Hand Gesture Recognition

DATA 606: Capstone Project in Data Science

Team G
Daksh Intwala
Jaydeep Radadia
Kratika Aggarwal

Problems

The hearing impaired would benefit from greater accomodations aid in the difficulties they face.

01

Lack of opportunities

Fewer educational and job opportunities due to impaired communication.

02

Social Withdrawal

Reduced access to services and communcation barrier with peers.

03

Emotional Issues

Low self esteem and confidence.

Who are our target audience?

The hearing, speaking impaired

Expressing oneself

It is extremely expensive & time consuming for everyone to learn the language. This will help in live communication wherein the ASL user will be able to directly communicate with the everyone.

Mass Communication

Speech generation is possible which can give voice to the ASL user. It expands the scope of communication for the conversation.

Scope of Language

ASL user can communicate in multiple languages if the text output is translated in different languages using transfer learning (BERT-Hugging Face).

Literature Review

PROJECT - 01

CNN - Letter detection/recognition (93%)

Source: https://www.kaggle.com/code/danielmarom/sign-language-using-cnn-93-with-new-images

Methodology:

a. Read dataset
b. Divide it into train/test
c. Sequential model - relu +softmax
    i. Loss = sparse_categorical_crossentropy
    ii. Optimizer = adam
    iii. Metric = accuracy

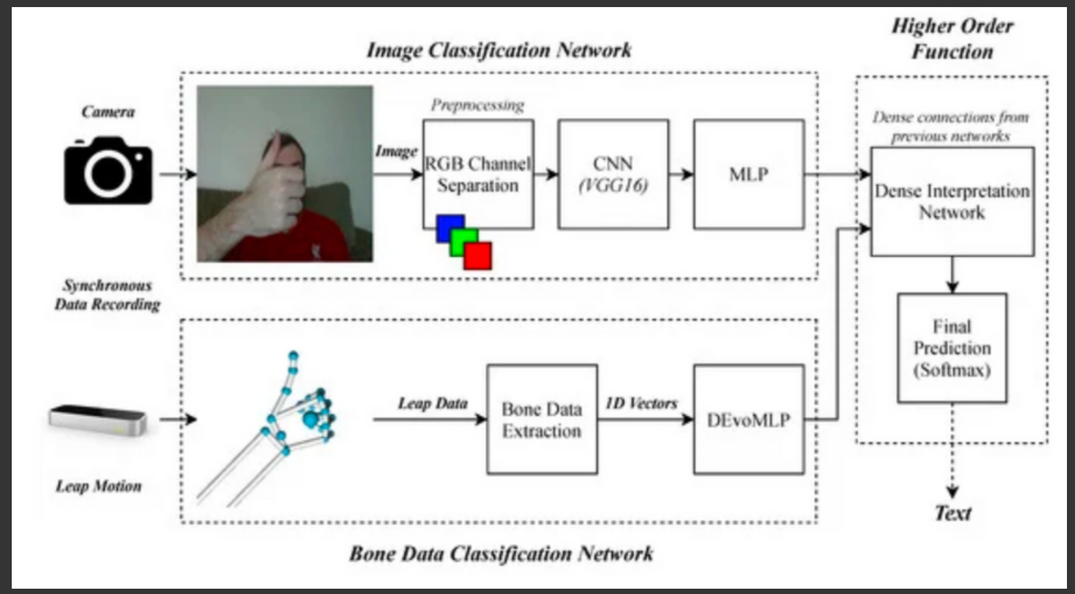
Literature Review

PROJECT - 02

British Sign Language Recognition via Late Fusion of Computer Vision and Leap Motion with Transfer Learning to American Sign Language

Source: https://www.mdpi.com/1424-8220/20/18/5151

Methodology:



Literature Review

PAPER - 01

Hand gestures for emergency situations: A video dataset based on words from Indian sign language

Source: https://www.sciencedirect.com/science/article/pii/S235234092030910O

Methodology:

In this paper, the author has analyzed the images of hand gestures indicating sign language using GoogLeNet CNN model coupled with LSTM architecture and evaluated the performance using F, Recall and Precision scores.

Literature Review

PAPER - 02

Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparison

Source: https://openaccess.thecvf.com/content_WACV_2020/papers/Li_Word_level_Deep_Sign_Language_Recognition_from_Video_A_New_Large-scale_WACV_2020_paper.pdf

Methodology:

Here, two different deep learning approaches are used to detect sign language from a video i.e.,

(i) holistic visual appearance based approach, and

(ii) 2D human pose based approach.

Data Source

WLASL

It is the largest video dataset for Word-Level American Sign Language (ASL) recognition

It was created with the intention of facilitating the research in sign language understanding and eventually benefit the communication between deaf and hearing communities.

This dataset had approximately 7-8 videos per action. We wanted to have a large amount of data to train out model hence we took this dataset as a reference and recorded our own data.

LINK

https://www.kaggle.com/code/risangbaskoro/reorganize-video-data-wlasl/data

Original Data

Word : Drink

1.96 GB

DATA VOLUME

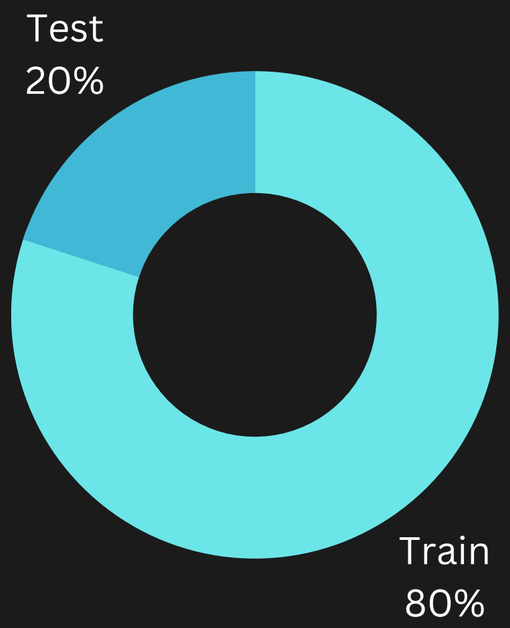
~30

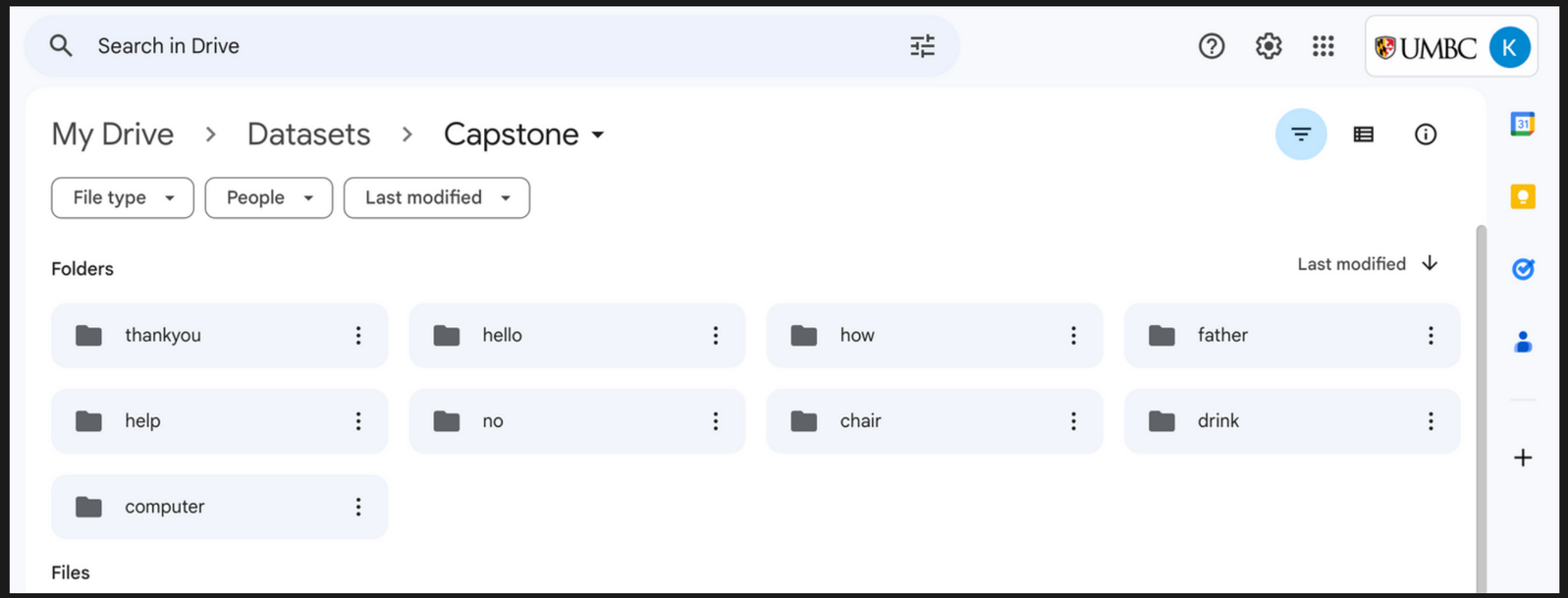
CLIPS PER WORD

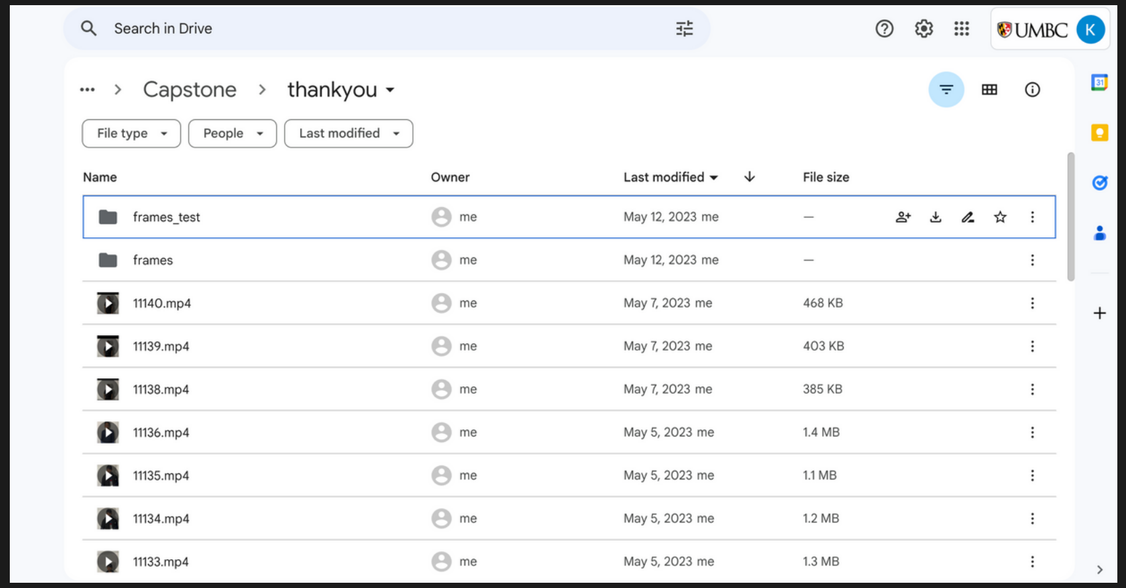
9

WORDS

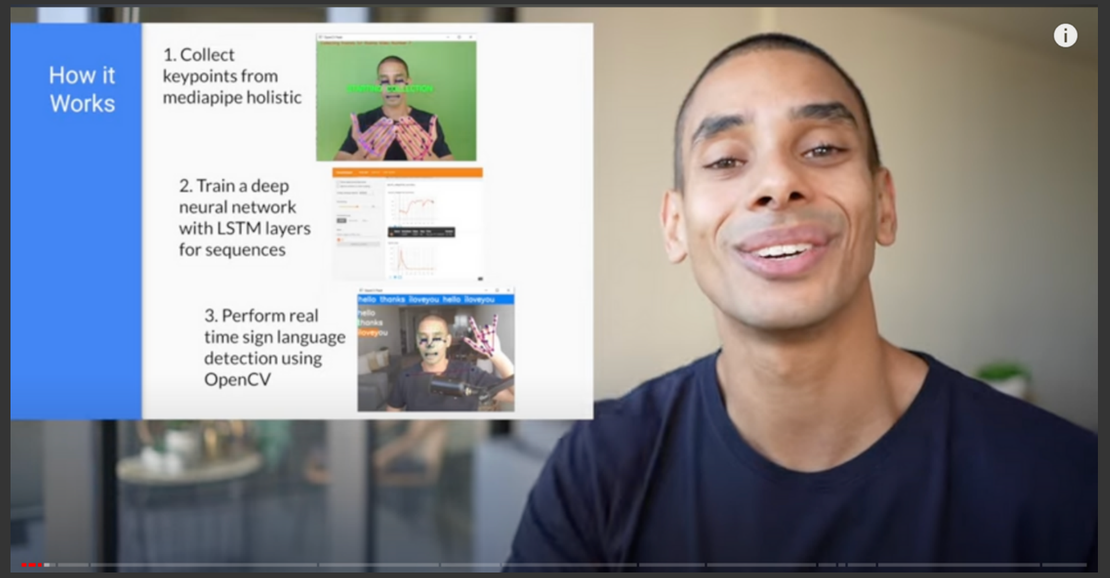
Data Preparation







Methodology - Part A



Algorithm - Classification

STEP 1

Using Mediapipe collect keypoints

STEP 2

Train LSTM Model

STEP 3

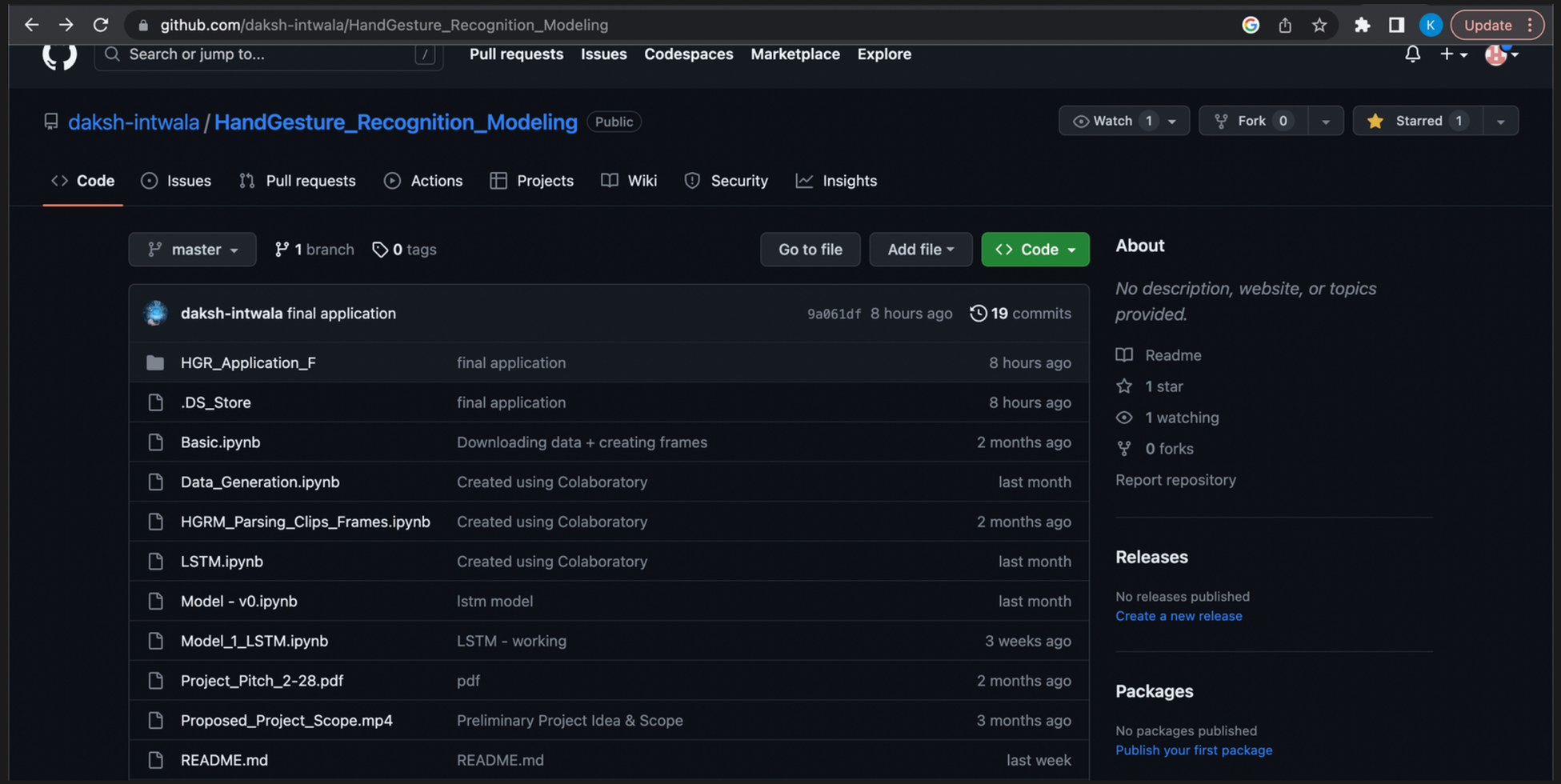
Perform real-time sign language detection using OpenCV

STEP 4

Model Evaluation: Accuracy

GitHub

https://github.com/daksh-intwala/HandGesture_Recognition_Modeling



Pseudo Code - Setting up the data

1. We recorded our own data. ~ roughly 2 seconds

Pseudo Code

1. We recorded our own data. ~ roughly 2 seconds
2. Parsed the videoclip using openCV

Pseudo Code

1. We recorded our own data. ~ roughly 2 seconds
2. Parsed the videoclip using openCV
3. Determined the optimum frame rate

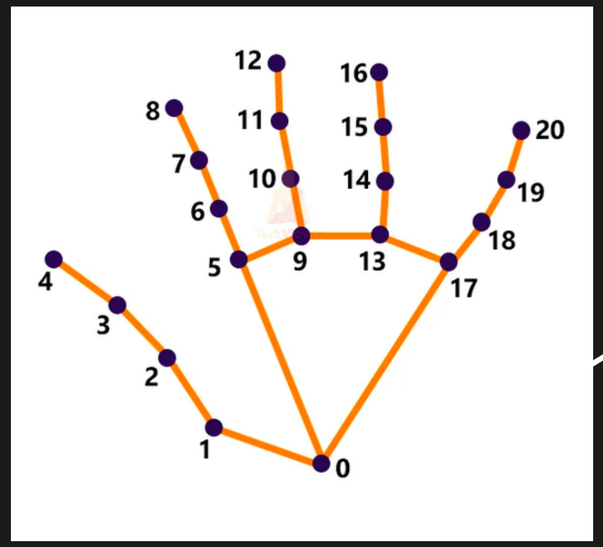
Pseudo Code

1. We recorded our own data. ~ roughly 2 seconds
2. Parsed the videoclip using openCV
3. Determined the optimum frame rate
4. Captured the frames using the .set() method

Pseudo Code

1. We recorded our own data. ~ roughly 2 seconds
2. Parsed the videoclip using openCV
3. Determined the optimum frame rate
4. Captured the frames using the .set() method
5. Saved the frames

Landmarks




```
Hand No. 1
WRIST:
x: 667.0050430297852
y: 612.0800757408142
z: 0.0007318987263715826
THUMB_CMC:
x: 665.6389045715332
y: 548.7694072723389
z: -9.656334221363068
Hand No. 2
WRIST:
x: 1333.542366027832
y: 557.2280859947205
z: 0.00044899124077346 64
THUMB_CMC:
x: 1341.1071395874023
y: 497.4174106121063
z: -29.865333437919617
```

0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

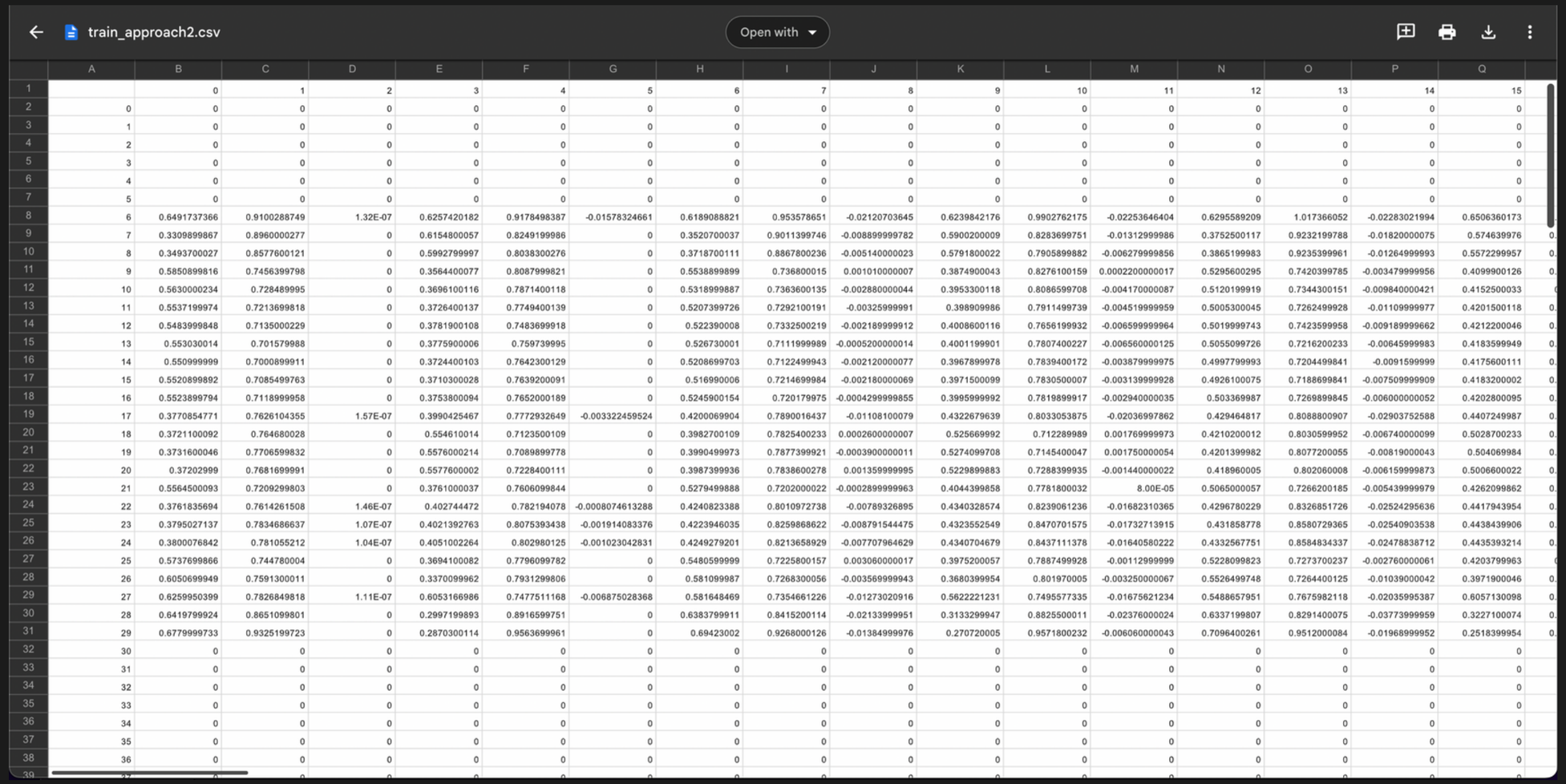
21 Landmarks * 3 Coordinates = 63 Landmarks
(x,y,z)

63 Landmarks * Two Hands = 126 Features
(left,right)

Extracting Landmarks

```python
def landmark_extractor(img):

  mp_hands = mp.solutions.hands

  with mp_hands.Hands(static_image_mode=True,
                      max_num_hands=2,
                      min_detection_confidence=0.5) as hands:

        landmark = []

        image = cv2.flip(cv2.imread(img), 1)

        results = hands.process(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))

        landmarks = results.multi_hand_landmarks

        if landmarks != None:
          landmark_idx = mp_hands.HandLandmark.WRIST.value
          landmark += (np.array([[lmk.landmark[landmark_idx].x,
                        lmk.landmark[landmark_idx].y,
                        lmk.landmark[landmark_idx].z] for lmk in landmarks]).flatten() if landmarks else np.zeros(3)).tolist()

          landmark_idx = mp_hands.HandLandmark.THUMB_CMC.value
          landmark += (np.array([[lmk.landmark[landmark_idx].x,
                        lmk.landmark[landmark_idx].y,
                        lmk.landmark[landmark_idx].z] for lmk in landmarks]).flatten() if landmarks else np.zeros(3)).tolist()
```

Dataset



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	6	0.6491737366	0.9100288749	1.32E-07	0.6257420182	0.9178498387	-0.01578324661	0.6189088821	0.953578651	-0.02120703645	0.6239842176	0.9902762175	-0.02253646404	0.6295589209	1.017366052	-0.02283021994	0.6506360173
9	7	0.3309899867	0.8960000277	0	0.6154800057	0.8249199986	0	0.3520700037	0.9011399746	-0.008899999782	0.5900200009	0.8283699751	-0.01312999986	0.3752500117	0.9232199788	-0.01820000075	0.574639976
10	8	0.3493700027	0.8577600121	0	0.5992799997	0.8038300276	0	0.3718700111	0.8867800236	-0.005140000023	0.5791800022	0.7905899882	-0.006279999856	0.3865199983	0.9235399961	-0.01264999993	0.5572299957
11	9	0.5850899816	0.7456399798	0	0.3564400077	0.8087999821	0	0.5538899899	0.736800015	0.001010000007	0.3874900043	0.8276100159	0.0002200000017	0.5295600295	0.7420399785	-0.003479999956	0.4099900126
12	10	0.5630000234	0.728489995	0	0.3696100116	0.7871400118	0	0.5318999887	0.7363600135	-0.002880000044	0.3953300118	0.8086599708	-0.004170000087	0.5120199919	0.7344300151	-0.009840000421	0.4152500033
13	11	0.5537199974	0.7213699818	0	0.3726400137	0.7749400139	0	0.5207399726	0.7292100191	-0.00325999991	0.398909986	0.7911499739	-0.004519999959	0.5005300045	0.7262499928	-0.01109999977	0.4201500118
14	12	0.5483999848	0.7135000229	0	0.3781900108	0.7483699918	0	0.522390008	0.7332500219	-0.002189999912	0.4008600116	0.7656199932	-0.006599999964	0.5019999743	0.7423599958	-0.009189999662	0.4212200046
15	13	0.553030014	0.701579988	0	0.3775900006	0.759739995	0	0.526730001	0.7111999989	-0.0005200000014	0.4001199901	0.7807400227	-0.006560000125	0.5055099726	0.7216200233	-0.00645999983	0.4183599949
16	14	0.550999999	0.7000899911	0	0.3724400103	0.7642300129	0	0.5208699703	0.7122499943	-0.002120000077	0.3967899978	0.7839400172	-0.003879999975	0.4997799993	0.7204499841	-0.0091599999	0.4175600111
17	15	0.5520899892	0.7085499763	0	0.3710300028	0.7639200091	0	0.516990006	0.7214699984	-0.002180000069	0.3971500099	0.7830500007	-0.003139999928	0.4926100075	0.7188699841	-0.007509999909	0.4183200002
18	16	0.5523899794	0.7118999958	0	0.3753800094	0.7652000189	0	0.5245900154	0.720179975	-0.0004299999855	0.3995999992	0.7819899917	-0.002940000035	0.503369987	0.7269899845	-0.006000000052	0.4202800095
19	17	0.3770854771	0.7626104355	1.57E-07	0.3990425467	0.7772932649	-0.003322459524	0.4200069904	0.7890016437	-0.01108100079	0.4322679639	0.8033053875	-0.02036997862	0.429464817	0.8088800907	-0.02903752588	0.4407249987
20	18	0.3721100092	0.764680028	0	0.554610014	0.7123500109	0	0.3982700109	0.7825400233	0.0002600000007	0.525669992	0.712289989	0.001769999973	0.4210200012	0.8030599952	-0.006740000099	0.5028700233
21	19	0.3731600046	0.7706599832	0	0.5576000214	0.7089899778	0	0.3990499973	0.7877399921	-0.0003900000011	0.5274099708	0.7145400047	0.001750000054	0.4201399982	0.8077200055	-0.00819000043	0.504069984
22	20	0.37202999	0.7681699991	0	0.5577600002	0.7228400111	0	0.3987399936	0.7838600278	0.001359999995	0.5229899883	0.7288399935	-0.001440000022	0.418960005	0.802060008	-0.006159999873	0.5006600022
23	21	0.5564500093	0.7209299803	0	0.3761000037	0.7606099844	0	0.5279499888	0.7202000022	-0.0002899999963	0.4044399858	0.7781800032	8.00E-05	0.5065000057	0.7266200185	-0.005439999979	0.4262099862
24	22	0.3761835694	0.7614261508	1.46E-07	0.402744472	0.782194078	-0.0008074613288	0.4240823388	0.8010972738	-0.00789326895	0.4340328574	0.8239061236	-0.01682310365	0.4296780229	0.8326851726	-0.02524295636	0.4417943954
25	23	0.3795027137	0.7834686637	1.07E-07	0.4021392763	0.8075393438	-0.001914083376	0.4223946035	0.8259868622	-0.008791544475	0.4323552549	0.8470701575	-0.01732713915	0.431858778	0.8580729365	-0.02540903538	0.4438439906
26	24	0.3800076842	0.781055212	1.04E-07	0.4051002264	0.802980125	-0.001023042831	0.4249279201	0.8213658929	-0.007707964629	0.4340704679	0.8437111378	-0.01640580222	0.4332567751	0.8584834337	-0.02478838712	0.4435393214
27	25	0.5737699866	0.744780004	0	0.3694100082	0.7796099782	0	0.5480599999	0.7225800157	0.003060000017	0.3975200057	0.7887499928	-0.00112999999	0.5228099823	0.7273700237	-0.002760000061	0.4203799963
28	26	0.6050699949	0.7591300011	0	0.3370099962	0.7931299806	0	0.581099987	0.7268300056	-0.003569999943	0.3680399954	0.801970005	-0.003250000067	0.5526499748	0.7264400125	-0.01039000042	0.3971900046
29	27	0.6259950399	0.7826849818	1.11E-07	0.6053166986	0.7477511168	-0.006875028368	0.581648469	0.7354661226	-0.01273020916	0.5622221231	0.7495577335	-0.01675621234	0.5488657951	0.7675982118	-0.02035995387	0.6057130098
30	28	0.6419799924	0.8651099801	0	0.2997199893	0.8916599751	0	0.6383799911	0.8415200114	-0.02133999951	0.3133299947	0.8825500011	-0.02376000024	0.6337199807	0.8291400075	-0.03773999959	0.3227100074
31	29	0.6779999733	0.9325199723	0	0.2870300114	0.9563699961	0	0.69423002	0.9268000126	-0.01384999976	0.270720005	0.9571800232	-0.006060000043	0.7096400261	0.9512000084	-0.01968999952	0.2518399954
32	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39	37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Splitting Dataset

1. Sequential Split

Using List slicing to spit the data, as order matters in training.

2. Batch Feeding

Created dataset batched to feed into LSTM data model

LSTM Configuration

```
[ ] from tensorflow.keras.callbacks import ModelCheckpoint
```

```
def create_model():
  max_len=31
  feature_len=126
  classes_len= 216 # len(x_train)
  model = Sequential()
  model.add(LSTM(256, return_sequences=True, input_shape=(max_len, feature_len)))
  model.add(Dropout(0.25))
  model.add(LSTM(256, return_sequences=True))
  model.add(Dropout(0.25))
  model.add(LSTM(128, return_sequences=False))
  model.add(Dense(64))
  model.add(BatchNormalization())
  model.add(Activation('relu'))
  model.add(Dense(9, activation='softmax'))
  model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
  return model
model_11 = create_model()
model_11.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 31, 256)           392192

 dropout (Dropout)           (None, 31, 256)           0

 lstm_1 (LSTM)               (None, 31, 256)           525312

 dropout_1 (Dropout)         (None, 31, 256)           0

 lstm_2 (LSTM)               (None, 128)               197120

 dense (Dense)               (None, 64)                8256

 batch_normalization (BatchN  (None, 64)               256
 ormalization)

 activation (Activation)     (None, 64)                0

 dense_1 (Dense)             (None, 9)                 585

=================================================================
Total params: 1,123,721
Trainable params: 1,123,593
Non-trainable params: 128
_________________________________________________________________
```

Results

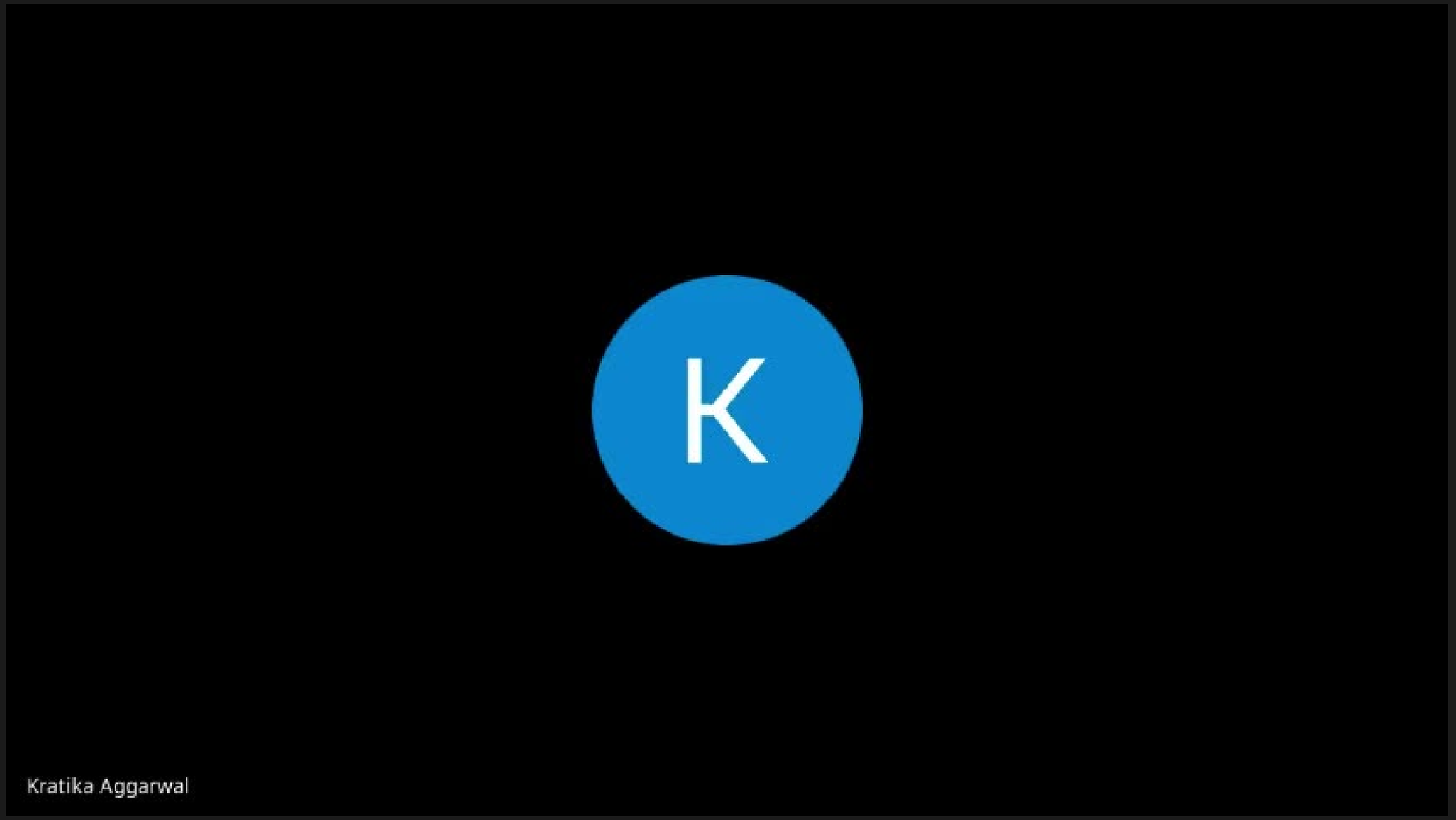
```python
# Actual Model
model_11.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
loss, acc = model_11.evaluate(x_test, test_y, verbose=2)
print("Actual Model, accuracy: {:5.2f}%".format(100 * acc))
```

Python

```
2/2 - 3s - loss: 101.4016 - accuracy: 0.6111 - 3s/epoch - 2s/step
Actual Model, accuracy: 61.11%
```

Validation Set accuracy: 61.11%

Demo



Challenges Faced

1. Data Generation

Team created their own data due to lack of availability of suitable resources. We ended up recording 270 video clips dedicated to 9 actions.

2. MacOS Vs. Windows

Two of the teammates worked on MacOS and one on Windows. BigSur in Mac comes with dependency issues when we work with tensorflow-keras. We found a working solution to use the previous BigSur version in our system which enables the access to library modules we used tensorflow-metal, tensorflow-keras.

3. Google Colab - Code development

To avoid GPU crashes, we had a backup plan to purchase external GPU LambdaLabs independent service following type of virtual hardware :

1x NVIDIA H100 PCIe New, 80 GB, 26 vCPUs, 200 GiB RAM, 512 GiB Storage, $2.40 / hr

Future Scope

Processing & Text Generation

STEP 6

RNN Action Predictions

STEP 7

Queuing actions for 2-3 sec frame

STEP 8

Generate summary through NLG (NLTK, SpaCy)

STEP 9

Grammar-Check Gingerit API

References

1. https://www.betterhealth.vic.gov.au/health/conditionsandtreatments/hearing-loss-how-it-affects-people
2. Muchnik, A. (2023, February 27). Who: 1 in 4 people projected to have hearing problems by 2050. World Health Organization. https://www.who.int/news/item/02-03-2021-who-1-in-4-people-projected-to-have-hearing-problems-by-2050
3. Babour, A., Bitar, H., Alzamzami, O., Alahmadi, D., Barsheed, A., Alghamdi, A. & Almshjary, H. (2023). Intelligent gloves: An IT intervention for deaf-mute people. Journal of Intelligent Systems, 32(1), 20220076. https://doi.org/10.1515/jisys-2022-0076
4. Suarez, J., & Murphy, R. R. (2012, September). Hand gesture recognition with depth images: A review. In 2012 IEEE RO-MAN: the 21st IEEE international symposium on robot and human interactive communication (pp. 411-417). IEEE.