Presentation Slides

WilliamBuryat Takahashi

Team Name: Signalyze
Members:Marco Chen, Rigoberto Ponce,Willian Buryat Takahashi
Idea:Sign language detection system that recognizes hand gestures, converts them to text, and outputs speech in real-time.

Week 8 TA notes:
I appreciated you all sharing what you've been up to and communicating more clearly how everything is coming together.
Please read my previous week notes if you haven't already since you updated your doc, I've included my previous week notes.
I'm still waiting for a team name!
Week 7 TA notes:
Look at some resources from Claudio's project which is similar to yours:
layout: forward target: https://developers.google.com/mediapipe/solutions/vision/hand_landmarker title: Hands parent: MediaPipe Legacy Solutions nav_order: 4 — MediaPipe v0.7.5 documentation
Sign Language Detection using Images

Tutorials tutorials tutorials!
how to set up a backend to test your model and feed it images or video streams.
Flask is a good option but has a higher learning curve and you need to read documentation.

Week 6 TA notes:
3Blue1Brown Neural Nets Series on YT, really good watch explaining how NN's work.
I watched this to understand things in my ML class and to do my CTP project.
Link to the notebook of my CTP project done with Georgios, the Friday TA.
I did emotion classification on audio data, it's a bit different than yours but still uses neural networks which might be of interest to you. You can check out how we did cleaning and preprocessing, as well as a technique called data augmentation: increasing your data artificially so you have more to train the model with. You can also see how we made a CNN model and evaluated its performance.
Focus on your MVP (minimum viable project/product)
try to get the core functionality of your project to work decently.
Remember you can use a pre-trained model which can be found on HuggingFace or Kaggle and use it. You can also fine-tune the model to work on the data you want.
There are definitely a handful of ASL detection models already built and published by others.
Please update your design doc with your progress by week if possible.
Please come up with a team name!

Notes:

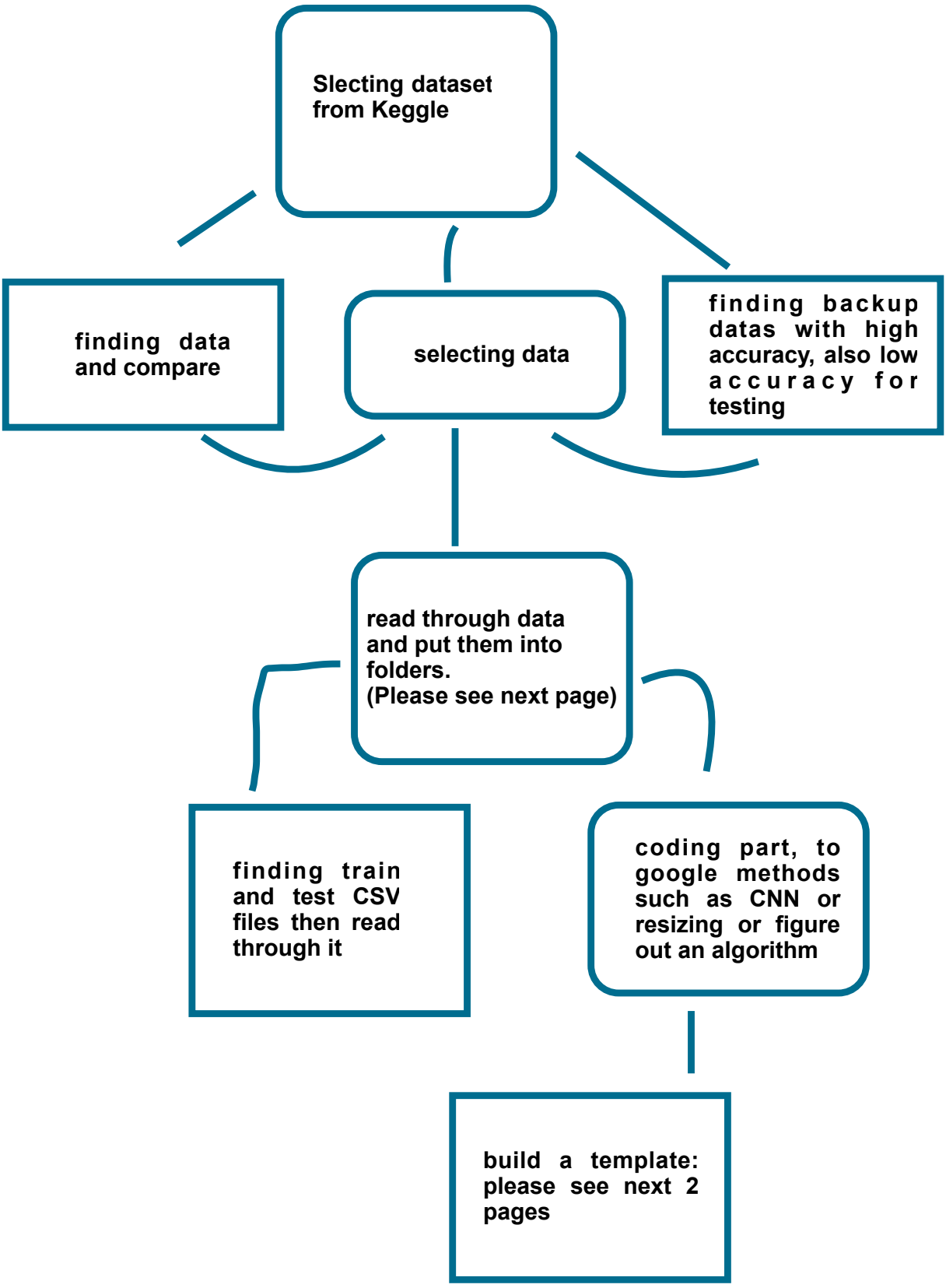
—----------
Concept:
A deep learning-based model to recognize sign language gestures.
Uses OpenCV for real-time video capture and gesture classification.
Converts recognized gestures into corresponding text and speech using TTS (text-to-speech) tools.
Tools:
TensorFlow: Deep learning models and large-scale machine learning.
Keras: Simplify building and training deep learning models.
OpenCV: Image and video processing for real-time computer vision.
gTTS/Pyttsx3
gTTS: Python library for converting text to speech using Google's TTS service.
Pyttsx3: Cross-platform offline text-to-speech library for Python, extendable to different TTS engines.

Solutions:

Main Goal 1: Clean Database and Make CNN Model
Choose a Database:
Find a publicly available sign language gesture dataset, or collect one.
Kaggle, Sign Language MNIST
Ensure the dataset includes a variety of hand gestures, labeled appropriately.
Preprocess and Clean Data:
Convert all images to a uniform size (e.g., 64x64 pixels) and format (e.g., grayscale).
Normalize the pixel values (scale them between 0 and 1).
Split the dataset into training, validation, and test sets.
Consider using data augmentation techniques such as rotation, zooming, and flipping to enhance the diversity of the dataset.
Build CNN Model:
Using Keras (with TensorFlow as the backend), design a CNN architecture that includes layers such as convolutional, pooling, and dense layers for classification.
Compile the model with an optimizer like Adam and a loss function like categorical cross-entropy for multi-class classification.
Train the model on the preprocessed dataset and track performance using accuracy and loss metrics.
Evaluate the model on the test set to see how well it generalizes to unseen data.
Main Goal 2: Use OpenCV to Test and Optimize the Model
Access Live Video:
Use OpenCV to capture a live video stream from a webcam.
Preprocess each frame in real-time to prepare it for the CNN model (resize, normalize).
Apply the CNN Model:
Load the trained CNN model and apply it to the preprocessed video frames.
For each frame, classify the gesture and output the prediction in real-time.
Optimize for Real-Time Performance:
Adjust model complexity to enhance frame rate.
Test various model configurations to strike a balance between accuracy and performance.
Optimize video feed processing steps like resizing and color space conversion to reduce processing time per frame.
Main Goal 3: Work on Text-to-Speech Part
Classify Gesture and Convert to Text:
Once a gesture is classified by the CNN, map it to its corresponding word or phrase (e.g., gesture for "hello" maps to the text "hello").
Convert Text to Speech:
Use Python libraries like pyttsx3 or gTTS (Google Text-to-Speech) to convert the classified gesture (text) into speech.
Integrate this step so that after recognizing a gesture, the system speaks out the corresponding word or phrase.
Main Goal 4: Built a interface to host the model
Design the Interface:
Build a user interface (UI) using a framework like Flask, Django, or Streamlit.
Allow users to interact with the model by uploading videos or accessing a webcam for live gesture classification.
Host the Model:
Integrate the trained CNN model with the UI to classify gestures.
Display the results in real-time or after video uploads.

My task: Testing A training mode
finding a platform or hosting solution
once the model is build my teammate can implement them into other platform with cleaned data and text to speech add on

My solution is to design a system that can read and train a large csv file then split them into x train x test, y train, y test. After that I will use CNNs mode to formatting the data. If suceed, I will produce a h5 file with usually 10 epoches and accuracy rates.



Learnings:
Week1 : learning Pandas and numpy through CTP projects. we will be posting linkendin posts, completing related homeworks, as well as do additional studies.

Week2: DS overview and data. We explored furture on data science.

week3: Data Analytics and Viz Bi, where we learnt about how to use tableu and other tools

week 4: Making teams and discussions. I had a hard time, but eventually find a late team.

week5: unsupervised learning, where we learnt about clustering and finding a line that can better make senses of random data. Then we discussed about potential project.

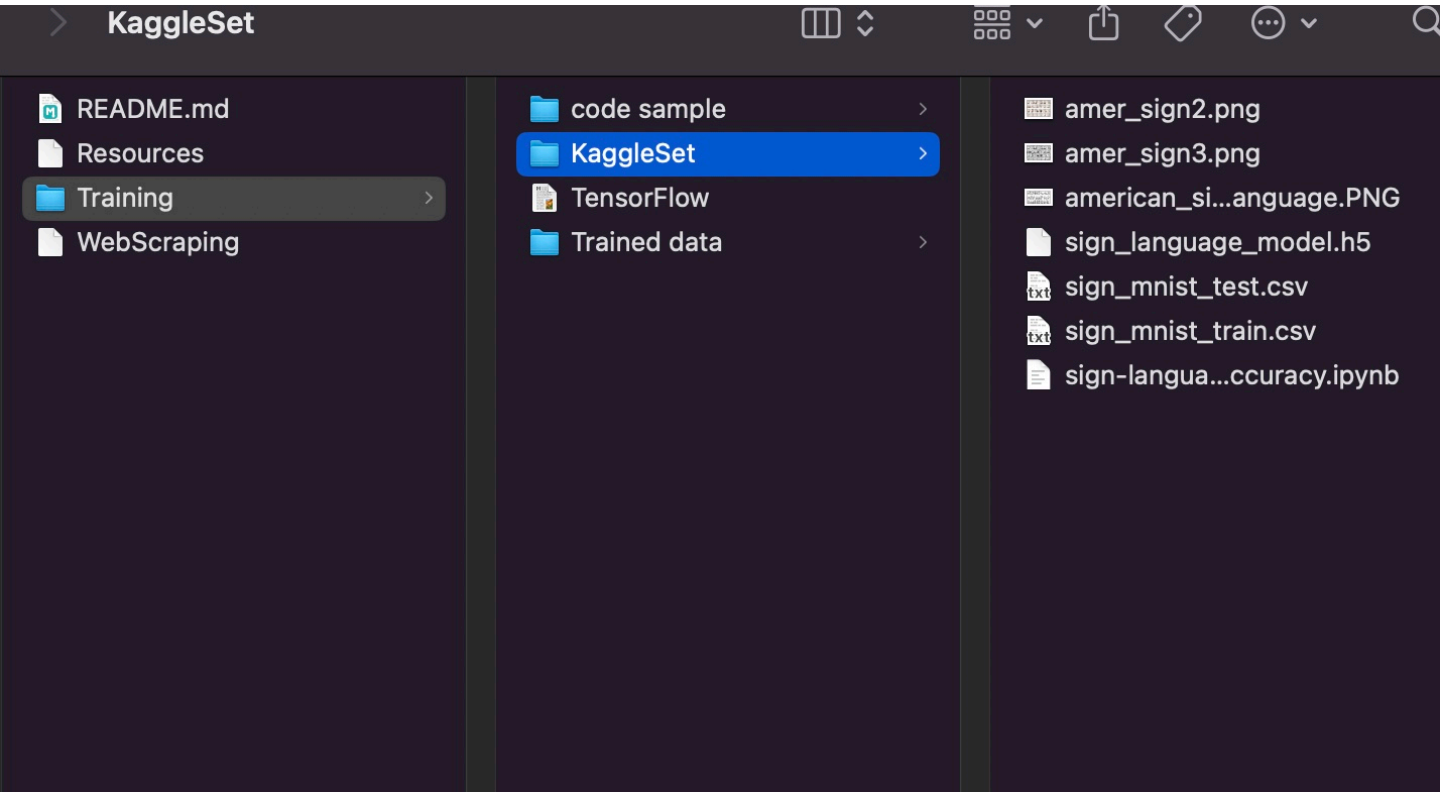
week6: Classification, and finding project. Our team decided to do a sign language deteciton using image processing technic

week 7: Random Forest and tree, and Hugging face. I am thinking about using hugging face open sources to improve our mode.

For my repo and homework completion:

https://github.com/WillSnakeTaka/my-fork-2024-DS-Tue

**The training section consists of code sample,
Kaggle set, with test and train csv.
Trained data consists of h5 file, and TensorFlow
testing codes.**



Rectify layer units
Relu convert negative numbers to 0 then goes Pooling:

-1 3 8. 0 3 8
-2 9 -1. 0 9 0
1 4 8. 1 4 8

During pooling it continue break down and take the right number.
Data => Hidden layers(Convolution_Relu(CR)-> CR.. repeat a few times) , then Pooling then classification.

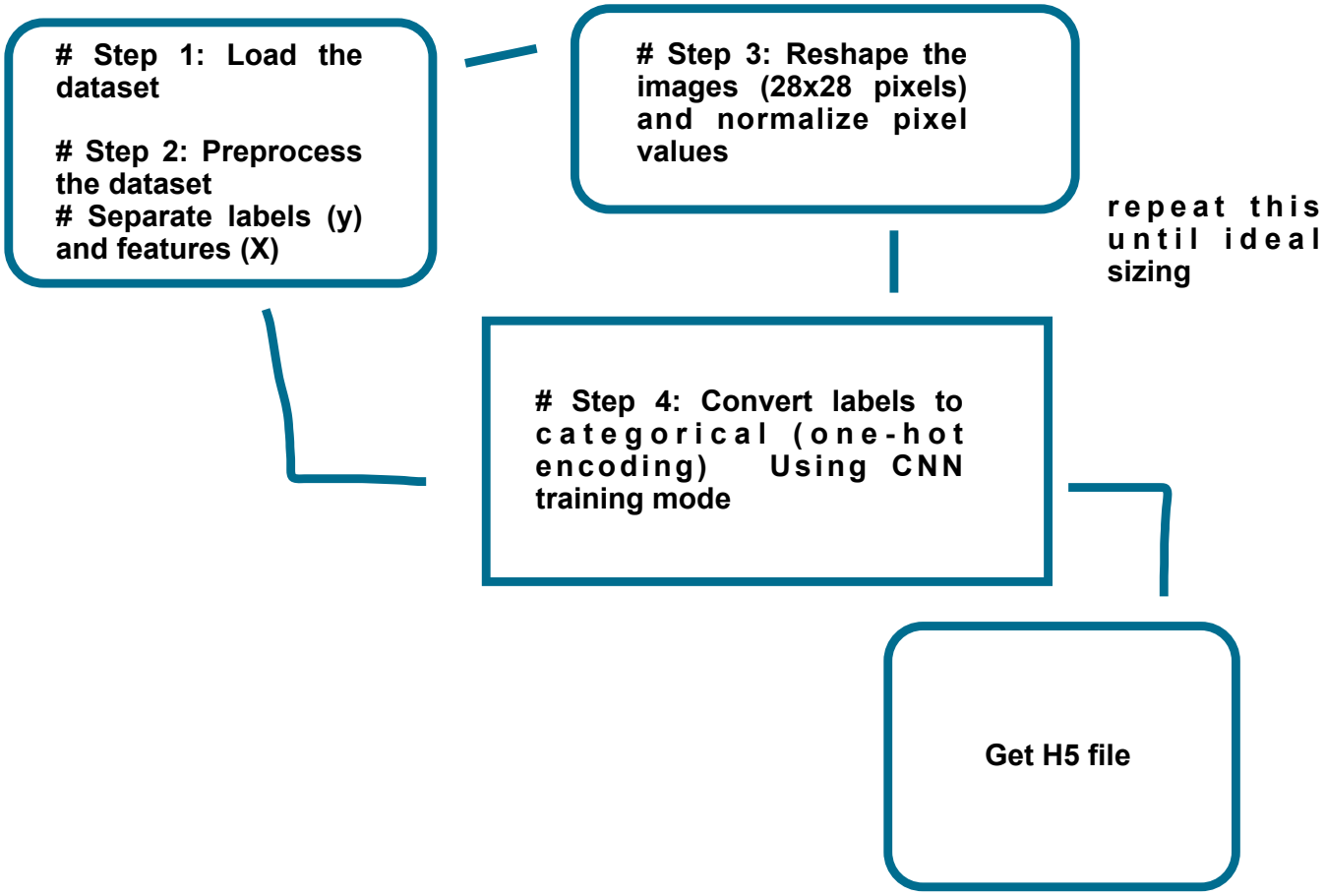
Can start over with newly trained data, or transfer learning where one trained data can be used for similar questions.
Feature extraction can use pre trained CNN to predict data.

Trained data in h5 or HDF5 formate can be reused to predict new data.
For version control, there can be M1.h5 and M2.h5 so on and make improvements.
Trained data can be used in transfer learning. It can also be used to test the performances of new data.

```
From tensorflow.keras.models import load_model
Model = load_model('UrFile.h5')
Predictions = model.predict(newData)
```

Python's BytesIO from io module can read uploaded file directly from memory.

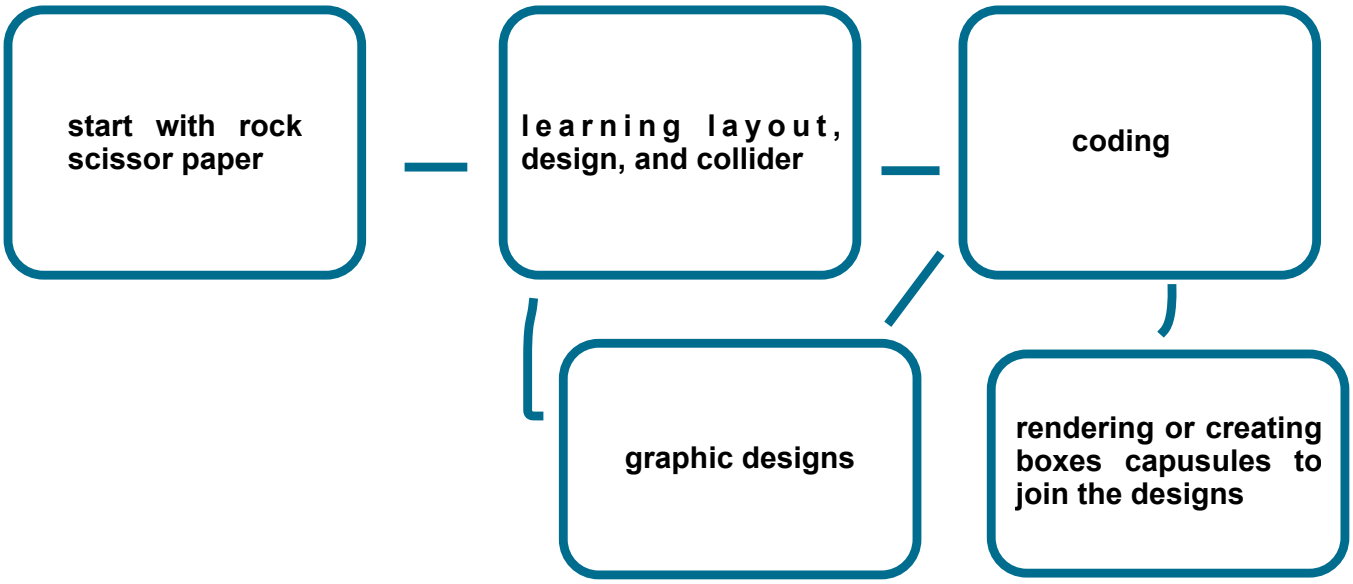
Based on these studying notes. I decided to explore CNN mode for our data.



```
import pandas as pd
import numpy as np
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

# Step 1: Load the dataset
train_data = pd.read_csv('sign_mnist_train.csv')
test_data = pd.read_csv('sign_mnist_test.csv')

# Step 2: Preprocess the dataset
# Separate labels (y) and features (X)
X_train = train_data.iloc[:, 1:].values
y_train = train_data.iloc[:, 0].values
X_test = test_data.iloc[:, 1:].values
y_test = test_data.iloc[:, 0].values

# Step 3: Reshape the images (28x28 pixels) and normalize pixel values
X_train = X_train.reshape(-1, 28, 28, 1) / 255.0
X_test = X_test.reshape(-1, 28, 28, 1) / 255.0

# Step 4: Convert labels to categorical (one-hot encoding)
y_train = to_categorical(y_train, 25)  # 25 classes (A-Y)
y_test = to_categorical(y_test, 25)

# Step 5: Define the CNN model
model = Sequential()

# First Convolutional layer + MaxPooling
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Second Convolutional layer + MaxPooling
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Flatten the output for fully connected layers
model.add(Flatten())

# Fully connected layer
model.add(Dense(128, activation='relu'))

# Dropout to prevent overfitting
model.add(Dropout(0.5))

# Output layer (25 classes for A-Y)
model.add(Dense(25, activation='softmax'))

# Step 6: Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Step 7: Train the model
model.fit(X_train, y_train, epochs=10, batch_size=64, validation_data=(X_test, y_test))

# Step 8: Save the trained model
model.save('sign_language_model.h5')

# Print a message to confirm the model has been saved
print("Model has been trained and saved as sign_language_model.h5")
```

```
from flask import Flask, request, jsonify
from io import BytesIO
import tensorflow as tf
from PIL import Image
import numpy as np

# Load model from a saved .h5 file using BytesIO
# Use the correct file path to your model
with open("/Users/gavtaka/Desktop/Desktop/CV_Handsigns/sign_language_model.h5", "rb") as model_file:
    model_bytes = BytesIO(model_file.read())
        model = tf.keras.models.load_model("/Users/gavtaka/Desktop/Desktop/CV_Handsigns/sign_language_model.h5")



# Initialize Flask app
app = Flask(__name__)

# Preprocessing function (customize as needed)
def preprocess_image(image):
    # Resize image or other preprocessing steps based on your model's requirements
    image = image.resize((224, 224))  # Example: Resize to 224x224, adjust if necessary
    image = np.array(image) / 255.0  # Normalize if required
    image = np.expand_dims(image, axis=0)  # Add batch dimension
    return image

@app.route('/predict', methods=['POST'])
def predict():
    # Check if an image file is part of the request
    if 'file' not in request.files:
        return jsonify({"error": "No file part in the request"}), 400

    # Get file from request
    file = request.files['file']
    if file.filename == '':
        return jsonify({"error": "No file selected"}), 400

    try:
        # Read image file and preprocess
        image = Image.open(BytesIO(file.read())).convert('RGB')  # Ensure the image is in RGB mode
        image = preprocess_image(image)

        # Predict using the loaded model
        predictions = model.predict(image)
        predicted_class = np.argmax(predictions, axis=1).tolist()  # Assuming classification model

        # Return JSON response
        return jsonify({
            "predicted_class": predicted_class,
            "predictions": predictions.tolist()
        })

    except Exception as e:
        return jsonify({"error": str(e)}), 500

if __name__ == '__main__':
    app.run(debug=True)
```

Flask has disadvantages, and my TA also complained about this platform so I have a plan B to learn Unity platform, as well as develope a game. Our team also agreed that we should have a plan B just in case if our project unfortunately failed, we can tranfer our knowledges into another project with higher sucessful rate. I chose gaming unity platform because it has so many build in designs. It even has AI training mode recently. Also, the drag and drop system is very good for add on files that has working. Youtube also has someone posted that training h5 file can be used by Unity with a plugin method.



Unity and TensorFlow can be used together to implement machine learning in Unity games:
TensorFlowSharp: Allows users to use pre-trained TensorFlow graphs in Unity games. This feature is experimental and Unity Technologies does not officially support it for production games.
Unity to TensorFlowJS: An open-source community with a backend that supports advanced training techniques.
TensorFlow plugin: Can be used with Unity 2019.2 or above. To use the plugin, users can:
Open the project in Unity
Install the TensorFlow plugin
Open the Classify or Detect scene in the Assets folder
Add ENABLE_TENSORFLOW to the Scripting Define Symbols
Set Scripting runtime version to .NET 4.6 Equivalent
Build and run
TensorFlow Lite for Unity: Can be installed using Git-LFS.
Image classification: Can be performed using Unity Barracuda and TensorFlow. For example, users can gather a dataset of images to classify, such as beach-balls and footballs.
Python Tensorflow model: Can be imported into Unity using a web socket connection.
TensorFlow is a platform that can help with data automation, model tracking, performance monitoring, and model retraining.

For gaming backup plan. I have created a delivery Cat game

The game summarized most basic ideas about unity, and it can be transfered into our main project, or as backup plan if we failed, we can put up a less complicated datasets in to continue our project.

The game consists of an intro page, followed by click buttons

The game has graphic designs and mappings, include two main cameras. main game play camera and mini maping.

The game has collider codes, delivery codes, constrains (to block moving), drive through and hit and bonce, destory(remove collided objects), changing colors when touch, pick up objects but cannot pick up a second until it is delivered or "destoryed", display console text in screen, Screen change from intro to game, or game to intro by hitting ESC, so there is an ESC button. Then the game is controlled by direction keys on keyboard.

Here is a sample, though the format and sizing need to be fixed later:

playable below:

https://gavtaka.itch.io/deliverycat

or overrall presentation:
https://youtube.com/shorts/OXUQzR4dqOA?si=kZ4p0eOqdBE0KvBQ

Next few slides will be codes

Back to Menu

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class BackToMenu : MonoBehaviour
{
   void Update()
   {
      // Check if the Escape key is pressed
      if (Input.GetKeyDown(KeyCode.Escape))
      {
         ReturnToMainMenu();
      }
   }

   void ReturnToMainMenu()
   {
      // Load the main menu scene
      SceneManager.LoadScene("Intro"); // Replace "MainMenu" with the exact name of your main menu scen
   }
}
```

camera follows (where the moving object is following the camera so it wont drop out of the screen)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CameraFollow : MonoBehaviour
{

   [SerializeField] GameObject FollowMe;
   // Update is called once per frame
   void LateUpdate()
   {
      transform.position = FollowMe.transform.position + new Vector3 (0,0,-10);
   }
}
```

Console to Screen (where console error texts are shown on screen, this is helpful for short conversation after collider)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro; // Import TextMeshPro namespace

public class ConsoleDisplay : MonoBehaviour
{
    public TextMeshProUGUI consoleText; // Use TextMeshProUGUI instead of Text
    public int maxMessages = 2; // Maximum number of messages displayed on screen
    public float messageDisplayDuration = 3f; // Duration each message is displayed (in seconds)

    private Queue<string> messageQueue = new Queue<string>(); // Store messages in a queue

    void OnEnable()
    {
        Application.logMessageReceived += HandleLog;
    }

    void OnDisable()
    {
        Application.logMessageReceived -= HandleLog;
    }

    void HandleLog(string logString, string stackTrace, LogType type)
    {
        messageQueue.Enqueue(logString);

        if (messageQueue.Count > maxMessages)
        {
            messageQueue.Dequeue();
        }

        UpdateConsoleText();
        StartCoroutine(RemoveMessageAfterDelay());
    }

    private void UpdateConsoleText()
    {
        consoleText.text = string.Join("\n", messageQueue.ToArray());
    }

    private IEnumerator RemoveMessageAfterDelay()
    {
        yield return new WaitForSeconds(messageDisplayDuration);

        if (messageQueue.Count > 0)
        {
            messageQueue.Dequeue();
            UpdateConsoleText();
        }
    }
}
```

SrializeField

SpriteRenderer

Main events

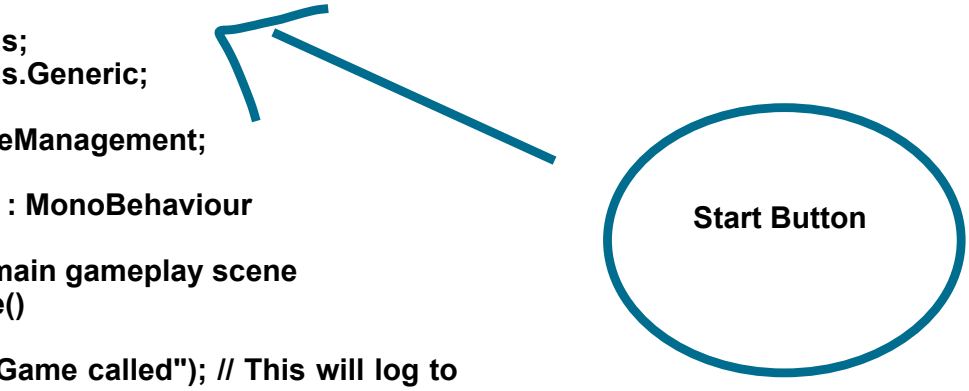
OnTrigger, and if conition

Run debug when condition has been made

This is the main code, where color can be initialized into default then change as object change. Sprite Renderer can change colors of the object after event.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Delivery : MonoBehaviour
{
    [SerializeField] Color32 hasPopcornColor = new Color32 (1,1,1,1);
    [SerializeField] Color32 NoPopcornColor = new Color32 (1,1,1,1);
    [SerializeField] float destroyDelay =0.5f;
    bool hasPopcorn;

    SpriteRenderer spriteRenderer;

    void Start(){
      spriteRenderer = GetComponent<SpriteRenderer>(); // This is for changing colors
    }
    void OnTriggerEnter2D(Collider2D other) {
      if(other.tag == "Popcorn" && !hasPopcorn){ // cannot pick up other packages
      Debug.Log("I wonder do they need a pet?");
      hasPopcorn = true;
      spriteRenderer.color = hasPopcornColor;
      Destroy(other.gameObject, destroyDelay);
      }
      if(other.tag == "Snake" && hasPopcorn){
      Debug.Log("You can stay with us Sweetheart!");
      hasPopcorn = false;
      spriteRenderer.color= NoPopcornColor;
      }
    }
}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class StartButton : MonoBehaviour
{
    // Method to load the main gameplay scene
    public void StartGame()
    {
        Debug.Log("StartGame called"); // This will log to the console if the method is triggered.

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex+1); // Replace "MainGame" with the actual name of your gameplay scene
    }
}
```




```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Driver : MonoBehaviour
{
    [SerializeField]float steerSpeed = 0.1f;
    [SerializeField]float moveSpeed = 0.01f;
    void Start()
    {
        // transform.Rotate(0, 0, 45); // calling the rotate method
    }

    // Update is called once per frame
    void Update()
    {
        float steerAmount = Input.GetAxis("Horizontal") * steerSpeed*Time.deltaTime;
        float moveAmount = Input.GetAxis("Vertical") * moveSpeed*Time.deltaTime;
        transform.Rotate(0, 0, -steerAmount);
        transform.Translate(0, moveAmount, 0);
    }
}
```

Music cut scene, where start and end and fade in out can help menage audio files

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class MusicCut : MonoBehaviour
{
        public AudioSource audioSource; // Reference to the
AudioSource component
   public float startTime = 0f;    // Start time in seconds
   public float endTime = 48f;     // End time in seconds (1 minute
and 3 seconds)
    public float fadeDuration = 3f; // Duration of fade-in and fade-
out in seconds
   private bool isFadingOut = false;

   void Start()
   {
      audioSource.time = startTime;
       audioSource.volume = 0f; // Start with volume at 0 for fade-
in
      audioSource.Play();
      StartCoroutine(FadeIn()); // Start the fade-in coroutine
   }

   void Update()
   {
            if (!isFadingOut && audioSource.time >= endTime -
fadeDuration)
      {
               StartCoroutine(FadeOut()); // Start fade-out before
reaching endTime
      }

      if (audioSource.time >= endTime && !isFadingOut)
      {
         audioSource.time = startTime;
         audioSource.Play();
         StartCoroutine(FadeIn()); // Restart and fade-in again
      }
   }
```

```
IEnumerator FadeIn()
    {
        float targetVolume = 1f; // Target volume level for full sound
        float currentTime = 0f;

        while (currentTime < fadeDuration)
        {
            currentTime += Time.deltaTime;
            audioSource.volume = Mathf.Lerp(0f, targetVolume, currentTime / fadeDuration);
            yield return null;
        }
        audioSource.volume = targetVolume; // Ensure full volume at the end of fade-in
    }

    IEnumerator FadeOut()
    {
        isFadingOut = true;
        float startVolume = audioSource.volume;
        float currentTime = 0f;

        while (currentTime < fadeDuration)
        {
            currentTime += Time.deltaTime;
            audioSource.volume = Mathf.Lerp(startVolume, 0f, currentTime / fadeDuration);
            yield return null;
        }
        audioSource.volume = 0f; // Ensure volume is zero at the end of fade-out
        // Reset the audio source and play from the start time
        audioSource.time = startTime;
        audioSource.Play();

        // Start the fade-in again
        StartCoroutine(FadeIn());
    }
}
```

To be improved:
Formatting, resizing, and layout. As for data science I am still learning