Enhanced Multimodal Gesture Recognition System: Integrating Hand Gestures and Facial Expressions

Abstract: The project aims to address the challenge of accurately recognizing complex human gestures in dynamic environments. Traditional unimodal gesture recognition systems often struggle with ambiguity and noise, leading to limited applicability in real-world scenarios. By integrating hand gestures and facial expressions. This project seeks to create a more robust and versatile gesture recognition system that can interpret a wide range of human interactions. This integration is interesting because it mirrors human communication's complexity, offering a richer, more intuitive interface for human-computer interaction.

Objectives:

- To develop a multimodal gesture recognition system that outperforms unimodal approaches in accuracy and reliability.
- To create a model capable of real-time processing for interactive applications.

Novelty of our project:

- Logical Mapping between hand and face expressions.
- Using Hybrid fusion convolutional neural network architecture.
- Complementing hand gestures with the help of facial expressions.

Real World Datasets:

- **https://www.kaggle.com/datasets/msambare/fer2013 : Facial Emotion Recognition 2013**
- **https://www.kaggle.com/datasets/datamunge/sign-language-mnist/code : Sign Language Prediction**

What we did for analysis.ipynb:

- Data Preprocessing
- Individually training Facial Expression data and Hand gesture recognition data.

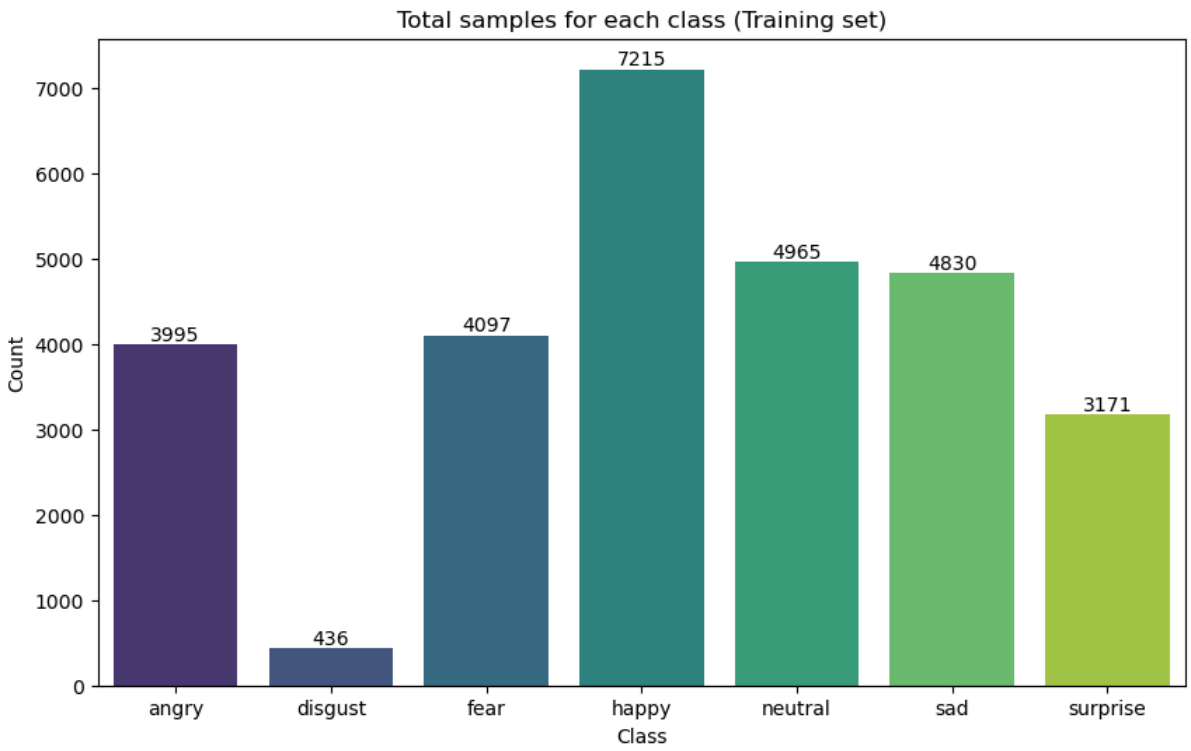
Data Preprocessing:

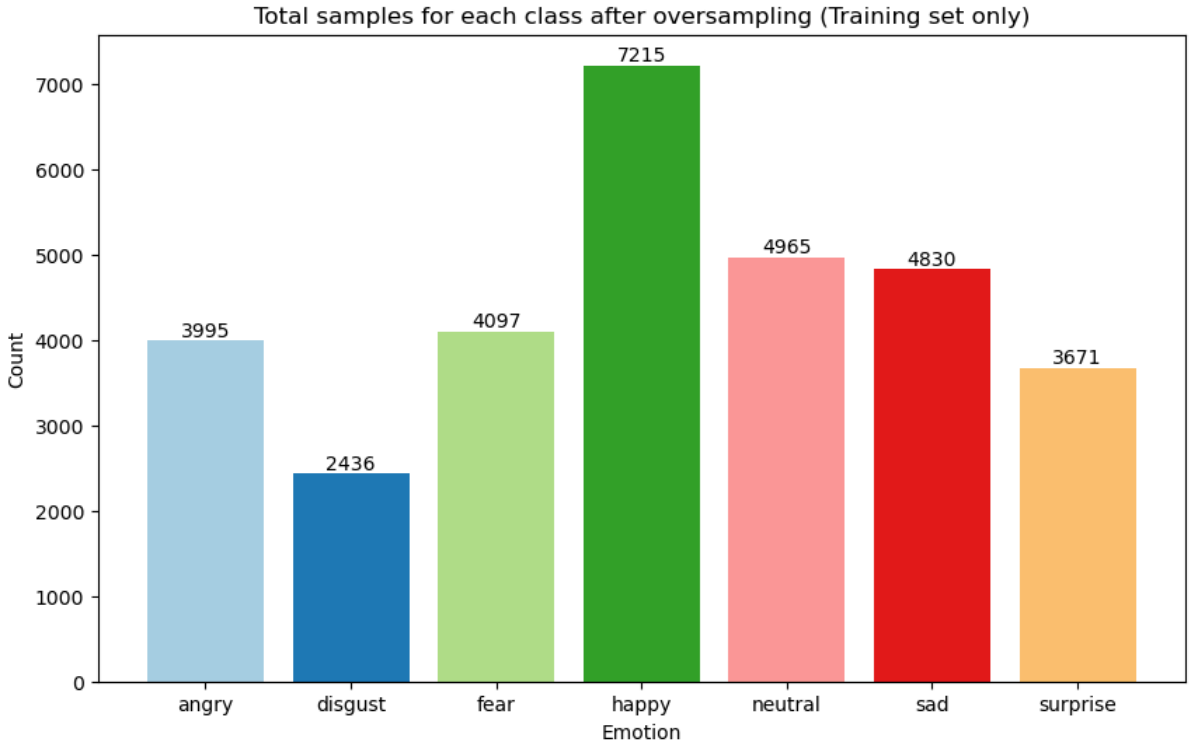
- For Facial Expressions:
    - Normalized image.
    - SMOTE for imbalanced labels.
    - Handled outliers.
- For Hand gestures:
    - Normalized pixel values.

Visualizations:

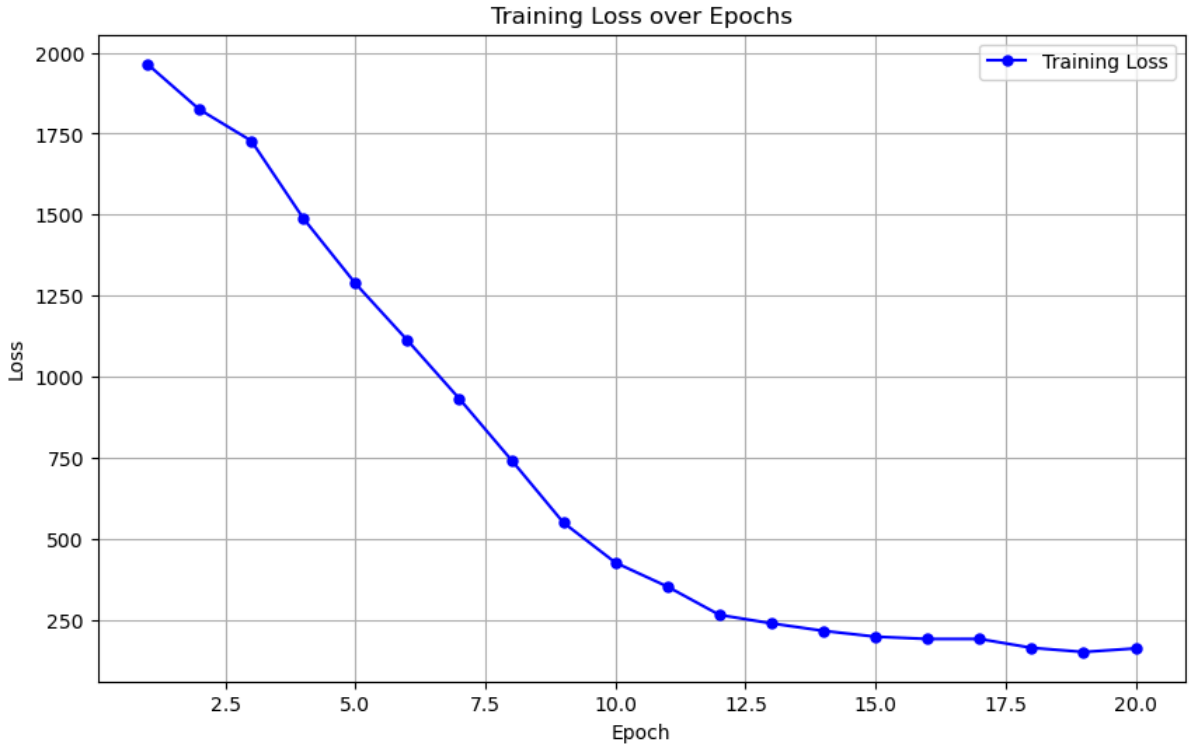
For Facial Expressions:


angry
disgust
fear
happy
neutral
sad
surprise

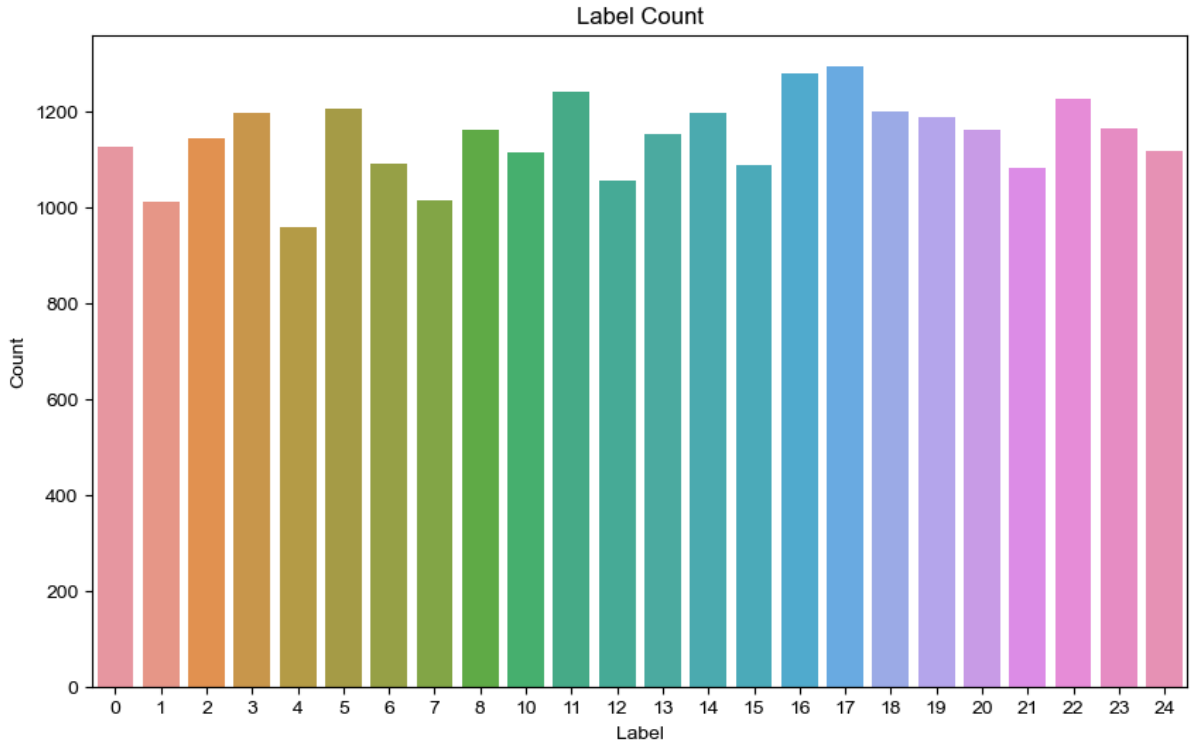

Total samples for each class (Training set)
Count
Class
angry
disgust
fear
happy
neutral
sad
surprise
3995
436
4097
7215
4965
4830
3171
0
1000
2000
3000
4000
5000
6000
7000

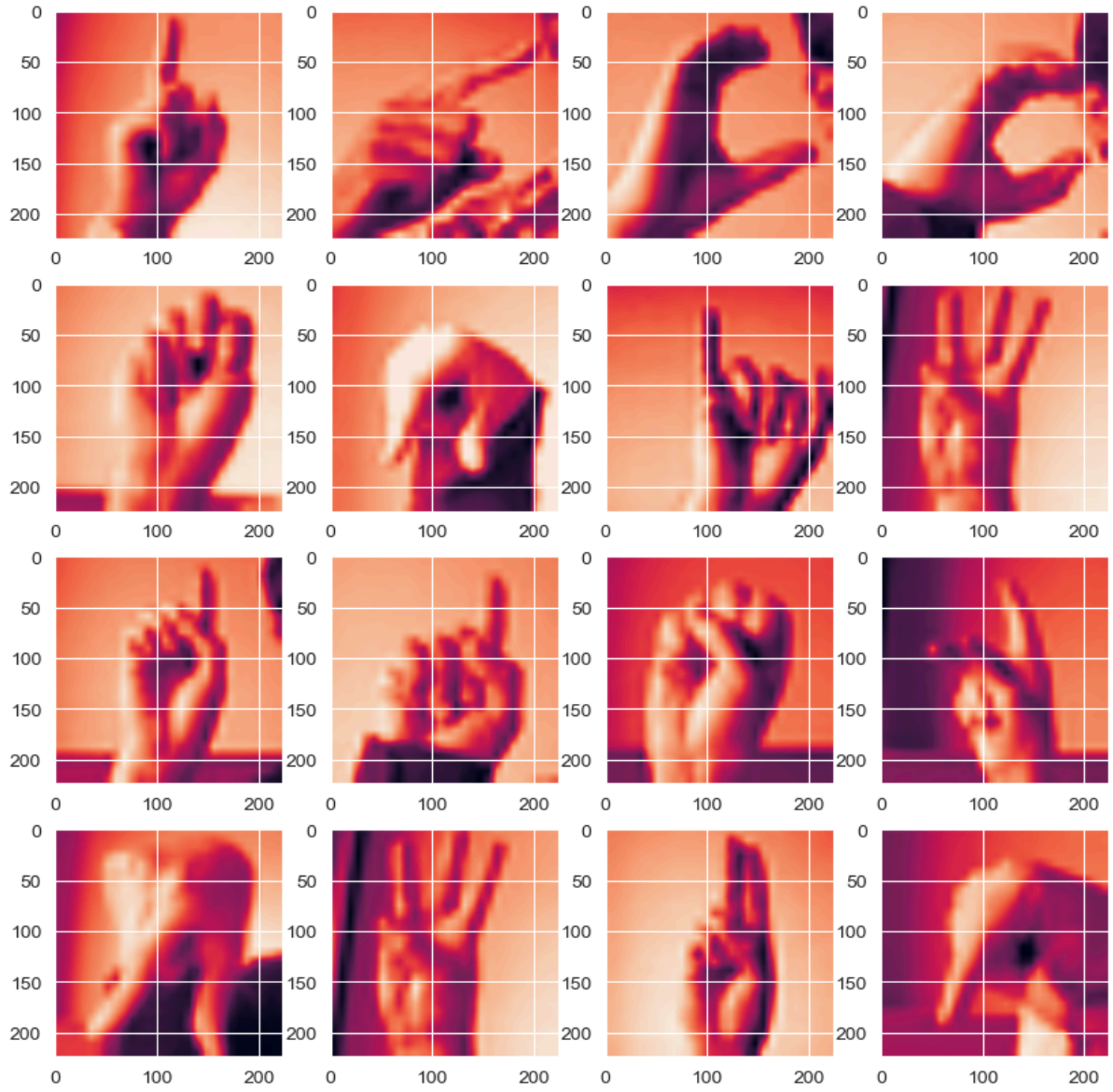

Total samples for each class after oversampling (Training set only)
Count
Emotion
angry
disgust
fear
happy
neutral
sad
surprise
3995
2436
4097
7215
4965
4830
3671
0
1000
2000
3000
4000
5000
6000
7000


Original: Surprise
Original: Surprise
Original: Surprise
Original: Surprise
Original: Surprise
Synthetic: Surprise
Synthetic: Surprise
Synthetic: Surprise
Synthetic: Surprise
Synthetic: Surprise
Original: Disgust
Original: Disgust
Original: Disgust
Original: Disgust
Original: Disgust
Synthetic: Disgust
Synthetic: Disgust
Synthetic: Disgust
Synthetic: Disgust
Synthetic: Disgust


Training Loss over Epochs
Training Loss
Loss
Epoch
2000
1750
1500
1250
1000
750
500
250
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

For Hand Gestures:





Initial results for face data and hand gesture data

Data Transformation with Augmentation:

```
transform = transforms.Compose([
    transforms.Resize((28, 28)),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(10),
    transforms.RandomAffine(degrees=0, translate=(0.1, 0.1)),
    transforms.ToTensor(),
    transforms.Grayscale(),
    transforms.Normalize((0.5,), (0.5,))
])
```

Discarded Disgust labelled images:

```
def load_facial_expressions(base_dir):
   expressions = {}
   for expression in os.listdir(base_dir):
       if expression == 'disgust':
           continue  # Skip disgust images
       expression_path = os.path.join(base_dir, expression)
       if os.path.isdir(expression_path):
           expressions[expression] = [os.path.join(expression_path, file) for file
in os.listdir(expression_path)]
   return expressions
```

Face expressions mapped to integer values:

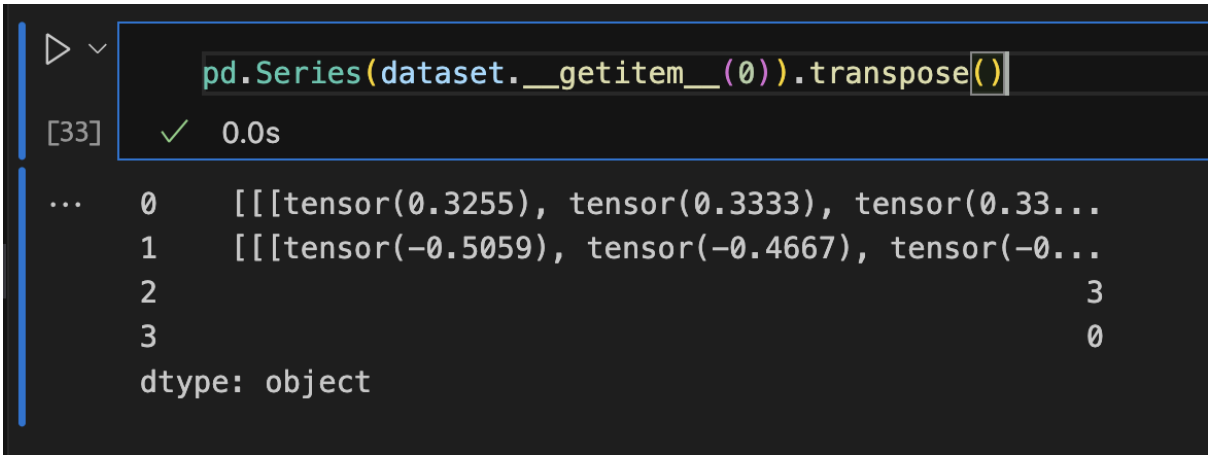
```
self.expression_to_number = {
           'angry': 0,
           'fear': 1,
           'happy': 2,
           'neutral': 3,
           'sad': 4,
           'surprise': 5
       }
```

Logical mapping:

```
expression_to_gesture = {
       'angry': list([0,1,2,3]),
       'fear': list([4,5,6,7]),
       'happy': list([8,9,10,11,12]),
       'neutral': list([13,14,15,16]),
       'sad': list([17,18,19,20]),
       'surprise': list([21,22,23,24])
   }
```

Firstly, we mapped hand gestures to facial expressions based on sequential assignment for this project purpose. As we are considering the hand gestures as an individual alphabet which has no meaning, we just mapped some set of hand gestures to an angry face, and so on. This logical mapping supports the deep learning model by adding an extra feature as expression for hand gesture data, which complements the model predictions.

Sample Dataset structure:



0 - hand image tensor
1 - face image tensore
2 - hand gesture label
3 - mapped face expression label

Hybrid Fusion Architecture:

```
DualInputCNN(
  (hand_cnn): Sequential(
    (0): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): ReLU()
    (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (face_cnn): Sequential(
    (0): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): ReLU()
    (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (fc): Sequential(
    (0): Linear(in_features=6272, out_features=512, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.2, inplace=False)
    (3): Linear(in_features=512, out_features=128, bias=True)
  )
  (fc_hand): Sequential(
    (0): Linear(in_features=128, out_features=25, bias=True)
  )
  (fc_face): Sequential(
    (0): Linear(in_features=128, out_features=6, bias=True)
  )
```

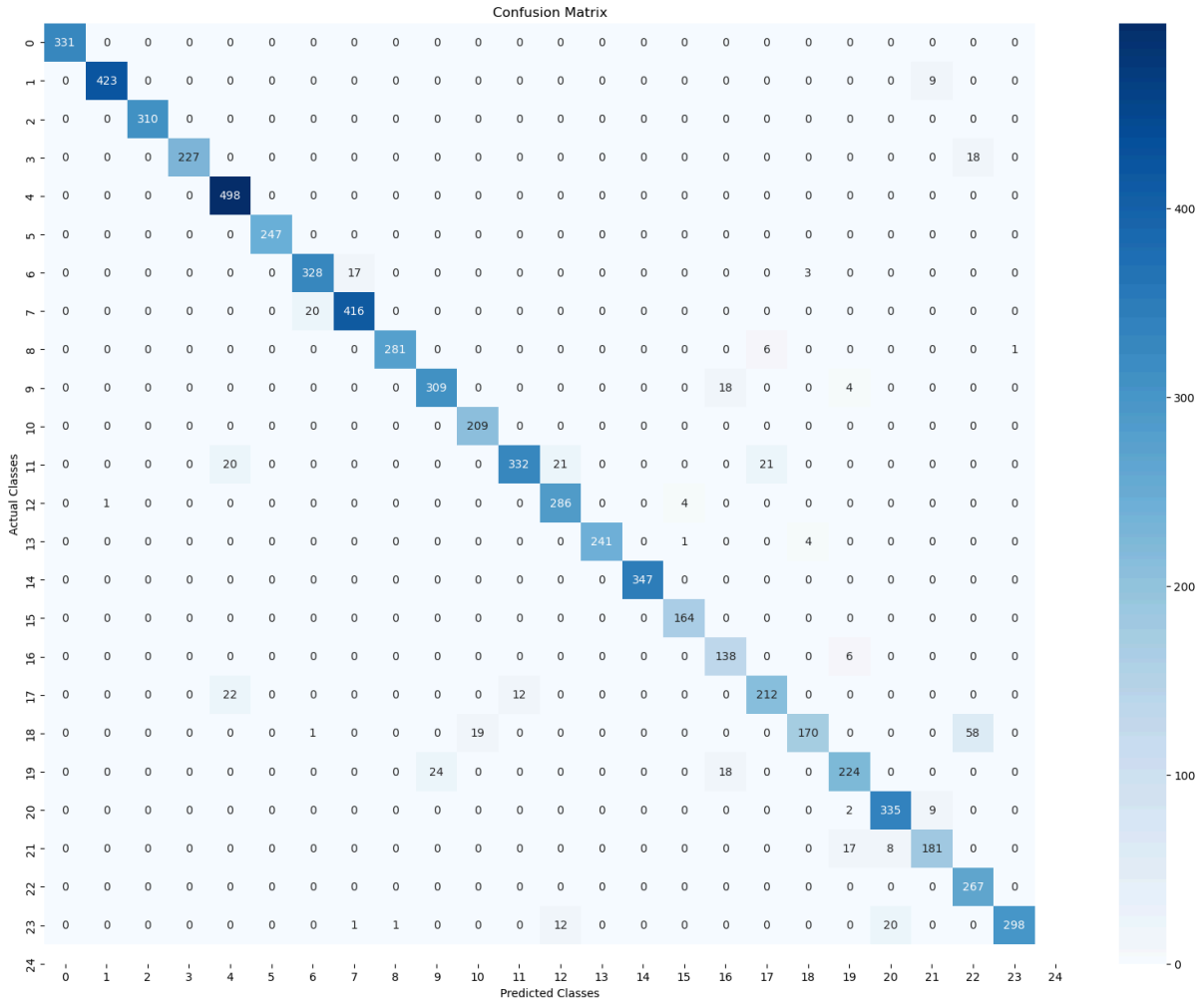
(dropout2): Dropout(p=0.2, inplace=False)
(dropout5): Dropout(p=0.5, inplace=False)
)

Hand gesture evaluation classification report:

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	331
1	1.00	0.98	0.99	432
2	1.00	1.00	1.00	310
3	1.00	0.93	0.96	245
4	0.92	1.00	0.96	498
5	1.00	1.00	1.00	247
6	0.94	0.94	0.94	348
7	0.96	0.95	0.96	436
8	1.00	0.98	0.99	288
10	0.93	0.93	0.93	331
11	0.92	1.00	0.96	209
12	0.97	0.84	0.90	394
13	0.90	0.98	0.94	291
14	1.00	0.98	0.99	246
15	1.00	1.00	1.00	347
16	0.97	1.00	0.98	164
17	0.79	0.96	0.87	144
18	0.89	0.86	0.87	246
19	0.96	0.69	0.80	248
20	0.89	0.84	0.86	266
21	0.92	0.97	0.94	346
22	0.91	0.88	0.89	206
23	0.78	1.00	0.88	267
24	1.00	0.90	0.94	332
accuracy			0.94	7172
macro avg	0.94	0.94	0.94	7172
weighted avg	0.95	0.94	0.94	7172

Confusion Matrix:



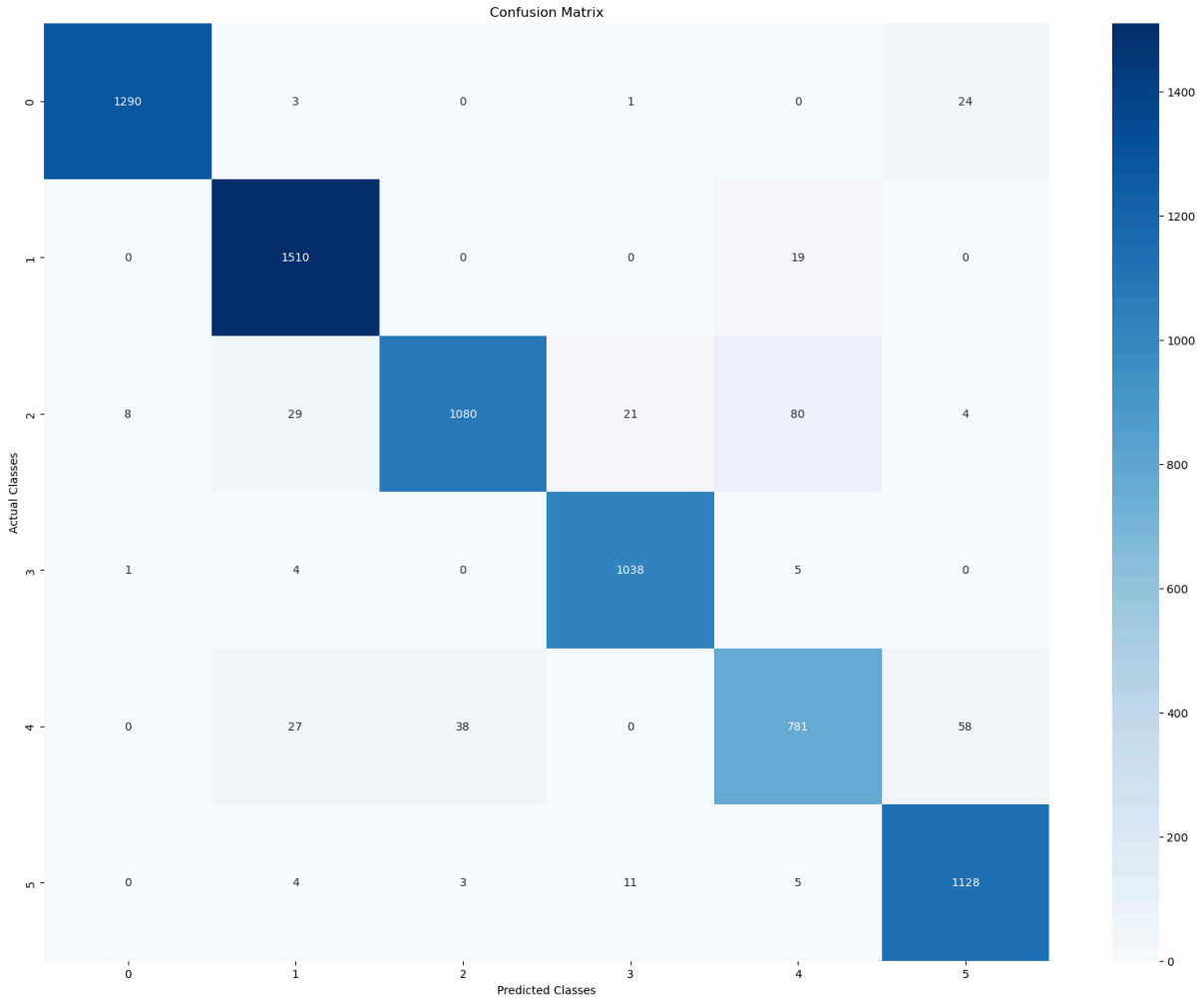
Accuracy Score for Hand gesture recognition:

```
0.9445064138315672
```

Classification report for facial expressions:

```
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      1318
           1       0.96      0.99      0.97      1529
           2       0.96      0.88      0.92      1222
           3       0.97      0.99      0.98      1048
           4       0.88      0.86      0.87       904
           5       0.93      0.98      0.95      1151

    accuracy                           0.95      7172
   macro avg       0.95      0.95      0.95      7172
weighted avg       0.95      0.95      0.95      7172
```

Confusion Matrix:



Accuracy Score for Face expression recognition:
0.9518962632459564

Hyperparameter Tuning:

Learning Rate scheduling:

```
scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=3, gamma=0.9)
```

Early Stopping:

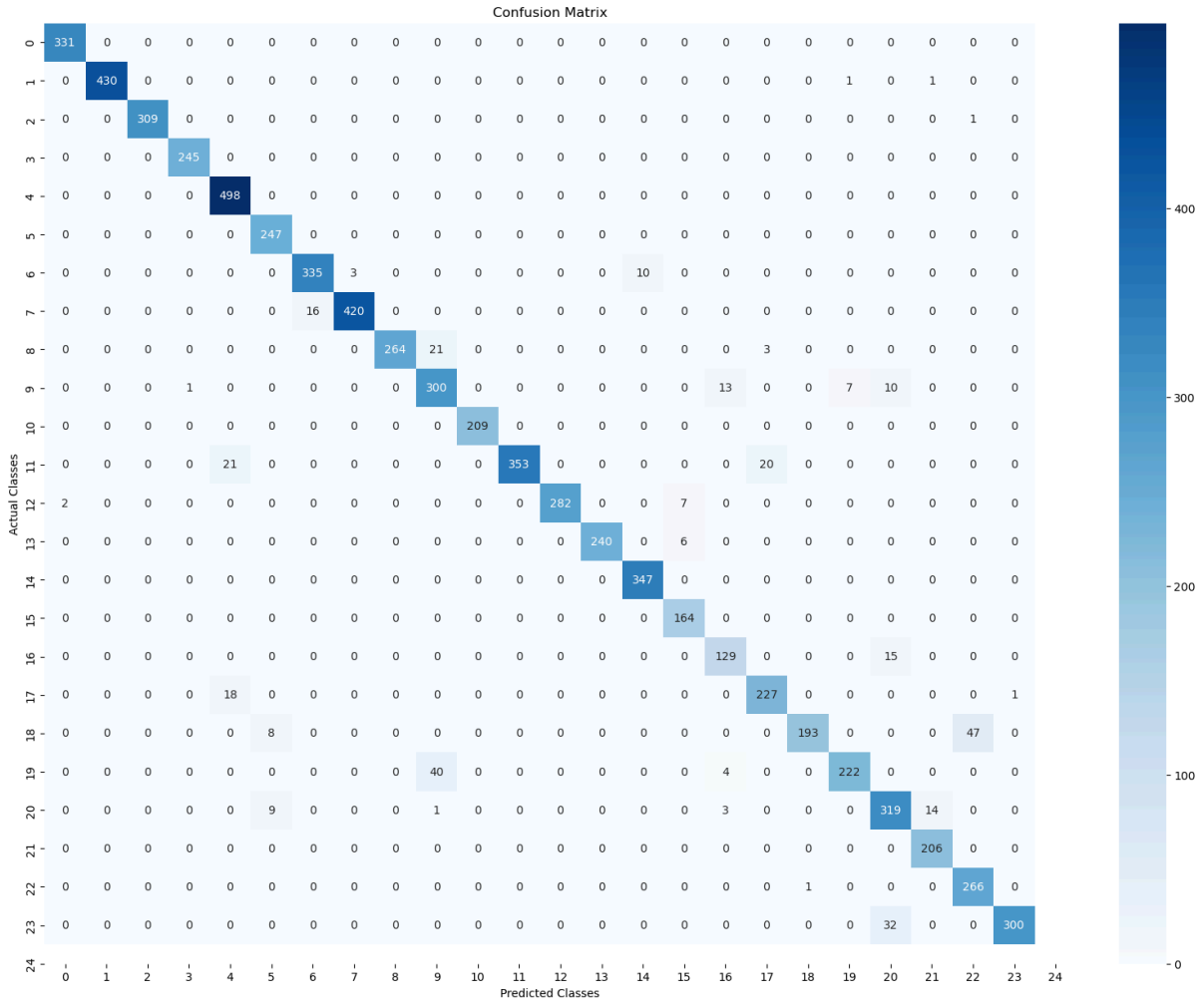
```
if epoch_loss < best_loss:
            best_loss = epoch_loss
            torch.save(lr_model.state_dict(), 'best_fusion_model.pth')
```

Optimized Hand gesture evaluation classification report:

```
Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      1.00       331
```

1	1.00	1.00	1.00	432
2	1.00	1.00	1.00	310
3	1.00	1.00	1.00	245
4	0.93	1.00	0.96	498
5	0.94	1.00	0.97	247
6	0.95	0.96	0.96	348
7	0.99	0.96	0.98	436
8	1.00	0.92	0.96	288
10	0.83	0.91	0.87	331
11	1.00	1.00	1.00	209
12	1.00	0.90	0.95	394
13	1.00	0.97	0.98	291
14	1.00	0.98	0.99	246
15	0.97	1.00	0.99	347
16	0.93	1.00	0.96	164
17	0.87	0.90	0.88	144
18	0.91	0.92	0.92	246
19	0.99	0.78	0.87	248
20	0.97	0.83	0.90	266
21	0.85	0.92	0.88	346
22	0.93	1.00	0.96	206
23	0.85	1.00	0.92	267
24	1.00	0.90	0.95	332
accuracy			0.95	7172
macro avg	0.95	0.95	0.95	7172
weighted avg	0.96	0.95	0.95	7172

Confusion Matrix:

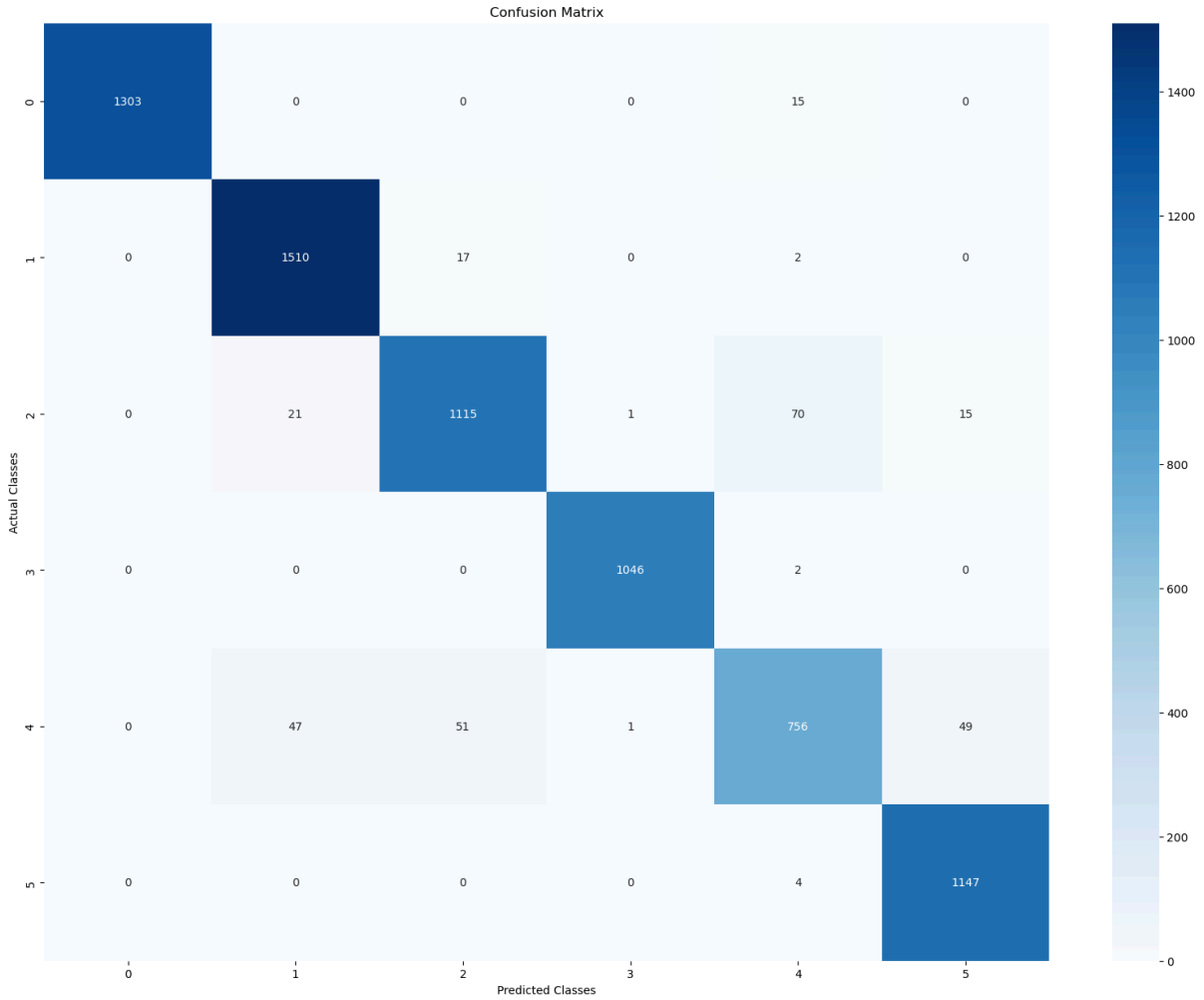



```
Accuracy Score:
0.9531511433351925
```

Optimized Facial expression evaluation classification report:

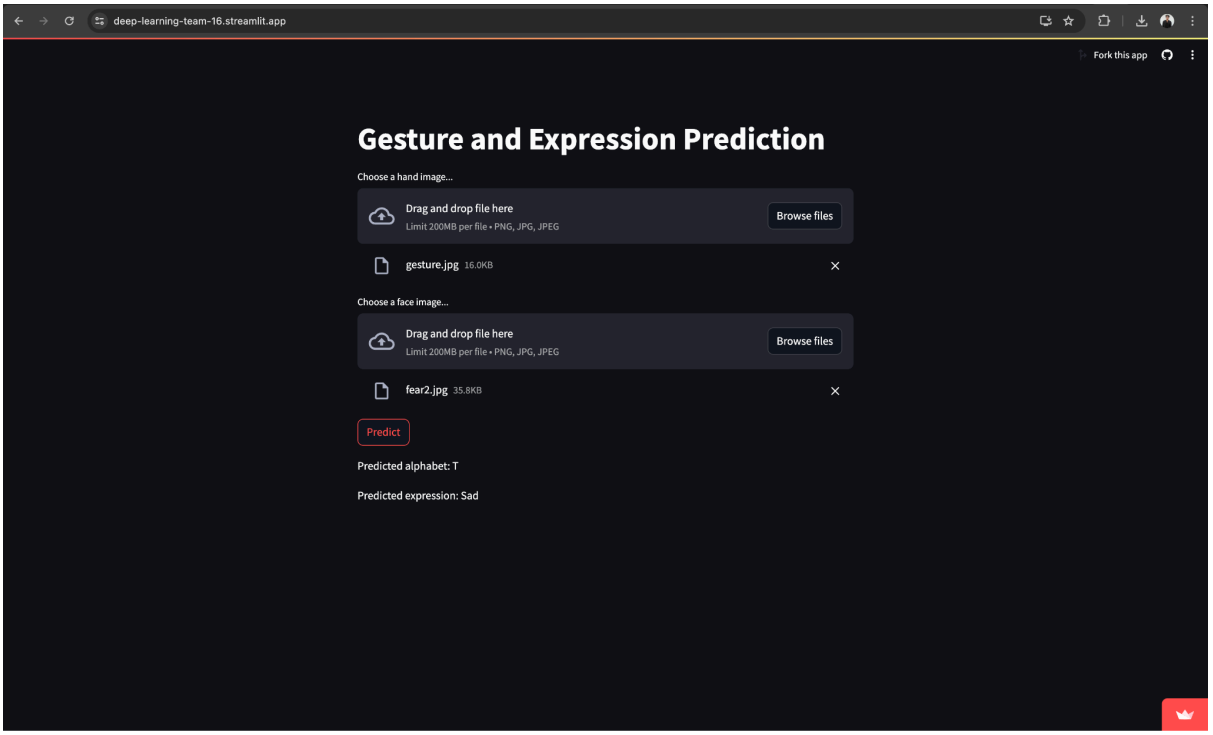
```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      1318
           1       0.96      0.99      0.97      1529
           2       0.94      0.91      0.93      1222
           3       1.00      1.00      1.00      1048
           4       0.89      0.84      0.86       904
           5       0.95      1.00      0.97      1151

    accuracy                           0.96      7172
   macro avg       0.96      0.95      0.95      7172
weighted avg       0.96      0.96      0.96      7172
```

Confusion Matrix:



Accuracy Score:
0.9588678192972672

Deployment:



Link to the streamlit app: https://deep-learning-team-16.streamlit.app/

References:

- https://matplotlib.org/
- https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
- https://pytorch.org/docs/stable/generated/torch.nn.RNN.html
- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10486648/#:~:text=The%20KNN%20(K%20nearest%20neighbor,using%20the%20up%2Dsampling%20technique.
- https://ieeexplore.ieee.org/document/8265571
- https://arxiv.org/abs/2401.03828
- https://ieeexplore.ieee.org/document/9862878

Contributions:

Team Member	Project Part	Contribution (%)
Charan Kumar Nara	• Data cleaning. • Data Preprocessing. • Data Analysis. • Decide on the model architecture and parameters. • Training and Testing • Evaluate the trails. • Hyperparameter tuning. • Deploy the Model.	50%

Sri Sahith Reddy Kuncharam	• Data cleaning. • Data Preprocessing. • Data Analysis. • Decide on the model architecture and parameters. • Training and Testing. • Hyperparameter tuning. • Evaluate the trails. • Deploy the Model.	50%