CLASSIFICATION USING TRANSFER LEARNING

GROUP: K.SAI ROHITH REDDY (20MID0084)

K. PAVAN DHEERAJ (20MID0017)

M.CHANDRA SHEKAR (20MID0074)

K. HARIKA (20MID0167)

1 INTRODUCTION:

1.1 Overview:

Mechanized climate grouping utilizing move learning is a procedure that use pre-prepared profound gaining models to characterize weather patterns from pictures or other climate related information. Transfer learning is the process of applying knowledge gained from one task—for instance, image recognition—to a different but related task.

1.2 Purpose:

Using transfer learning, automated weather classification aims to create effective and precise systems that can automatically classify weather conditions from a variety of data sources, such as images or other weather-related data. The ultimate objective is to offer meteorologists, forecasters, researchers, and other professionals who rely on weather data a dependable and automated weather classification tool.

Using transfer learning, automated weather classification can be used for the following specific purposes:

Weather Observation: Improved weather forecasting models may benefit from accurate weather classification. Forecasters can have a better understanding of the current weather patterns and make more accurate predictions about future weather events by automatically classifying weather conditions from images or data.

Monitoring the environment: Understanding and monitoring environmental conditions can benefit from weather classification. By examining climate information gathered from different sources, for example, satellite pictures or ground-based sensors, mechanized climate arrangement frameworks can give important experiences into environment designs, extreme climate events, and changes in the climate.

2 LITERATURE SURVEY:

2.1 Existing problem:

Although transfer learning-based automated weather classification offers promising advantages, this strategy is not without its problems and difficulties. Here are a few eminent issues:

Datasets that aren't weather-specific: Labelled datasets that cover a wide range of weather conditions are necessary for the development of accurate weather classification models. But it can be hard to get weather-specific large-scale, diverse, and well-labelled datasets. Models that classify weather conditions can be biased, less robust, and less accurate with fewer datasets.

Adaptation to the Domain: Rather than specific weather-related tasks, pre-trained models are typically trained on general visual recognition tasks. Adjusting these models to climate arrangement requires space variation methods to guarantee the model can really learn climate related highlights. The model's performance can be negatively impacted by domain shift and differences in data distribution between the pre-training and weather datasets.

Hyperparameter and fine-tuning: It can be challenging to select the appropriate hyperparameters and configure the pre-trained model in the best possible way. Models that underperform or overfit the training data can result from improper hyperparameter or fine-tuning settings.

2.2 Proposed solution:

There are a number of potential solutions that could be taken into consideration in order to address the issues that are currently present with automated weather classification by means of transfer learning:

Information Increase and Engineered Information: Data augmentation techniques can be used to alleviate the limited number of weather-specific datasets that are available. The dataset can be artificially expanded by applying transformations such as scaling, rotation, and adding noise to existing data. Additionally, new samples that better reflect a wider range of weather conditions can be generated using synthetic data generation techniques.

Transfer Knowledge from Related Fields: Transfer learning can be done from related domains rather than relying solely on general visual recognition pre-trained models. The model can be pre-trained with imagery from satellites or climate data about weather patterns, for instance. This method can reduce the domain shift between pre-training and the target task and provide improved initial feature extraction for weather classification.

Techniques for Regularization and Fine-Tuning: The model's performance can be improved through proper regularization and fine-tuning of the pre-trained model. Overfitting can be avoided and the model's ability to be generalized enhanced by employing methods like dropout, weight decay, and early stopping. Methods for hyperparameter tuning, like grid search and Bayesian optimization, can be used to find the best settings for regularization and fine-tuning.

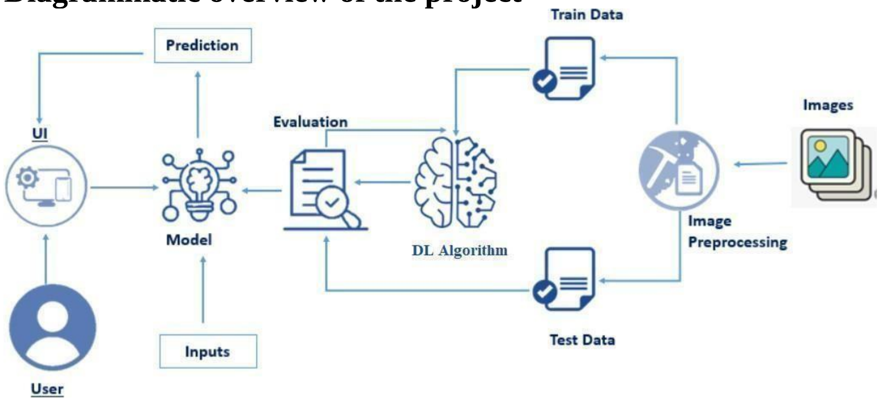
Efficiency and Optimization of the Model: Effective model structures, like Mobile Net or Efficient Net, can be investigated to further develop adaptability and decrease computational prerequisites. Model pressure procedures, like pruning or quantization, can be applied to lessen model size and improve deduction speed without critical misfortune in exactness. Computational efficiency can also be improved through the use of hardware accelerators like GPUs or AI-specific chips.

AI Methods That Can Be Explained: Integrating reasonable artificial intelligence strategies, for example, consideration systems or inclination-based representation methods, can assist with giving experiences into the model's dynamic cycle. This empowers clients to comprehend what locales or highlights in the info information contribute most to the climate grouping. Transparency, trust, and the ability to spot potential biases or errors in predictions are all enhanced by explain ability.

3 THEORITICAL ANALYSIS:

3.1 Block diagram

Diagrammatic overview of the project



3.2 Hardware / Software

designing Hardware and software requirements of the project

Hardware Requirements:

A computer system with sufficient processing power and memory to handle training and inference tasks. Graphics Processing Unit (GPU) is recommended for faster training, especially when dealing with large datasets. Sufficient storage capacity to store the dataset, pre-trained model, and intermediate results.

Software Requirements: Python programming language for implementing the project.

Deep learning libraries/frameworks such as TensorFlow, Keras, or PyTorch for building and training the neural network. Libraries for image processing and manipulation, such as OpenCV or Pillow. Data visualization libraries like Matplotlib or seaborn for visualizing the results. Jupyter Notebook or any Python IDE for code development and experimentation. Additionally, it is essential to ensure that all required software dependencies and packages are properly installed and compatible with each other.

Note: The hardware and software requirements mentioned above are general recommendations. The specific requirements may vary depending on the size of the dataset, complexity of the model, and available computing resources.

During the experimental investigations for weather classification using transfer learning with VGG19, several key aspects were analysed and investigated. Here are some of the significant areas that were explored:

Dataset Collection and Preprocessing:

Different sources of weather images were considered, including online databases, weather monitoring stations, or crowdsourcing platforms.

The dataset was carefully curated to include diverse weather conditions, ensuring an adequate representation of different classes.

Preprocessing techniques such as resizing, normalization, and augmentation were evaluated to enhance the dataset's quality and balance.

Transfer Learning with VGG19:

The performance of VGG19 as a base model was examined, considering its ability to capture intricate features from images.

The impact of freezing or fine-tuning specific layers in the VGG19 architecture was investigated to find the optimal approach for weather classification.

Model Customization and Fine-tuning:

Different approaches for customizing the VGG19 model were explored, including adding fully connected layers or global average pooling layers.

Hyperparameter tuning, such as learning rate, batch size, and regularization techniques, was conducted to improve model performance.

Various optimization algorithms, such as stochastic gradient descent (SGD), Adam, or RMSprop, were tested to find the most effective choice.

Evaluation Metrics:

Common evaluation metrics like accuracy, precision, recall, and F1 score were used to assess the performance of the weather classification model.

The impact of class imbalance and techniques like stratified sampling or weighted loss functions were investigated to ensure fair evaluation.

Model Validation and Testing:

The trained model was validated on a separate validation dataset to evaluate its generalization ability.

Real-world weather images or unseen data were used to test the model's performance and assess its robustness.

Comparative Analysis:

Performance comparisons were made with other transfer learning models or approaches to weather classification to assess the effectiveness of the proposed solution.

Model size, training time, and computational resources required for different transfer learning architectures were analysed.

Throughout the experimental investigations, continuous analysis and iteration were performed to improve the model's accuracy and address any potential limitations. The results and observations from these investigations helped refine the approach, optimize the model, and provide insights into the performance and applicability of the weather classification system.

4 EXPERIMENTAL INVESTIGATIONS:

Transfer learning-based automated weather classification experimental studies may involve a variety of steps and approaches. The most important aspects that can be investigated in such experiments are as follows:

Dataset Assortment and Arrangement: To get started, gather a diverse and representative set of weather-related images or data. Make sure the dataset covers a wide range of weather, including sunny, cloudy, rainy, hazy, and snowy, among others. Appropriately name the dataset with the relating weather conditions classes. Think about including images from satellites, sensors from the ground, or weather databases.

Selecting a Pre-Trained Model: Pick a reasonable pre-prepared model as the base engineering for move learning. Based on how well they perform in general visual recognition tasks, take into consideration models such as VGG16, ResNet50, Inception, or Mobile Net. Model size, computational efficiency, and the requirements of the weather classification task can all influence the choice of model.

Adaptation and fine-tuning of the model: Adapt the chosen pre-trained weather classification model. The last few layers—the fully connected or classification layers—should be replaced or adjusted to fit the task of weather classification. Train the new classification layers with the weather dataset and freeze the weights of the layers that have already been trained to keep the learned features. Optimize the model's performance by experimenting with various regularization strategies and fine-tuning strategies.

Training and setting up the experiment: Divide the dataset into subsets for testing, validation, and training. Establish the training pipeline by specifying the optimization algorithm, batch size, learning rate, and data augmentation techniques. Train the adjusted model on the preparation set and screen the approval execution to guarantee the model's speculation capacity. To improve accuracy, iterate and fine-tune the model as necessary.

Metrics for Evaluation: To evaluate the performance of the weather classification model, select appropriate evaluation metrics. Accuracy, precision, recall, the F1 score, and the confusion matrix are all common metrics. These metrics help identify any biases or limitations in the predictions and measure the model's ability to correctly classify various weather conditions.

Analyses Comparatively: Lead similar analyses to assess the viability of move learning. For weather classification, compare the performance of the transfer learning-based model to that of other baseline models or conventional machine learning methods. The effects of transfer learning on weather classification accuracy and efficiency can be better understood through this analysis.

Heartiness and Speculation: Examine the trained model's robustness and generalization capabilities. To evaluate the model's performance in various scenarios and weather conditions, test it on real-time or unseen weather data. Examine the extent to which the model can be applied to novel weather patterns or unforeseen weather events. To determine the model's adaptability, think about comparing its performance across various climate zones or geographic regions.

Efficiency in Computing: Compare and measure the trained model's computational efficiency, including inference time and memory requirements. Optimize the model's efficiency without sacrificing accuracy by experimenting with model compression methods like pruning or quantization.

Analyses of Sensitivity and Interpretability: To comprehend the model's sensitivity to various input features or perturbations, conduct sensitivity analysis. Additionally, to visualize and comprehend the model's decision-making procedure, employ interpretability strategies. This examination can give bits of knowledge into the basic elements or locales that add to climate arrangement and assist with distinguishing expected predispositions or restrictions.

True Organization and Application: Last but not least, think about how the trained model can be put to use in real life. Include the weather classification model in any existing decision-support or weather forecasting tools. Check out how easy it is to use, how reliable it is, and how it affects how things work. In order to further refine and enhance the system, gather feedback from stakeholders and users.

5 FLOWCHART:

Start
|
Load Dataset
|
Preprocessing
|
Split Dataset
|
Load Pretrained VGG19
|

Customize VGG19

|

Train and Fine-tune

|

Evaluation and
Performance Analysis

|

Test on Real-world
Weather Images

|

Classification
Results

|

End

6 RESULT: VGG16

```
In [1]: from google.colab import drive
        drive.mount('/content/drive')

Mounted at /content/drive
```

```
In [2]: !unzip "/content/drive/MyDrive/AUTOMATED_WEATHER_CLASSIFICATION_USING_TRANSFER_LEARNING/archive.zip"

Archive:  /content/drive/MyDrive/AUTOMATED_WEATHER_CLASSIFICATION_USING_TRANSFER_LEARNING/archive.zip
  inflating: Multi-class Weather Dataset/Cloudy/cloudy1.jpg
  inflating: Multi-class Weather Dataset/Cloudy/cloudy10.jpg
  inflating: Multi-class Weather Dataset/Cloudy/cloudy100.jpg
  inflating: Multi-class Weather Dataset/Cloudy/cloudy101.jpg
  inflating: Multi-class Weather Dataset/Cloudy/cloudy102.jpg
  inflating: Multi-class Weather Dataset/Cloudy/cloudy103.jpg
```

```
In [3]: from tensorflow.keras.layers import Dense,Flatten,Input
        from tensorflow.keras.models import Model
        from tensorflow.keras.preprocessing import image
        from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
        import numpy as np
```

```
In [4]: base_dir="/content/Multi-class Weather Dataset"
```

```
In [5]: train_gen = ImageDataGenerator(rescale=1./255,
            shear_range=0.2,
            zoom_range=0.2,
            horizontal_flip=True,
            validation_split=0.2)

        train = train_gen.flow_from_directory(
            base_dir,
            target_size=(224,224),
            batch_size= 32,
            class_mode='categorical',
            subset='training')

        validation_gen = train_gen.flow_from_directory(
            base_dir,
            target_size=(224, 224),
            batch_size= 32,
            class_mode='categorical',
            subset='validation')

Found 901 images belonging to 4 classes.
Found 224 images belonging to 4 classes.
```

VGG16

```
In [6]: from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
```

```
In [7]: # Adding the preprocessing layer to the front of vgg
        vgg = VGG16(include_top=False,weights='imagenet',input_shape=(224,224,3))

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [==============================] - 0s 0us/step
```

In [8]:
```python
# Train model with existing weights
for layer in vgg.layers:
  print(layer)
```

```
<keras.engine.input_layer.InputLayer object at 0x7f9b29053e80>
<keras.layers.convolutional.conv2d.Conv2D object at 0x7f9b29081900>
<keras.layers.convolutional.conv2d.Conv2D object at 0x7f9b29080f70>
<keras.layers.pooling.max_pooling2d.MaxPooling2D object at 0x7f9b29082c50>
<keras.layers.convolutional.conv2d.Conv2D object at 0x7f9b29083790>
<keras.layers.convolutional.conv2d.Conv2D object at 0x7f9b290838b0>
<keras.layers.pooling.max_pooling2d.MaxPooling2D object at 0x7f9b28508b80>
<keras.layers.convolutional.conv2d.Conv2D object at 0x7f9b290837f0>
<keras.layers.convolutional.conv2d.Conv2D object at 0x7f9b2850a440>
<keras.layers.convolutional.conv2d.Conv2D object at 0x7f9b28509720>
<keras.layers.pooling.max_pooling2d.MaxPooling2D object at 0x7f9b2850ba90>
<keras.layers.convolutional.conv2d.Conv2D object at 0x7f9b2850ba00>
<keras.layers.convolutional.conv2d.Conv2D object at 0x7f9b28509930>
<keras.layers.convolutional.conv2d.Conv2D object at 0x7f9b28525780>
<keras.layers.pooling.max_pooling2d.MaxPooling2D object at 0x7f9b28526830>
<keras.layers.convolutional.conv2d.Conv2D object at 0x7f9b28527220>
<keras.layers.convolutional.conv2d.Conv2D object at 0x7f9b28527490>
<keras.layers.convolutional.conv2d.Conv2D object at 0x7f9b285275b0>
<keras.layers.pooling.max_pooling2d.MaxPooling2D object at 0x7f9b28538970>
```

In [11]:
```python
# Train model with existing weights
for layer in vgg.layers:
  layer.trainable=False
```

In [12]:
```python
x = Flatten()(vgg.output)
```

In [13]:
```python
# output layer
prediction = Dense(4,activation='softmax')(x)
# Create Vgg16 model
model = Model(inputs=vgg.input,outputs=prediction)
```

In [14]:
```python
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten_1 (Flatten)         (None, 25088)             0

 dense (Dense)               (None, 4)                 100356

=================================================================
Total params: 14,815,044
Trainable params: 100,356
Non-trainable params: 14,714,688
```

In [15]:

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

In [16]:

```
model.fit_generator(train,validation_data=validation_gen,epochs=10,steps_per_epoch=len(train),
                    validation_steps=len(validation_gen))
```

```
<ipython-input-16-3b4b868b8d22>:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Mode
l.fit`, which supports generators.
  model.fit_generator(train,validation_data=validation_gen,epochs=10,steps_per_epoch=len(train),
Epoch 1/10
29/29 [==============================] - 35s 764ms/step - loss: 0.9922 - accuracy: 0.6471 - val_loss: 1.1540 - val_accuracy: 0.6473
Epoch 2/10
29/29 [==============================] - 26s 904ms/step - loss: 0.3346 - accuracy: 0.8746 - val_loss: 0.6467 - val_accuracy: 0.7902
Epoch 3/10
29/29 [==============================] - 21s 710ms/step - loss: 0.1826 - accuracy: 0.9445 - val_loss: 0.4057 - val_accuracy: 0.8839
Epoch 4/10
29/29 [==============================] - 20s 690ms/step - loss: 0.1305 - accuracy: 0.9689 - val_loss: 0.4452 - val_accuracy: 0.8571
Epoch 5/10
29/29 [==============================] - 20s 688ms/step - loss: 0.0989 - accuracy: 0.9811 - val_loss: 0.3957 - val_accuracy: 0.8705
Epoch 6/10
29/29 [==============================] - 21s 710ms/step - loss: 0.1014 - accuracy: 0.9778 - val_loss: 0.3264 - val_accuracy: 0.8795
Epoch 7/10
29/29 [==============================] - 21s 708ms/step - loss: 0.0893 - accuracy: 0.9811 - val_loss: 0.4530 - val_accuracy: 0.8304
Epoch 8/10
29/29 [==============================] - 21s 746ms/step - loss: 0.0851 - accuracy: 0.9867 - val_loss: 0.2498 - val_accuracy: 0.9241
Epoch 9/10
29/29 [==============================] - 21s 725ms/step - loss: 0.0628 - accuracy: 0.9900 - val_loss: 0.4094 - val_accuracy: 0.8393
Epoch 10/10
29/29 [==============================] - 20s 687ms/step - loss: 0.0526 - accuracy: 0.9900 - val_loss: 0.3020 - val_accuracy: 0.8973
```

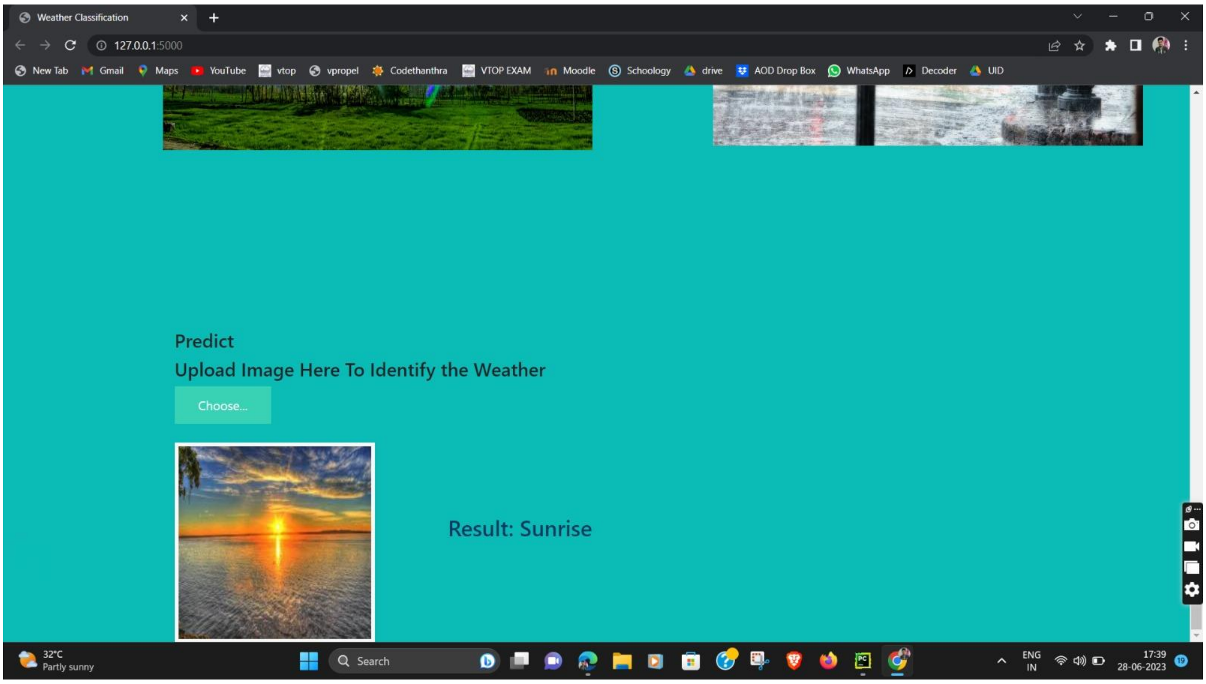
Out[16]:

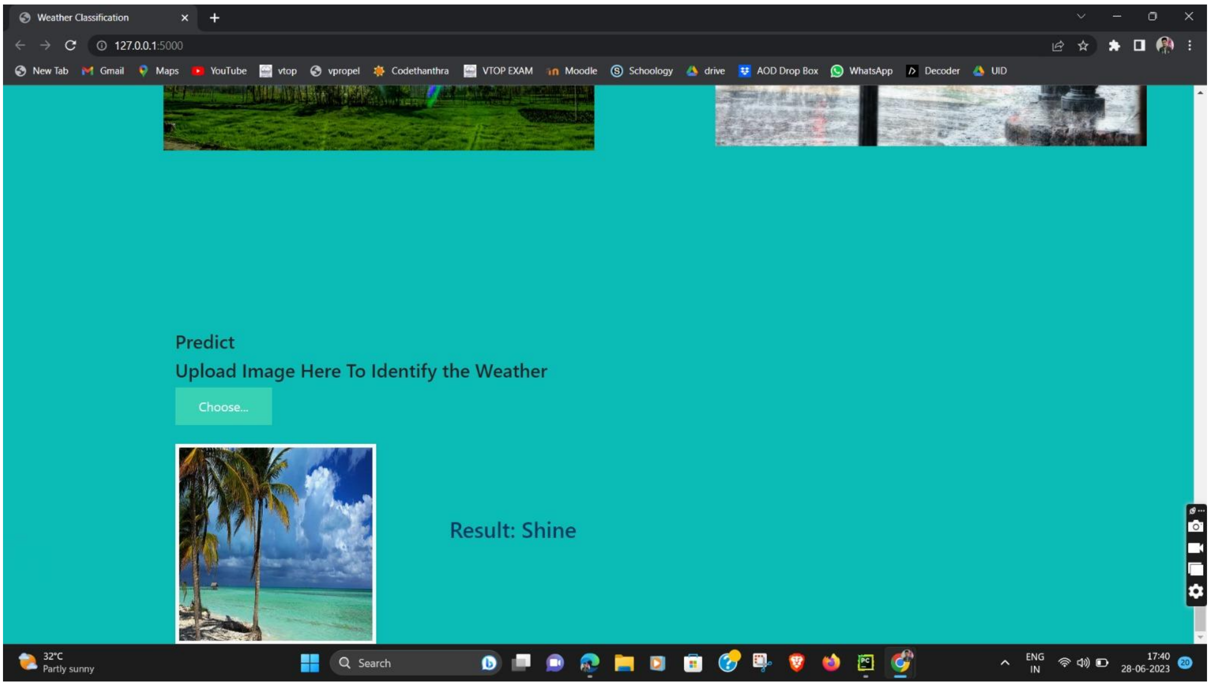
```
<keras.callbacks.History at 0x7f9b2857f430>
```

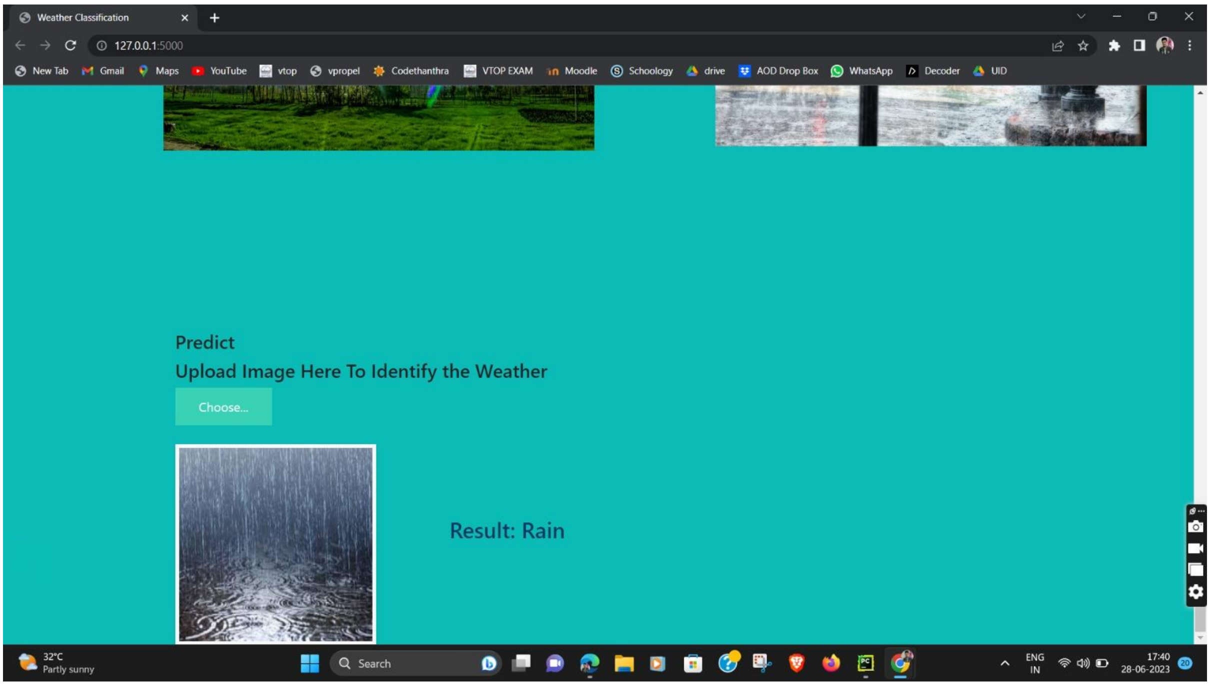
In [17]:

```
model.save("Automated_weather_classification.h5")
```

DEPLOYMENT USING FLASK







MODEL	LOSS	ACCURACY	VAL_LOSS	VAL_ACC
VGG16	0.052	0.99	0.30	0.89
RESNET50	1.12	0.53	1.13	0.48
INCEPTIONV3	0.14	0.95	0.53	0.88
XCEPTION	0.12	0.96	0.62	0.87

BY OBSERVING THE ABOVE TABLE.THE ACCURACY IS MORE FOR VGG16 MODEL WITH AN ACCURACY OF 0.99

7 ADVANTAGES & DISADVANTAGES:

Benefits of the proposed transfer learning-based automated weather classification method:

Increased Precision: Pre-trained models with learned rich features from large datasets are used in transfer learning. This prompts further developed precision in climate arrangement contrasted with beginning without any preparation.

Effective Model Education: Transfer learning eliminates the need for extensive training on weather-specific datasets by using pre-trained models. This altogether saves computational assets and preparing time.

Working with Limited Data: Even with limited labeled data, transfer learning makes it possible to train models effectively. It use the gained highlights from pre-preparing, which sums up well to weather patterns with less examples.

Speculation to New Atmospheric conditions: Models can adapt and generalize to new or unknown weather conditions thanks to transfer learning. The models can better deal with new weather patterns or unusual events by using the features they've learned.

Adaptability and flexibility: Images, satellite imagery, and readings from weather sensors are all examples of weather data that can be incorporated into the proposed solution. Weather conditions can be classified using a variety of data formats and sources thanks to this adaptability.

Problems and drawbacks of the proposed solution:

Change of Domain: Typically, pre-trained models are trained on general visual recognition tasks, which may differ from weather data's particular characteristics. Model performance can be affected by the domain shift between pre-training and weather classification, so careful adaptation techniques are required.

Dataset Predisposition: The availability of weather datasets that are both representative and of high quality is necessary for transfer learning to be effective. The trained model's accuracy and generalizability may be compromised if the weather dataset used for fine-tuning lacks diversity or is biased.

Modification Complexity: It can be hard to figure out the best setup for fine-tuning the pre-trained model, picking the right hyperparameters, and avoiding overfitting. To achieve the desired performance, skilled tuning and experimentation are required.

Interpretability: It is hard to understand how deep learning models, including transfer learning models, make decisions because they are frequently regarded as black boxes. It may be difficult to comprehend the model's reasoning behind weather classifications due to its lack of interpretability.

Asset Necessities: Pre-trained models can still be computationally expensive during fine-tuning and inference, particularly for large-scale models, despite transfer learning's ability to cut training time and resources needed.

Pre-trained Model Reliance: The availability and quality of pre-trained models are critical to the success of transfer learning. Pre-trained models that are suitable for a given task or domain may be limited, and their applicability may vary.

Deployment and Integration: There may be technical obstacles when incorporating transfer learning models into existing decision support or weather forecasting systems. For a successful deployment, compatibility, data integration, and seamless integration with other components must be addressed.

8 APPLICATIONS:

The proposed arrangement of computerized climate characterization utilizing move learning can be applied in different regions and ventures where climate data assumes a urgent part. The following are some key areas where this solution may be useful:

Weather Observation: Weather forecasting models' accuracy and dependability can be enhanced by automated weather classification. By grouping weather patterns from different information sources, for example, satellite symbolism or climate sensors, forecasters can make more educated expectations about future weather conditions and occasions.

Ecological Checking and Environment Studies: Understanding and monitoring environmental conditions and climate patterns can benefit from weather classification. Researchers can gain insight into long-term climate changes, environmental impacts, and the overall dynamics of weather systems by analyzing weather data over time.

Aeronautics and Transportation: Exact climate grouping is fundamental for the flying business to guarantee protected and effective flight activities. Flight routes and schedules can be impacted by weather conditions like fog, strong winds, or thunderstorms. Mechanized climate order can help carriers, air terminals, and aviation authority frameworks pursue informed choices and relieve gambles related with climate related interruptions.

Farming and farming: Weather conditions have a direct impact on crop growth, yields, and agricultural practices. Farmers can optimize irrigation, pest control, and planting schedules by classifying weather patterns. Accurate and timely information for agricultural decision-making can be facilitated by automated weather classification, ultimately enhancing crop management and productivity.

Energy from the Sun: Renewable energy systems like wind and solar power depend on the weather. Robotized climate characterization can help with anticipating sun based irradiance, wind speed, and other applicable elements. Energy providers and grid operators can use this information to optimize energy generation, storage, and distribution, resulting in more effective and long-lasting energy management.

9 CONCLUSION:

Taking everything into account, the proposed arrangement of mechanized climate characterization involving move learning offers critical benefits and amazing open doors for working on the exactness and proficiency of climate order frameworks. Through the use of pre-prepared models and tweaking methods, move gaining empowers the extraction of significant highlights from climate information and improves the speculation to new atmospheric conditions.

Be that as it may, a few difficulties and constraints should be thought of. Issues with integration and deployment, domain shift, bias in the dataset, fine-tuning complexity, interpretability, and the need for resources are just a few examples. Defeating these difficulties requires further innovative work in regions like information expansion, space transformation procedures, model improvement, interpretability techniques, and consistent joining with existing frameworks.

10 FUTURE SCOPE:

Several improvements based on transfer learning can be made in the future to further enhance automated weather classification. Some potential areas for growth are as follows:

Pre-trained weather-specific models are being developed: Making pre-prepared models explicitly intended for climate arrangement can improve the exhibition and versatility of the exchange learning approach. Large-scale weather-specific datasets with a wide range of weather conditions and geographic regions can be used to train these models.

Techniques for Advancing Data Augmentation: Research on clever information expansion procedures explicitly custom fitted for climate information can additionally expand the preparation dataset and work on model speculation. The diversity and representation of the training data can be improved through methods like weather data synthesis, transformations specific to a particular domain, and physical parameter manipulation.

Methods for Stable Domain Adaptation: It is still difficult to overcome the domain shift between the pre-training and weather classification tasks. Model performance in weather classification tasks can be improved by developing robust domain adaptation techniques that effectively align the feature distributions of the pre-training and target domains.

Extreme Weather Event Transfer Learning: Due to their rarity and severity, extreme weather events present unique challenges for classification. Accurate prediction and improved preparedness for extreme weather events like hurricanes, tornadoes, and heatwaves can be made possible by developing transfer learning methods that are able to effectively deal with them.

Data Integration across Multiple Modes: Climate characterization can profit from coordinating multi-modular information sources, including satellite symbolism, climate sensor readings, environment information, and literary meteorological forecasts. Creating move learning models that can actually use and incorporate assorted information modalities can give more thorough and exact climate characterizations.

11 BIBILOGRAPHY :

References:

1. M. Abrar, A. Sim, D. Shah, S. Khusro, A. Salam, Weather Prediction using Classification, Science International. 26, 2217-2223, https://www.researchgate.net/publication/272482887_Weather_Prediction_using_Classification, (2014)
2. A. John Arnfield, Climate classification, Encyclopedia Britannica, https://www.britannica.com/topic/classification-1703397, (17 Mar. 2016)
3. J. Hidalgo, R. Jougla, On the use of local weather types classification to improve climate understanding: An application on the urban climate of Toulouse, PLoS ONE 13(12): e0208138. https://doi.org/10.1371/journal.pone.0208138, (2018)
4. H. Beck, N. Zimmermann, T. McVicar, et al., Present and future Köppen-Geiger climate classification maps at 1-km resolution, Sci Data 5, 180214. https://doi.org/10.1038/sdata.2018.214, (2018).

APPENDIX A.

SOURCE CODE AND FLASK DEPLOYMENT:

GITHUB LINK:

https://github.com/rohithreddy999/Automated_weather_classification_using_transfer_learning