Project Report

Weather Classification Using Teachable Machine and TensorFlow Keras

Introduction:

The objective of this project is to classify weather conditions using an image classification model trained with Google's Teachable Machine. The model is implemented using TensorFlow's Keras library and is capable of recognizing different weather conditions based on input images. The model is loaded and used to predict weather categories from test images.

Tools and Technologies Used:

- Programming Language: Python
- Libraries: TensorFlow Keras, NumPy, PIL (Pillow), ImageOps
- Hardware: Standard computing system with GPU/CPU support for TensorFlow
- Dataset: Weather condition images (e.g., sunrise, cloudy, rainy, etc.)
- Dataset Source: https://www.kaggle.com/datasets/vijaygiitk/multiclass-weather-dataset
- Teachable Machine: Used for training the classification model

Model Implementation:

Step 1: Loading the Pre-trained Model

- The trained model is stored as keras_model.h5 and is loaded using tensorflow.keras.models.load_model. Since the DepthwiseConv2D layer may contain the groups argument, a custom function custom_depthwise_conv2d is defined to handle its loading properly.

```
from tensorflow.keras.models import load_model
from tensorflow.keras.layers import DepthwiseConv2D
def custom_depthwise_conv2d(*args, **kwargs):
    if 'groups' in kwargs:
        del kwargs['groups']  # Remove the groups argument
    return DepthwiseConv2D(*args, **kwargs)

model = load_model("D:\\svasti vector solution\\Weather
Classification_image_model\\keras_model.h5", compile=False,
                   custom_objects={'DepthwiseConv2D': custom_depthwise_conv2d})
```

Step 2: Loading Class Labels

```
class_names = open("D:\\svasti vector solution\\Weather Classification_image_model\\labels.txt",
"r").readlines()
```

Step 3: Preprocessing Input Images

- To ensure consistency with the model's input format, images are preprocessed by resizing to 224x224 pixels, converting to a numpy array, and normalizing pixel values.

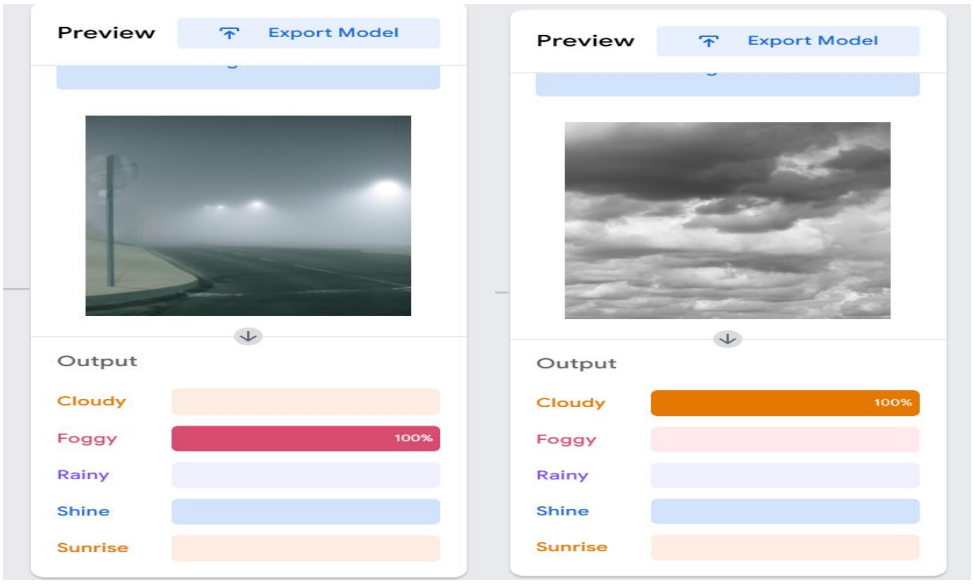
```
from PIL import Image, ImageOps
```

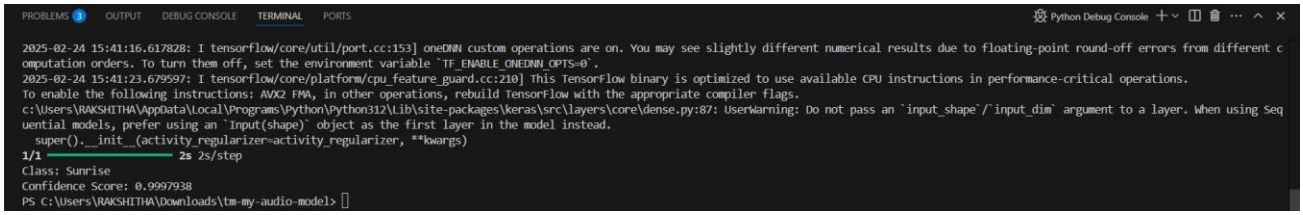
```
import numpy as np

np.set_printoptions(suppress=True)

# Create an array of the right shape

data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)

# Load and preprocess the image

image = Image.open("D:\\svasti vector solution\\Weather
Classification_image_model\\dataset\\sunrise\\sunrise105.jpg").convert("RGB")

size = (224, 224)

image = ImageOps.fit(image, size, Image.Resampling.LANCZOS)

image_array = np.asarray(image)

normalized_image_array = (image_array.astype(np.float32) / 127.5) - 1

# Load the image into the array

data[0] = normalized_image_array
```

Step 3: Making Predictions

```
prediction = model.predict(data)

index = np.argmax(prediction)

class_name = class_names[index]

confidence_score = prediction[0][index]

print("Class:", class_name[2:], end="")  # Remove leading newline

print("Confidence Score:", confidence_score)
```

Results:





Code: https://drive.google.com/file/d/1LuSACioU9LWxFlfBfUBS9k1Lred3PR1C/view?usp=drive_link

Conclusion:

This project demonstrates the effective use of Teachable Machine for training an image classification model and deploying it with TensorFlow Keras. The approach can be extended for real-time weather classification applications by integrating it with web-based or IoT systems.

Future Enhancements:

- Improve dataset diversity for better generalization.
- Optimize model for real-time performance.
- Deploy model as a web service for remote accessibility.
- Integrate with mobile applications for on-the-go weather classification.

References:

- TensorFlow Documentation: https://www.tensorflow.org
- Google Teachable Machine: https://teachablemachine.withgoogle.com

Web-Based Weather Classification System

Introduction:

In this project, we developed a web-based weather classification system using Teachable Machine and TensorFlow.js. The system utilizes a pre-trained image classification model to recognize different weather conditions from a webcam feed. The categories include:

- Cloudy
- Foggy
- Rainy
- Shine
- Sunrise

The primary goal of this project is to provide a real-time, accessible, and user-friendly weather classification tool using machine learning.

Dataset Source: https://www.kaggle.com/datasets/vijaygiitk/multiclass-weather-dataset

Objectives:

- Implement an interactive web-based weather classification system.
- Utilize **Teachable Machine** for model training and **TensorFlow.js** for deployment.
- Enable real-time classification using a webcam feed.
- Provide a smooth and user-friendly UI with responsive design.

Technologies Used:

Frontend:

- HTML: Structure of the webpage.
- CSS: Styling and UI enhancements.
- JavaScript: Handling user interactions and model inference.

Machine Learning & Model Deployment:

- Teachable Machine (Google): Pre-training the image classification model.
- TensorFlow.js: Loading and running the model in the browser.

System Architecture:

Model Training (Teachable Machine):

- Data Collection: Images were collected for five weather categories (Cloudy, Foggy, Rainy, Shine, Sunrise).
- Model Training: The dataset was uploaded to Teachable Machine, and a classification model was trained.
- Model Export: The trained model was exported as model.json and metadata.json.

Web Application Workflow:

- Model Loading: The model is loaded from the local server (http://localhost:8000/).
- Webcam Activation: Users can start and stop the webcam feed.
- Image Processing: The model processes frames in real time and classifies them into weather categories.
- Prediction Display: The classified label and confidence score are displayed dynamically.

Implementation Details:

Step 1: User Interface (UI)

- Dark-themed UI: Uses CSS for a visually appealing layout.
- Start & Stop Buttons: Users can control the webcam feed.
- Loading Spinner: Indicates when the model is being loaded.
- Real-time Display: Webcam feed and prediction results are updated dynamically.

Step 2: JavaScript Code Breakdown

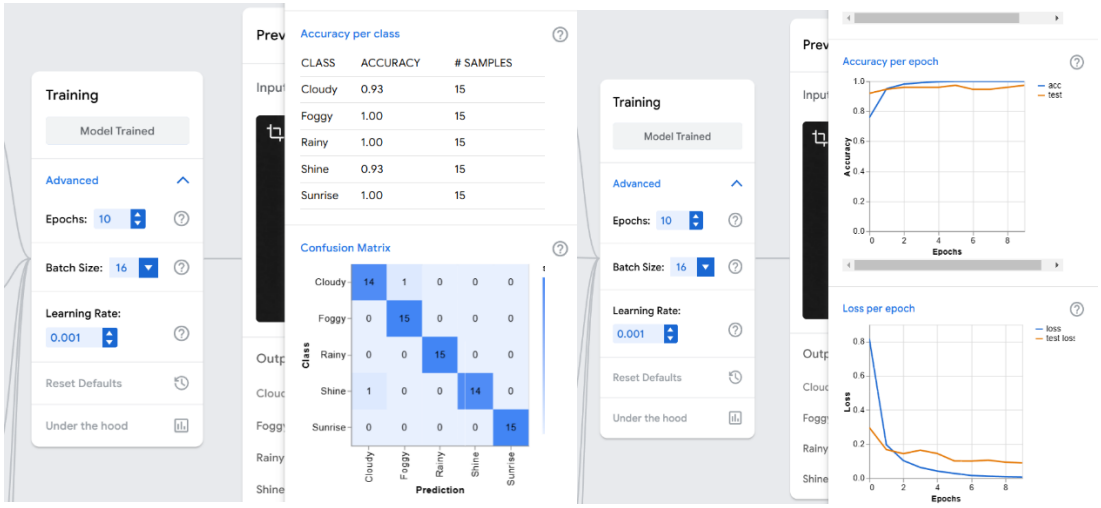
- Model Loading: The model is loaded asynchronously using tmImage.load(modelURL, metadataURL).
- Webcam Setup: The webcam is initialized with tmImage.Webcam(200, 200, flip).

Step 3: Real-time Prediction:

```
async function predict() {
   const prediction = await model.predict(webcam.canvas);
   for (let i = 0; i < maxPredictions; i++) {
      const classPrediction = prediction[i].className + ": " + prediction[i].probability.toFixed(2);
      labelContainer.childNodes[i].innerHTML = classPrediction;
   }
}
```

Start & Stop Functionality:

```
function stop() {
   isRunning = false;
   webcam.stop();
   document.getElementById("stopButton").style.display = 'none';
   document.getElementById("startButton").style.display = 'inline-block';
}
```



Results:

- The system successfully classifies weather conditions in real-time.
- The UI is intuitive and allows users to control the classification process.
- The model runs efficiently in a web browser without the need for external servers.
- Predictions are displayed with confidence scores, allowing users to assess accuracy.

Code:

https://drive.google.com/file/d/1kKNPJs9YYlARwwHa7bixHdMK7pQZAuxF/view?usp=drive_link

Conclusion:

This project successfully demonstrates a **real-time, web-based weather classification system** using **Teachable Machine and TensorFlow.js**. It provides an intuitive and efficient way to classify weather conditions directly from a webcam feed. Future enhancements can further improve the system's accuracy, usability, and application scope.

Web-Based Real-Time Pose Estimation Model

Abstract:

The Web-Based Real-Time Pose Estimation Model is an interactive application that utilizes machine learning and computer vision techniques to recognize and analyze human poses through a web interface. The model is built using Teachable Machine's Pose Model and TensorFlow.js, allowing real-time pose classification directly in the browser. This project aims to provide an accessible and efficient solution for applications such as fitness tracking, gesture recognition, and human-computer interaction.

Introduction:

Pose estimation is a crucial component in various fields, including sports analytics, healthcare, and security. Traditional pose estimation methods require high computational power and specialized hardware. This web-based model overcomes these limitations by leveraging browser-based machine learning using TensorFlow.js and Teachable Machine.

Objectives:

- Implement a real-time pose detection system that runs in a web browser.
- Utilize Teachable Machine's pose model for classification.
- Provide an intuitive and user-friendly interface.
- Ensure lightweight and efficient execution without external dependencies.

Technologies Used:

- **HTML, CSS, JavaScript** for front-end development.
- **TensorFlow.js** for machine learning in the browser.
- **Teachable Machine Pose Model** for pose classification.
- **Canvas API** for visualizing pose detection.

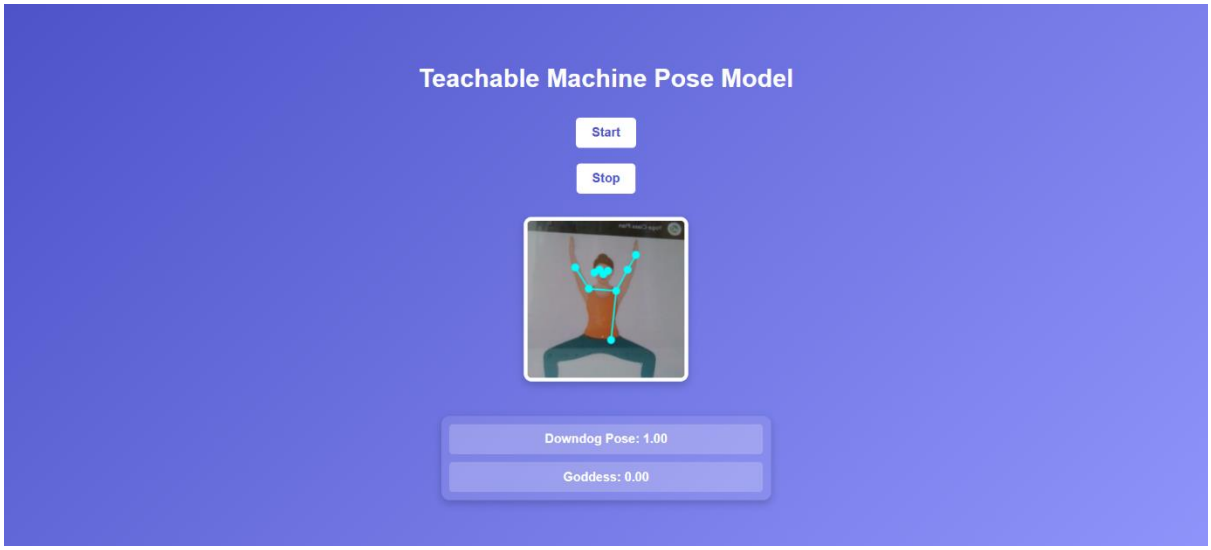
System Architecture:

- **Data Collection & Model Training**: The model is trained using Google's Teachable Machine, which classifies different human poses.
- **Webcam Integration**: The browser accesses the user's webcam to capture real-time images.
- **Pose Prediction**: The model processes the webcam feed, detects key body points, and classifies poses.
- **Visualization**: Key points and skeletons are drawn on the canvas for real-time feedback.

Implementation:

- **User Interaction**: Users can start and stop the pose estimation model using buttons.
- **Pose Detection**: The webcam captures frames, and the model predicts the pose in real time.
- **Results Display**: The detected pose and confidence scores are displayed dynamically.

Results:



Code:https://drive.google.com/file/d/12ZTO0dcpNeO-RWDsF4hiH5tJcGpqwYjY/view?usp=drive_link

Applications:

- Fitness tracking and posture correction.
- Gesture-based human-computer interaction.
- Interactive gaming and motion-based applications.
- Real-time monitoring in healthcare and rehabilitation.

Conclusion:

The Web-Based Real-Time Pose Estimation Model offers a lightweight, browser-compatible solution for pose recognition. With advancements in machine learning and web technologies, future enhancements can include improved accuracy, support for additional poses, and integration with cloud-based analytics.