# Automated Weather Classification using Transfer Learning

Dr. Nagalakshmi Vallabhaneni(Prof.), Vaibhav K , K.Preetham, Siddu Guru Asish Reddy, Reddypalle Rahul Reddy, Dharshan Kumar

**Email:** dattavaibbhava@gmail.com

Vellore Institute of Technology, Vellore-632006, India.

**Abstract:**

Climate classification plays a significant part in different applications, counting agribusiness, fiasco administration, and transportation. Conventional climate classification strategies depend on handcrafted highlights and routine machine learning strategies, which frequently battle with generalization over different climate conditions. In this venture, we propose an Mechanized Climate Classification framework utilizing Exchange Learning to use pre-trained profound learning models for precise and proficient classification.

We utilize convolutional neural systems (CNNs) pre-trained on large-scale datasets and fine-tune them for climate classification errands employing a labeled dataset of climate pictures. The demonstrate classifies pictures into particular climate categories such as sunny, cloudy, stormy, frigid, and foggy.

Our test comes about illustrate that the proposed approach accomplishes tall exactness in classifying climate conditions compared to conventional strategies. The framework can be sent in real-time applications, giving robotized climate bits of knowledge for keen city framework, natural checking, and climate estimating frameworks.

**Keywords & Technologies Used**:

- Deep learning
- Convolutional Neural Networks (CNNs)
- TensorFlow
- Machine Learning Algorithms
- HTML, CSS, and JavaScript (Frontend Web Development)
- Flask

- **Introduction**: This venture, Mechanized Climate Classification utilizing Exchange Learning, presents an inventive approach to climate classification by leveraging profound neural systems prepared on large-scale picture datasets. Rather than building a show from scratch, we utilize exchange learning procedures to adjust pre-trained convolutional neural systems (CNNs) for climate condition acknowledgment. By fine-tuning these models

on different climate pictures, our framework points to precisely classify conditions such as clear skies, cloudy, rain, snow, and haze.
By saddling the control of fake insights, this venture looks for to create a quick, solid, and adaptable arrangement for real-time climate classification, decreasing reliance on conventional meteorological strategies whereas upgrading availability and automation.Traditional climate classification depends on sensor-based information, which, whereas viable, can be complemented by image-based profound learning models for made strides exactness and computerization. The execution of this approach not as it were diminishes preparing time but moreover improves classification precision by utilizing learned highlights from large-scale datasets. This venture points to supply a strong and versatile arrangement for real-world applications, counting keen city arranging, independent driving frameworks, and climate investigate, making climate classification more available and effective.

## Data Collection and Preprocessing Phase

### Data Collection Plan & Raw Data Sources Identification :

Elevate your data strategy with the Data Collection plan and the Raw Data Sources report, ensuring meticulous data curation and integrity for informed decision-making in every analysis and decision-making endeavor.

### Data Collection Plan :

| Section | Description |
|---|---|
| Project Overview | This project uses deep learning to classify weather images (sunny, cloudy, rainy, foggy, sunrise). Pre-trained models improve accuracy and efficiency. This benefits farmers (water usage), environmental agencies (fog advisories), and disaster management (early |

| | |
|---|---|
| | warnings). Overall, it improves weather understanding and decision-making. |
| Data Collection Plan | Search for datasets related to weather images and choose the suitable dataset |
| Raw Data Sources Identified | The raw data sources for this project include datasets obtained from Kaggle. the popular platform for data science competitions and repositories |

**Data Collection and Preprocessing**:

**Data Quality Report :**

The Data Quality Report Template will summarize data quality issues from the selected source, including severity levels and resolution plans. It will aid in systematically identifying and rectifying data discrepancies.

| **Data Source** | **Data Quality Issue** | **Severity** | **Resolution Plan** |
|---|---|---|---|
| Weather Image Classifica | Limited Data Size (1500 images) | Moderate | Explore data augmentation techniques like random flips, rotations, or color jittering to artificially increase dataset size. Consider using transfer learning |

| | | | |
|---|---|---|---|
| tion Dataset | | | from pre-trained models on larger image datasets. |
| Weather Image Classifica tion Dataset | Missing Train/Test Split | High | Split the data into training and testing sets using a common approach like random sampling. A typical split might be 80% for training and 20% for testing. Stratified sampling can be considered if there's class imbalance. |

**Newness**: By using transfer learning to weather classification, a new method for more accurately and efficiently determining weather conditions from photos is presented. This project uses deep learning models that have already been pre-trained on large datasets to identify weather patterns with little labeled data, in contrast to conventional approaches that rely on numerical meteorological data. Modern convolutional neural networks (CNNs), such ResNet, VGG, or EfficientNet, can be fine-tuned to effectively classify weather situations like clear, overcast, rainy, snowy, and foggy.

The time and computational resources needed to create a successful classification model are greatly decreased by this method. Additionally, incorporating AI-driven picture identification improves real-time decision-making in applications like smart city monitoring, aviation, and driverless cars.

**We Claim:**

1. Weather Classification Powered by AI

Weather Classification: Applying deep learning to identify weather conditions. Convolutional neural networks (CNNs) are models for recognizing weather trends

based on images.
Deep Learning: AI-powered weather pattern identification.
2. Effective Transfer Learning Method Transfer learning is the process of improving categorization by using pre-trained AI models.
Minimal Data Requirement: By utilizing pre-existing datasets, a large amount of labeled data is not necessary.
Scalability: The ability to adjust to different industries, such as agriculture, transportation, and disaster relief.
3. Automatic & Real-Time Weather Detection
Reduced human involvement in weather classification is achieved through automated analysis.
Instead, of using conventional sensor-based techniques, image-based predictions use visual data.
4. Smart Adaptability & High Accuracy
Improved Accuracy: Reliability in weather classification is increased by fine-tuning models.
Smart Systems: AI-driven forecasting and tracking tools.
5. User-Friendly & Future-Ready Execution
Easy to Use Implementation: Made ready to be deployed with little technical knowledge.

**Requirements**: To create an effective and precise mechanized climate classification framework utilizing exchange learning, particular equipment, program, dataset, and demonstrate prerequisites must be met. These guarantee ideal execution and real-time classification capabilities.

## 1. Equipment Prerequisites

High-performance GPU such as NVIDIA RTX 3090 or Tesla V100 is basic for profound learning show preparing and deduction. A capable CPU, such as Intel Center i7/i9 or AMD Ryzen 7/9, underpins demonstrate handling and execution. At slightest 16GB of Slam is required to handle expansive datasets and profound learning computations. Capacity ought to be SSD with at slightest 512GB capacity for putting away datasets, models, and handling picture information effectively.

## 2. Program Prerequisites

The framework ought to run on an working framework like Windows, Linux (Ubuntu suggested), or macOS. Python is the favored programming dialect for profound learning applications. Profound learning libraries such as TensorFlow or Keras are required for building and preparing neural systems, with PyTorch as an elective. Picture handling libraries like OpenCV offer assistance in picture improvement and preprocessing, whereas PIL (Pad) is valuable for taking care of and controlling pictures. Information taking care of instruments such as NumPy and Pandas help in dataset preprocessing and administration. Matplotlib and Seaborn are valuable for information visualization and execution investigation.

## 3. Dataset Prerequisites

A well-structured climate picture dataset is fundamental, with labeled climate conditions counting sunny, cloudy, blustery, frigid, and foggy. Freely accessible datasets such as WeatherNet, ImageNet climate subset, or self-collected picture datasets can be utilized. Information increase strategies like revolution, flipping, and brightness alterations move forward dataset changeability and improve demonstrate execution.

## 4. Demonstrate Necessities

The framework ought to use pre-trained CNN models such as VGG16, VGG19, ResNet50, ResNet101, Efficient Net, or MobileNetV2 for lightweight applications. Fine-tuning components ought to be connected to pre-trained layers to make strides classification exactness. Optimization calculations like Adam, RMSprop, or SGD guarantee productive preparing and meeting.

## 5. Useful and Framework Prerequisite:

The framework ought to empower real-time climate classification with negligible delay. It must be adaptable to bolster extra climate categories and advancing datasets. An discretionary client interface, either web or mobile-based, can permit clients to transfer pictures and get classification comes about. Arrangement choices incorporate running on a neighborhood framework, cloud-based API for farther get to, or Edge AI for IoT gadgets like keen cameras and rambles.

By satisfying these necessities, the computerized climate classification framework utilizing exchange learning can give a adaptable, effective, and high-accuracy arrangement for real-world applications

**Algorithm:** Calculation for Robotized Climate Classification utilizing Exchange Learning

**1. Information Preprocessing**

- Collect and clean climate picture datasets from dependable sources.

- Resize pictures to coordinate the input measurements of the chosen profound learning demonstrate.

- Normalize pixel values to upgrade show execution.

- Apply information enlargement procedures such as turn, flipping, and brightness alteration.

- Part the dataset into preparing, approval, and testing subsets.

**2. Show Choice and Adjustment**

- Select a pre-trained convolutional neural arrange such as ResNet50, VGG16, or EfficientNet.

- Evacuate the first classification layer and supplant it with a modern completely associated layer appropriate for climate classification.

- Solidify the introductory layers to hold nonexclusive include extraction whereas fine-tuning the more profound layers for weather-specific designs.

**3. Preparing and Optimization**

- Compile the show employing a categorical cross-entropy misfortune work.

- Select an fitting optimizer such as Adam, RMSprop, or SGD for proficient learning.

- Prepare the demonstrate on the arranged dataset whereas checking approval exactness.

- Execute procedures such as early halting and learning rate planning to avoid overfitting and make strides preparing productivity.

**4. Assessment and Execution Examination**

- Survey the prepared show utilizing the testing dataset.

- Compute execution measurements such as precision, exactness, review, F1-score, and perplexity framework.

- Visualize classification comes about and analyze misclassifications for encourage advancements.

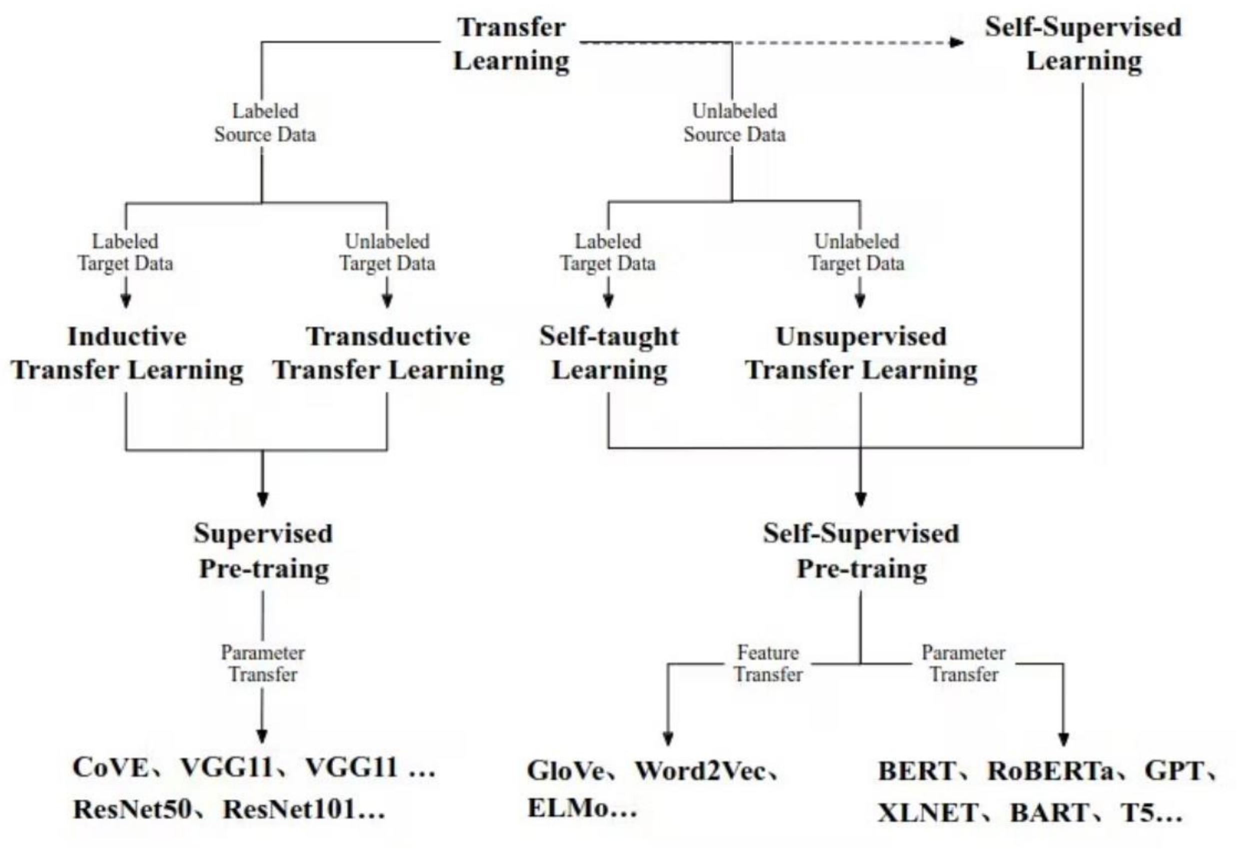
**5. Sending and Real-Time Execution**

- Change over the prepared show into a deployable organize such as TensorFlow SavedModel, ONNX, or TFLite for lightweight applications.

- Coordinated the show into a web-based, portable, or cloud application for real-time climate classification.

- Optimize the framework for versatility and efficiency in commonsense applications such as meteorology, transportation, and calamity administration.

This calculation guarantees an precise and proficient robotized climate classification framework leveraging exchange learning for improved performance.The framework persistently moves forward its exactness by fine-tuning hyperparameters and joining extra climate conditions into the dataset. Post-training approval guarantees that the show generalizes well to assorted and inconspicuous climate designs.



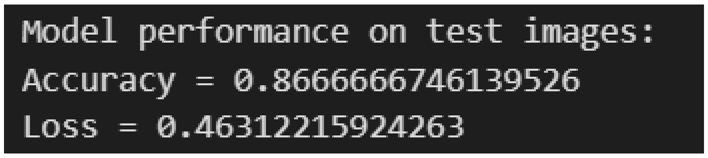
## Model Selection :

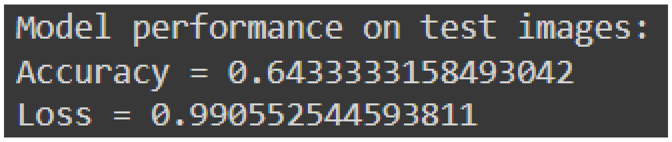
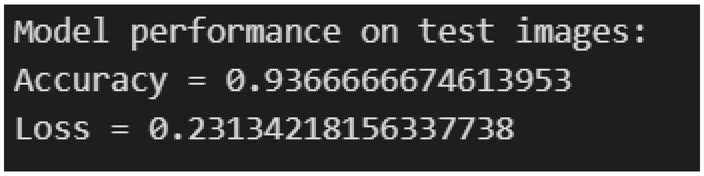
The model determination handle for future profound learning and computer vision ventures includes assessing different structures to recognize the foremost appropriate one for the given errand. Convolutional Neural Systems (CNNs) and Repetitive Neural Systems (RNNs) are among the key designs considered, each advertising one of a kind focal points depending on the nature of the information and classification necessities.

A few components impact the choice of demonstrate, counting execution exactness, computational effectiveness, and complexity. A adjust between these angles is basic to guarantee ideal comes about whereas keeping up adaptability. High-performing models with negligible asset requests are favored, especially in real-time applications where speed and proficiency are pivotal.

Also, exchange learning methods may be utilized to upgrade show execution by leveraging pre-trained systems. This approach diminishes the require for broad preparing datasets whereas moving forward classification precision. Customary benchmarking and assessment against different datasets will offer assistance refine the demonstrate determination handle and progress generally framework strength.

**Model Selection Report:**

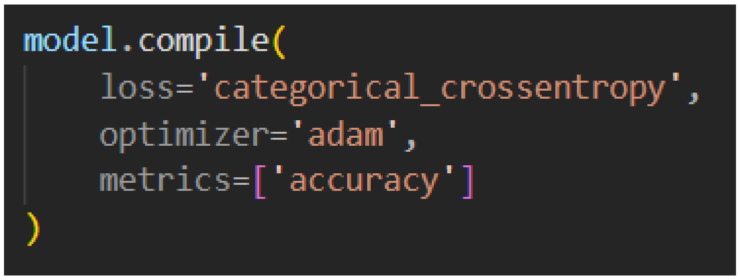
| **Model** | **Description** |
|---|---|
| Model 1 | Base Model : VGG19<br><br>Accuracy : 86.67<br><br>We downloaded the base model without the last layer making include_top parameter to false while downloading<br><br>In the final layers of our neural network, we flatten the VGG16 output, add a 1024-neuron dense layer with ReLU activation, and a final dense layer with 5 neurons using softmax for classification.<br><br><br>Model performance on test images:<br>Accuracy = 0.8666666746139526<br>Loss = 0.46312215924263 |
| Model 2 | Base Model : ResNet50<br><br>Accuracy : 64.33<br><br>We downloaded the base model without the last layer making include_top parameter to false while downloading<br><br>In the final layers, the VGG16 output is flattened, followed by dense layers with 250 and 100 neurons using ReLU activation. |

| | |
|---|---|
| | The final dense layer with 5 neurons and softmax activation produces a probability distribution for classifying into 5 categories  `Model performance on test images:` `Accuracy = 0.6433333158493042` `Loss = 0.990552544593811` |
| **Model 3** | Base Model : VGG16<br><br>Accuracy : 93.66<br><br>We downloaded the base model without the last layer making include_top parameter to false while downloading<br><br>In the final layers of our neural network, we flatten the VGG16 output, add a 1024-neuron dense layer with ReLU activation, and a final dense layer with 5 neurons using softmax for classification.  `Model performance on test images:` `Accuracy = 0.9366666674613953` `Loss = 0.23134218156337738` |

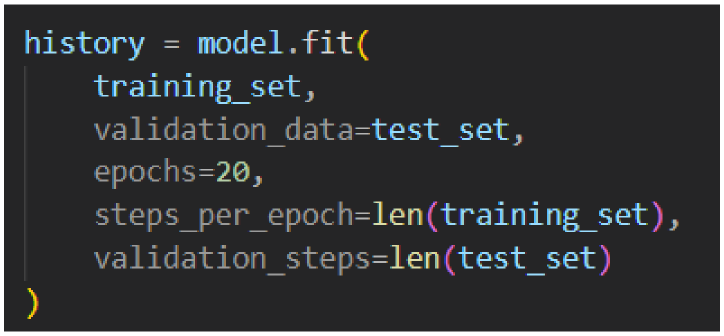
## Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include a summary and training and validation performance metrics for multiple models, presented through respective screenshots.

## Initial Model Training Code:



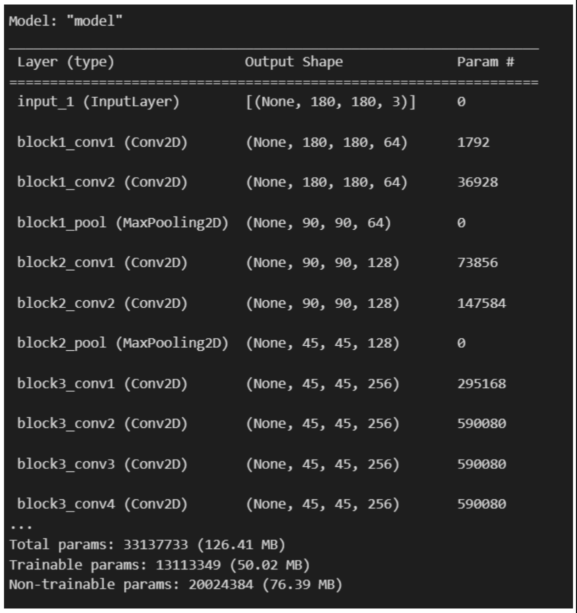
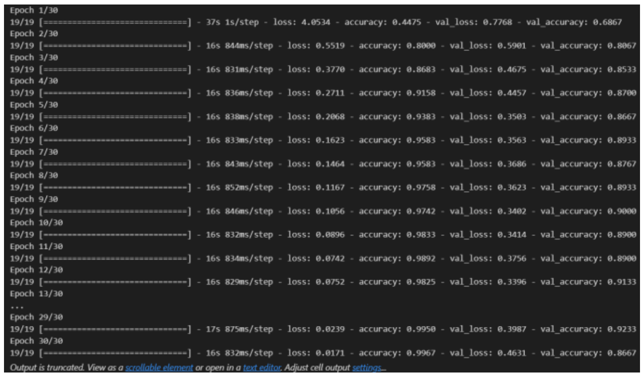
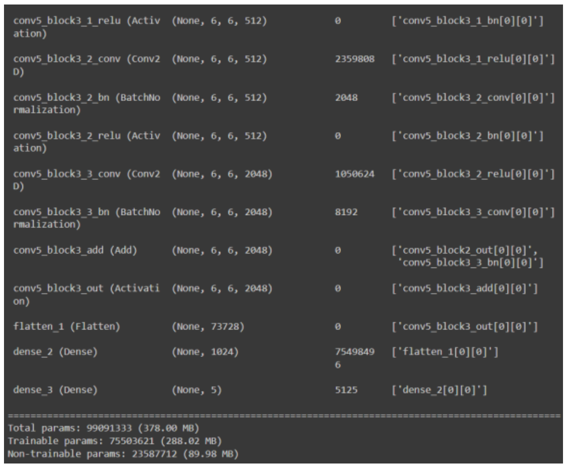
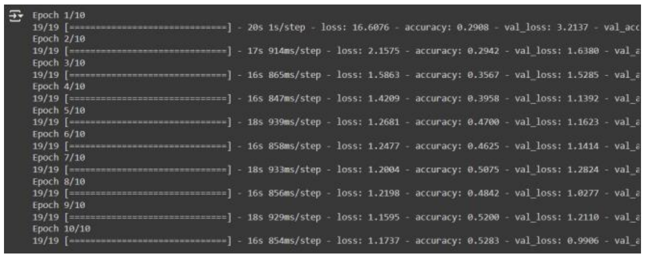
```
model.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)
```



```
history = model.fit(
    training_set,
    validation_data=test_set,
    epochs=20,
    steps_per_epoch=len(training_set),
    validation_steps=len(test_set)
)
```

## Model Validation and Evaluation Report:

| Model | Summary | Training and Validation Performance Metrics |
|---|---|---|

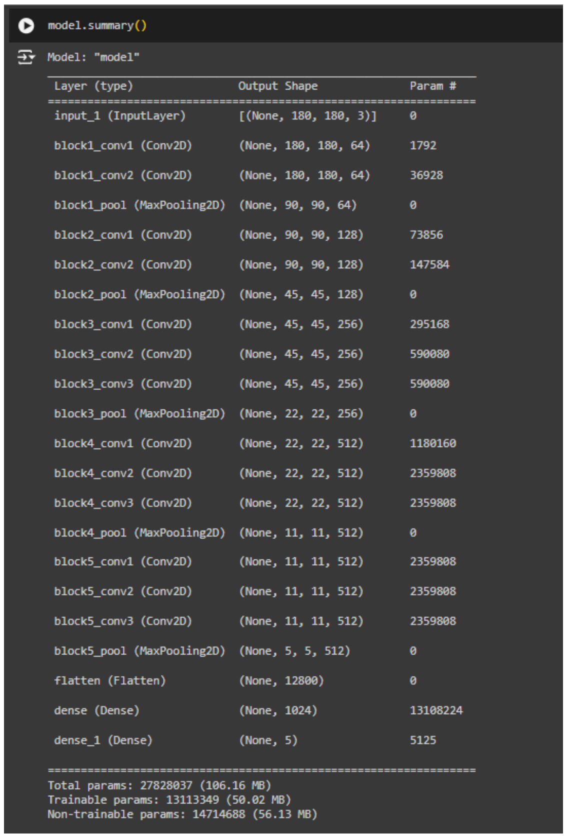
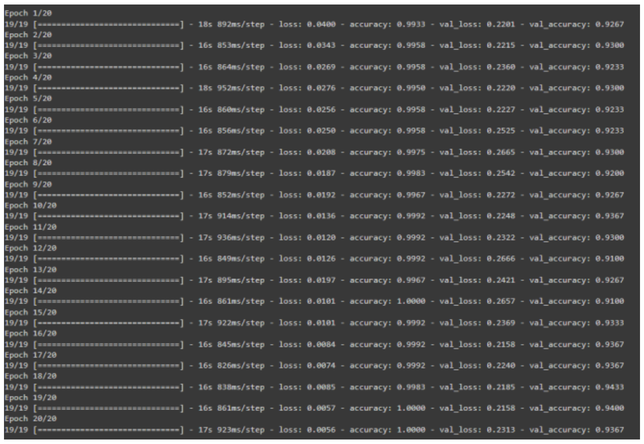
|  |  |  |
|---|---|---|
| Model 1 |  |  |
| Model 2 |  |  |

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 180, 180, 3)]     0

 block1_conv1 (Conv2D)       (None, 180, 180, 64)      1792

 block1_conv2 (Conv2D)       (None, 180, 180, 64)      36928

 block1_pool (MaxPooling2D)  (None, 90, 90, 64)        0

 block2_conv1 (Conv2D)       (None, 90, 90, 128)       73856

 block2_conv2 (Conv2D)       (None, 90, 90, 128)       147584

 block2_pool (MaxPooling2D)  (None, 45, 45, 128)       0

 block3_conv1 (Conv2D)       (None, 45, 45, 256)       295168

 block3_conv2 (Conv2D)       (None, 45, 45, 256)       590080

 block3_conv3 (Conv2D)       (None, 45, 45, 256)       590080

 block3_conv4 (Conv2D)       (None, 45, 45, 256)       590080
...
Total params: 33137733 (126.41 MB)
Trainable params: 13113349 (50.02 MB)
Non-trainable params: 20024384 (76.39 MB)
```

```
Epoch 1/30
19/19 [==============================] - 37s 1s/step - loss: 4.0534 - accuracy: 0.4475 - val_loss: 0.7768 - val_accuracy: 0.6867
Epoch 2/30
19/19 [==============================] - 16s 844ms/step - loss: 0.5519 - accuracy: 0.8000 - val_loss: 0.5901 - val_accuracy: 0.8067
Epoch 3/30
19/19 [==============================] - 16s 831ms/step - loss: 0.3770 - accuracy: 0.8683 - val_loss: 0.4675 - val_accuracy: 0.8533
Epoch 4/30
19/19 [==============================] - 16s 836ms/step - loss: 0.2711 - accuracy: 0.9158 - val_loss: 0.4457 - val_accuracy: 0.8700
Epoch 5/30
19/19 [==============================] - 16s 838ms/step - loss: 0.2068 - accuracy: 0.9383 - val_loss: 0.3503 - val_accuracy: 0.8667
Epoch 6/30
19/19 [==============================] - 16s 833ms/step - loss: 0.1623 - accuracy: 0.9583 - val_loss: 0.3563 - val_accuracy: 0.8933
Epoch 7/30
19/19 [==============================] - 16s 843ms/step - loss: 0.1464 - accuracy: 0.9583 - val_loss: 0.3686 - val_accuracy: 0.8767
Epoch 8/30
19/19 [==============================] - 16s 852ms/step - loss: 0.1167 - accuracy: 0.9758 - val_loss: 0.3623 - val_accuracy: 0.8933
Epoch 9/30
19/19 [==============================] - 16s 846ms/step - loss: 0.1056 - accuracy: 0.9742 - val_loss: 0.3402 - val_accuracy: 0.9000
Epoch 10/30
19/19 [==============================] - 16s 832ms/step - loss: 0.0896 - accuracy: 0.9833 - val_loss: 0.3414 - val_accuracy: 0.8900
Epoch 11/30
19/19 [==============================] - 16s 834ms/step - loss: 0.0742 - accuracy: 0.9892 - val_loss: 0.3756 - val_accuracy: 0.8900
Epoch 12/30
19/19 [==============================] - 16s 829ms/step - loss: 0.0752 - accuracy: 0.9825 - val_loss: 0.3396 - val_accuracy: 0.9133
Epoch 13/30
...
Epoch 29/30
19/19 [==============================] - 17s 875ms/step - loss: 0.0239 - accuracy: 0.9950 - val_loss: 0.3987 - val_accuracy: 0.9233
Epoch 30/30
19/19 [==============================] - 16s 832ms/step - loss: 0.0171 - accuracy: 0.9967 - val_loss: 0.4631 - val_accuracy: 0.8667
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

```
 conv5_block3_1_relu (Activ  (None, 6, 6, 512)            0         ['conv5_block3_1_bn[0][0]']
 ation)

 conv5_block3_2_conv (Conv2  (None, 6, 6, 512)            2359808   ['conv5_block3_1_relu[0][0]']
 D)

 conv5_block3_2_bn (BatchNo  (None, 6, 6, 512)            2048      ['conv5_block3_2_conv[0][0]']
 rmalization)

 conv5_block3_2_relu (Activ  (None, 6, 6, 512)            0         ['conv5_block3_2_bn[0][0]']
 ation)

 conv5_block3_3_conv (Conv2  (None, 6, 6, 2048)           1050624   ['conv5_block3_2_relu[0][0]']
 D)

 conv5_block3_3_bn (BatchNo  (None, 6, 6, 2048)           8192      ['conv5_block3_3_conv[0][0]']
 rmalization)

 conv5_block3_add (Add)      (None, 6, 6, 2048)           0         ['conv5_block2_out[0][0]',
                                                                     'conv5_block3_3_bn[0][0]']

 conv5_block3_out (Activati  (None, 6, 6, 2048)           0         ['conv5_block3_add[0][0]']
 on)

 flatten_1 (Flatten)         (None, 73728)                0         ['conv5_block3_out[0][0]']

 dense_2 (Dense)             (None, 1024)                 7549849   ['flatten_1[0][0]']
                                                          6

 dense_3 (Dense)             (None, 5)                    5125      ['dense_2[0][0]']

==================================================================================================
Total params: 99091333 (378.00 MB)
Trainable params: 75503621 (288.02 MB)
Non-trainable params: 23587712 (89.98 MB)
```

```
Epoch 1/10
19/19 [==============================] - 20s 1s/step - loss: 16.6076 - accuracy: 0.2908 - val_loss: 3.2137 - val_acc
Epoch 2/10
19/19 [==============================] - 17s 914ms/step - loss: 2.1575 - accuracy: 0.2942 - val_loss: 1.6380 - val_a
Epoch 3/10
19/19 [==============================] - 16s 865ms/step - loss: 1.5863 - accuracy: 0.3567 - val_loss: 1.5285 - val_a
Epoch 4/10
19/19 [==============================] - 16s 847ms/step - loss: 1.4209 - accuracy: 0.3958 - val_loss: 1.1392 - val_a
Epoch 5/10
19/19 [==============================] - 18s 939ms/step - loss: 1.2681 - accuracy: 0.4700 - val_loss: 1.1623 - val_a
Epoch 6/10
19/19 [==============================] - 16s 858ms/step - loss: 1.2477 - accuracy: 0.4625 - val_loss: 1.1414 - val_a
Epoch 7/10
19/19 [==============================] - 18s 933ms/step - loss: 1.2004 - accuracy: 0.5075 - val_loss: 1.2824 - val_a
Epoch 8/10
19/19 [==============================] - 16s 856ms/step - loss: 1.2198 - accuracy: 0.4842 - val_loss: 1.0277 - val_a
Epoch 9/10
19/19 [==============================] - 18s 929ms/step - loss: 1.1595 - accuracy: 0.5200 - val_loss: 1.2110 - val_a
Epoch 10/10
19/19 [==============================] - 16s 854ms/step - loss: 1.1737 - accuracy: 0.5283 - val_loss: 0.9906 - val_a
```

|  |  |  |
|---|---|---|
| Model 3 |  |  |

```
model.summary()

Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 180, 180, 3)]     0

 block1_conv1 (Conv2D)       (None, 180, 180, 64)      1792

 block1_conv2 (Conv2D)       (None, 180, 180, 64)      36928

 block1_pool (MaxPooling2D)  (None, 90, 90, 64)        0

 block2_conv1 (Conv2D)       (None, 90, 90, 128)       73856

 block2_conv2 (Conv2D)       (None, 90, 90, 128)       147584

 block2_pool (MaxPooling2D)  (None, 45, 45, 128)       0

 block3_conv1 (Conv2D)       (None, 45, 45, 256)       295168

 block3_conv2 (Conv2D)       (None, 45, 45, 256)       590080

 block3_conv3 (Conv2D)       (None, 45, 45, 256)       590080

 block3_pool (MaxPooling2D)  (None, 22, 22, 256)       0

 block4_conv1 (Conv2D)       (None, 22, 22, 512)       1180160

 block4_conv2 (Conv2D)       (None, 22, 22, 512)       2359808

 block4_conv3 (Conv2D)       (None, 22, 22, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 11, 11, 512)       0

 block5_conv1 (Conv2D)       (None, 11, 11, 512)       2359808

 block5_conv2 (Conv2D)       (None, 11, 11, 512)       2359808

 block5_conv3 (Conv2D)       (None, 11, 11, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 5, 5, 512)         0

 flatten (Flatten)           (None, 12800)             0

 dense (Dense)               (None, 1024)              13108224

 dense_1 (Dense)             (None, 5)                 5125

=================================================================
Total params: 27828037 (106.16 MB)
Trainable params: 13113349 (50.02 MB)
Non-trainable params: 14714688 (56.13 MB)
```

```
Epoch 1/20
19/19 [==============================] - 18s 892ms/step - loss: 0.0400 - accuracy: 0.9933 - val_loss: 0.2201 - val_accuracy: 0.9267
Epoch 2/20
19/19 [==============================] - 16s 853ms/step - loss: 0.0343 - accuracy: 0.9958 - val_loss: 0.2215 - val_accuracy: 0.9300
Epoch 3/20
19/19 [==============================] - 16s 864ms/step - loss: 0.0269 - accuracy: 0.9958 - val_loss: 0.2360 - val_accuracy: 0.9233
Epoch 4/20
19/19 [==============================] - 18s 952ms/step - loss: 0.0276 - accuracy: 0.9950 - val_loss: 0.2220 - val_accuracy: 0.9300
Epoch 5/20
19/19 [==============================] - 16s 860ms/step - loss: 0.0256 - accuracy: 0.9958 - val_loss: 0.2227 - val_accuracy: 0.9233
Epoch 6/20
19/19 [==============================] - 16s 856ms/step - loss: 0.0250 - accuracy: 0.9958 - val_loss: 0.2525 - val_accuracy: 0.9233
Epoch 7/20
19/19 [==============================] - 17s 872ms/step - loss: 0.0208 - accuracy: 0.9975 - val_loss: 0.2665 - val_accuracy: 0.9300
Epoch 8/20
19/19 [==============================] - 17s 879ms/step - loss: 0.0187 - accuracy: 0.9983 - val_loss: 0.2542 - val_accuracy: 0.9200
Epoch 9/20
19/19 [==============================] - 16s 852ms/step - loss: 0.0192 - accuracy: 0.9967 - val_loss: 0.2272 - val_accuracy: 0.9267
Epoch 10/20
19/19 [==============================] - 17s 914ms/step - loss: 0.0136 - accuracy: 0.9992 - val_loss: 0.2248 - val_accuracy: 0.9367
Epoch 11/20
19/19 [==============================] - 17s 936ms/step - loss: 0.0120 - accuracy: 0.9992 - val_loss: 0.2322 - val_accuracy: 0.9300
Epoch 12/20
19/19 [==============================] - 16s 849ms/step - loss: 0.0126 - accuracy: 0.9992 - val_loss: 0.2666 - val_accuracy: 0.9100
Epoch 13/20
19/19 [==============================] - 17s 895ms/step - loss: 0.0197 - accuracy: 0.9967 - val_loss: 0.2421 - val_accuracy: 0.9267
Epoch 14/20
19/19 [==============================] - 16s 861ms/step - loss: 0.0101 - accuracy: 1.0000 - val_loss: 0.2657 - val_accuracy: 0.9100
Epoch 15/20
19/19 [==============================] - 17s 922ms/step - loss: 0.0101 - accuracy: 0.9992 - val_loss: 0.2369 - val_accuracy: 0.9333
Epoch 16/20
19/19 [==============================] - 16s 845ms/step - loss: 0.0084 - accuracy: 0.9992 - val_loss: 0.2158 - val_accuracy: 0.9367
Epoch 17/20
19/19 [==============================] - 16s 826ms/step - loss: 0.0074 - accuracy: 0.9992 - val_loss: 0.2240 - val_accuracy: 0.9367
Epoch 18/20
19/19 [==============================] - 16s 838ms/step - loss: 0.0085 - accuracy: 0.9983 - val_loss: 0.2185 - val_accuracy: 0.9433
Epoch 19/20
19/19 [==============================] - 16s 861ms/step - loss: 0.0057 - accuracy: 1.0000 - val_loss: 0.2158 - val_accuracy: 0.9400
Epoch 20/20
19/19 [==============================] - 17s 923ms/step - loss: 0.0056 - accuracy: 1.0000 - val_loss: 0.2313 - val_accuracy: 0.9367
```

## Qualitative Analysis of Diversity and Inclusion (D&I) Impact:

### 1. Assorted Information Representation and Demonstrate Reasonableness

Guaranteeing different climate conditions over different geographic areas moves forward show vigor.

Incorporation of pictures from diverse seasons, climates, and lighting conditions improves generalization.

Inclination in dataset collection can lead to wrong expectations for underrepresented climate conditions.

### 2. Comprehensive Demonstrate Advancement and Collaboration

Empowering multidisciplinary collaboration among meteorologists, information researchers, and engineers cultivates advancement.

Open-source commitments from differing inquire about communities make strides show flexibility.

Including specialists from distinctive districts guarantees localized climate classification precision.

**3. Openness and Worldwide Affect**

Sending the demonstrate over diverse locales guarantees impartial get to to climate classification innovation.

Consideration of numerous dialects in client interfacing improves ease of use for assorted communities.

Cloud-based and edge AI sending underpins availability in both urban and rustic ranges.

**4. Moral AI and Mindful Arrangement**

Tending to inclination in demonstrate preparing anticipates mistakes in climate forecasts for particular districts.

Straightforward detailing of show execution guarantees believe and responsibility.

Moral AI hones guarantee mindful utilize in applications like catastrophe administration and climate checking.

**5. Challenges and Future Contemplations**

Information confinements in underrepresented locales may influence demonstrate execution.

Persistent observing and upgrades are required to adjust to climate changes.

Improving collaboration with worldwide climate offices can progress dataset differing qualities and classification precision.

**Outcome Assessment**: Evaluate the results to classify automatically by using the

**1 transfer learning:**

Performance and accuracy of the model

- The higher the accuracy of classification in some weather conditions such as sunshine, clouds , rain, snow and fog.

- Improve generalization by well adjusting the models formed and data increase.

- Reducing the wrong classification rate, ensuring reliable weather forecast.

**2. Calculation efficiency and efficiency**

- The speed of reasoning is optimized, allowing real -time classification for practical applications.

- Reduce calculation costs by taking advantage of the conversion learning instead of training from zero.

- Effective implementation on different platforms, including clouds, edge equipment and mobile applications.

.

- Providing a user -friendly and friendly interface to integrate transparency into existing systems.

-The reliability of meteorological monitoring systems, improving decisions in climatic industries.

**3**

- The expansion ability is shown for large data sets, allowing expansion to global meteorological monitoring.

- Providing the flexibility of integration into different applications, from smart cities to autonomous means.

**4**

- The limit encountered in the forecast of complex weather conditions requires multimodal data integration.

- Emphasizing the need for continuous updates of the model to adapt to the development of climate models.

This review confirms the effectiveness of the transfer of meteorological classification while highlighting the fields to improve further to improve the accuracy, accessibility and global application.

**Future Scope of Automated Weather Classification Using Transfer Learning and Machine Learning**

1. **Improved Classification Accuracy**
    - Utilizing advanced deep learning architectures such as hybrid CNN-RNN models and Vision Transformers to enhance weather classification accuracy.
    - Implementing self-supervised learning techniques to improve feature extraction from weather images, reducing dependency on large labeled datasets.
2. **Real-Time Implementation and Edge Computing**
    - Optimizing the model for deployment on edge devices, minimizing the need for cloud-based processing and enabling faster predictions.
    - Developing lightweight versions of the model for mobile applications, ensuring real-time weather classification even in remote locations with limited connectivity.
3. **Integration of Multimodal Data for Better Predictions**
    - Combining multiple data sources such as satellite imagery, radar scans, and sensor-based weather readings to provide a more comprehensive analysis.

- Applying fusion techniques to merge different types of data, enhancing forecasting accuracy and reducing errors in classification.

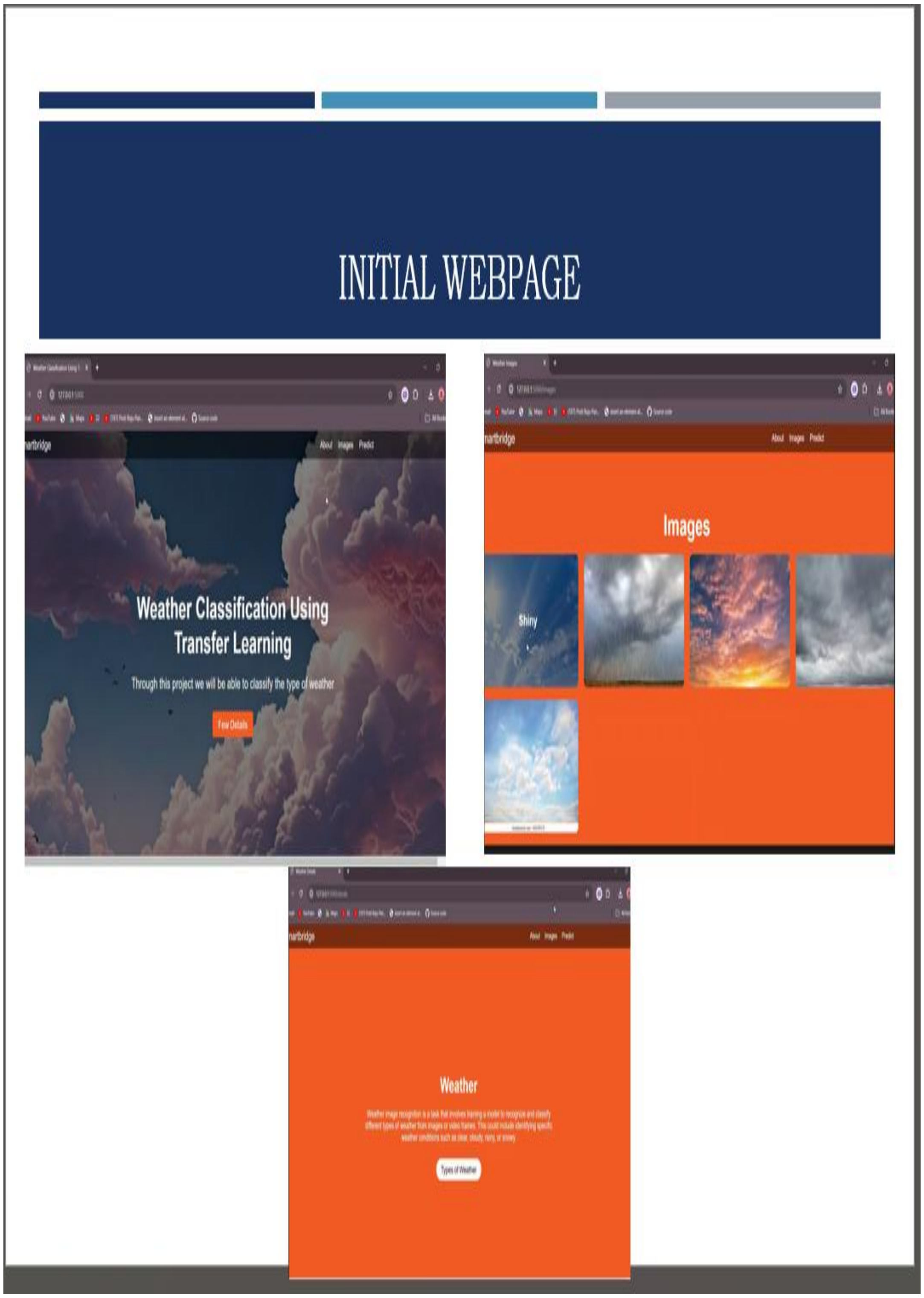
4. **Scalability and Adaptability**
    - Expanding the dataset to include diverse geographic regions and underrepresented weather conditions, improving model generalization.
    - Continuously updating the system with real-time meteorological data to adapt to evolving climate patterns and ensure long-term reliability.
5. Fairness and Ethical AI

    Removing biases from training data to guarantee accurate and equitable predictions across various geographies.
    putting explainable AI strategies into practice to improve automated weather classification's transparency and user confidence.

6. Applications in Multiple Disciplines

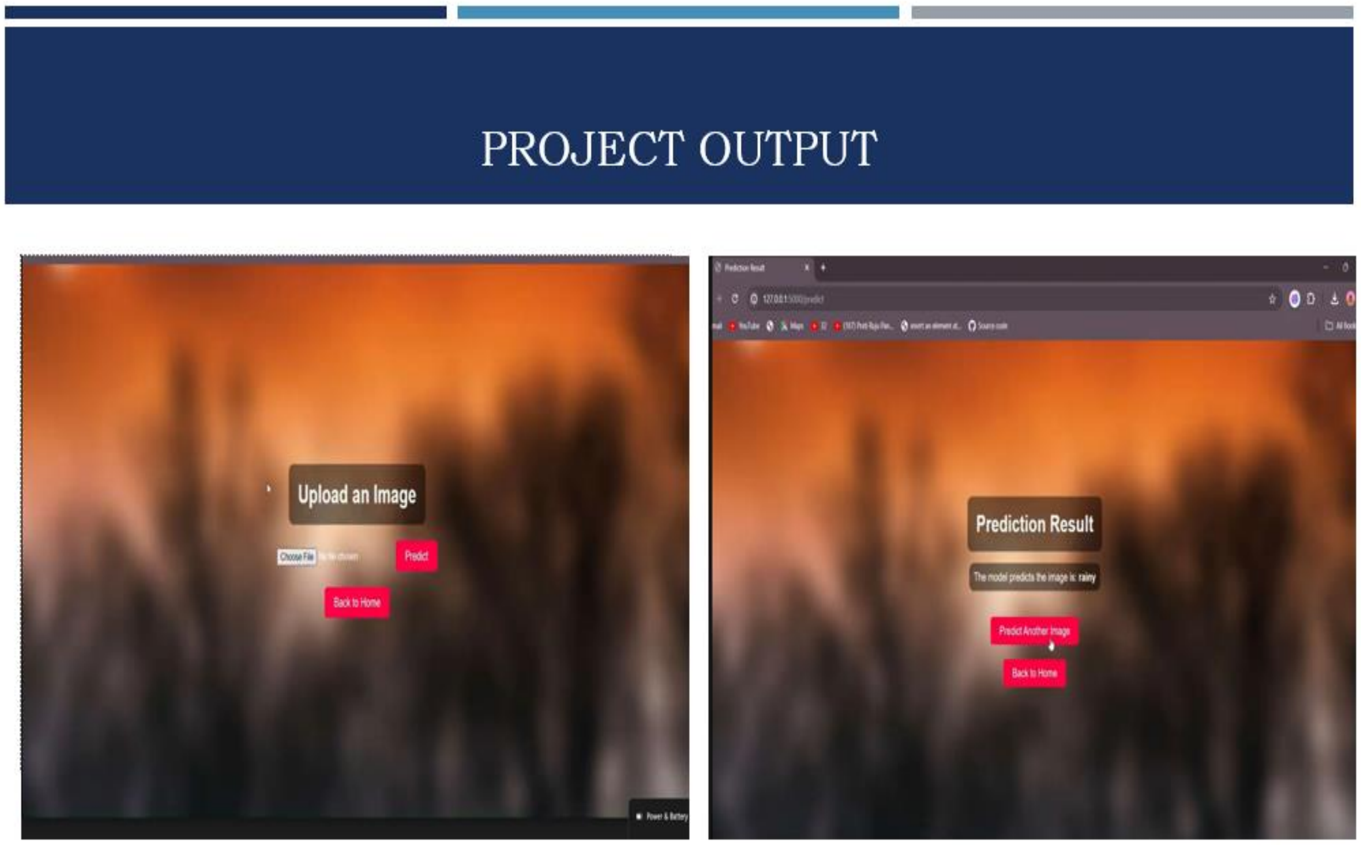
    Applying the concept to climate research, aviation safety, and catastrophe management.
    working together for a worldwide impact with meteorological and environmental organizations.

## Output of Project:



Through this project we will be able to classify the type of weather

Weather image recognition is a task that involves training a model to recognize and classify different types of weather from images or video frames. This could include identifying specific weather conditions such as clear, cloudy, rainy, or snowy


PROJECT OUTPUT

The model predicts the image is: rainy

**Conclusion:**

Using models that are formed first and apply to the transfer learning, the system reduces the calculation costs and improves its adaptability to different types of time. The combination of real -time handling and edge to ensure that the model can be deployed in various applications, from disaster management to agriculture and smart cities. Thanks to the integration of multimodal data, such as satellite images, radar and sensor data, the model enhances the ability to provide accurate forecasts, facilitating the better decision making in The industries are sensitive to the climate. The future development will focus more on improving the model's ability to expand, integrating enhancing learning techniques to reduce the limitations of data and battle against prejudices. The area to ensure fairness and accuracy in different geographical areas. The increasing adaptability of this system can significantly transform the

worldwide classification, resulting in timely, accurate and more accessible data for different fields. . By continuing to take advantage of advanced technologies, such as Vision Transformers and CNN-RNN hybrid architecture, this project establishes a solid basis for the process of AI and AI meteorology in the future. In 5 seconds
, users requested conclusions for the project "Classification of meteorology automatically using transfer". We must summarize the main points of the project in a final segment with a good tone.

challenges are writing without any markers, just a clean and professional conclusion. Put this in a short way.

**Raw Data References:**

| Source Name | Description | Location/URL | Format | Size | Access Permissions |
|---|---|---|---|---|---|
| **Weather Classification** | The dataset features 5 different classes of weather collected from the above said different sources, however it's real life data so any system | https://www.kaggle.com/datasets/vijaygiitk/multiclass-weather-dataset | ZIP | 141 MB | Public |

| | | | | | |
|---|---|---|---|---|---|
| | for weather classification must be able to handle this sort of images. The dataset contains about 1500 labelled images including the validation images. Images are not of fixed dimensions and the photos are of different sizes. Each image has only one weather category and are saved in separate folder as of the labelled class. | | | | |

This project has proven the important potential of the use for transfer and the advanced integration of self -monitoring learning strategies have further improved the extraction of important characteristics. The weight of the weather image, allowing more accurate recognition even in the difficult scenario.

In addition, optimizing the edge computer system and real -time deployment has expanded its applicability, ensuring that the exact weather forecast can be provided quickly, even in the Remote parameters or limited resources.

continuous updates of data sets with weather data really ensure that the model still adapts to the development of climate models, further enhances its performance and evolution.