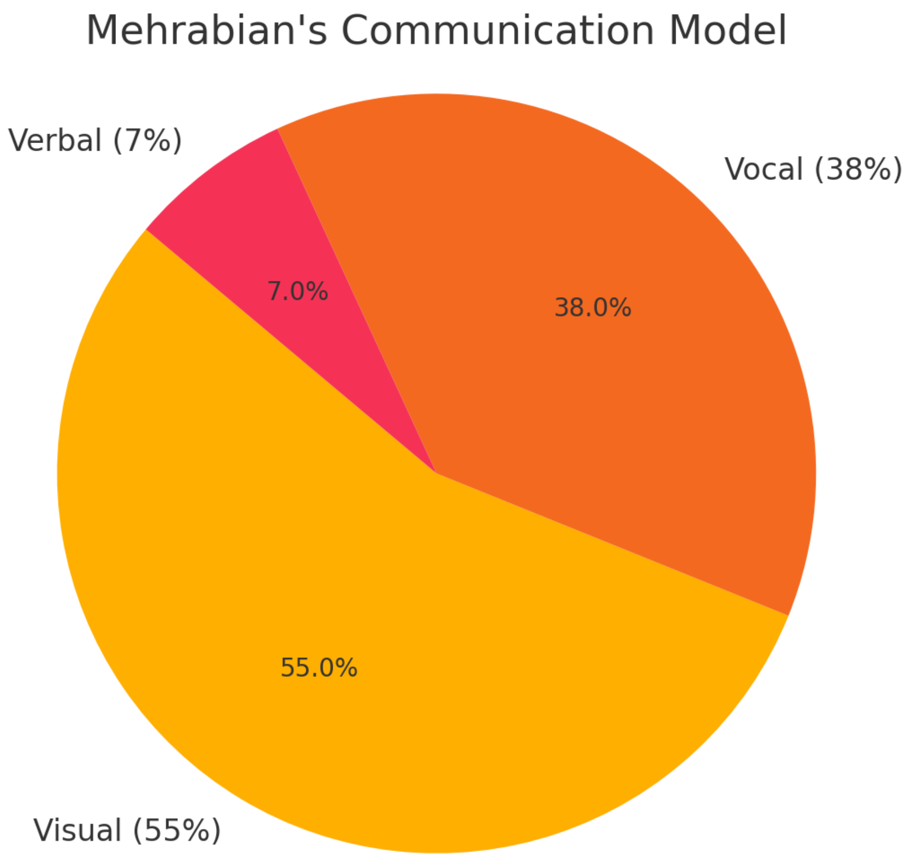
Emotion Recognition in Videos Through Deep Neural Network Models

By: Koushik Roy

Why Emotion Recognition

- Emotion is vital in social and human-robot interaction
- Study: 55% visual, 38% vocal, 7% verbal
- Facial cues (e.g., smile, brow movement) = key indicators
- Challenge: Emotion recognition in videos



Problem Statement

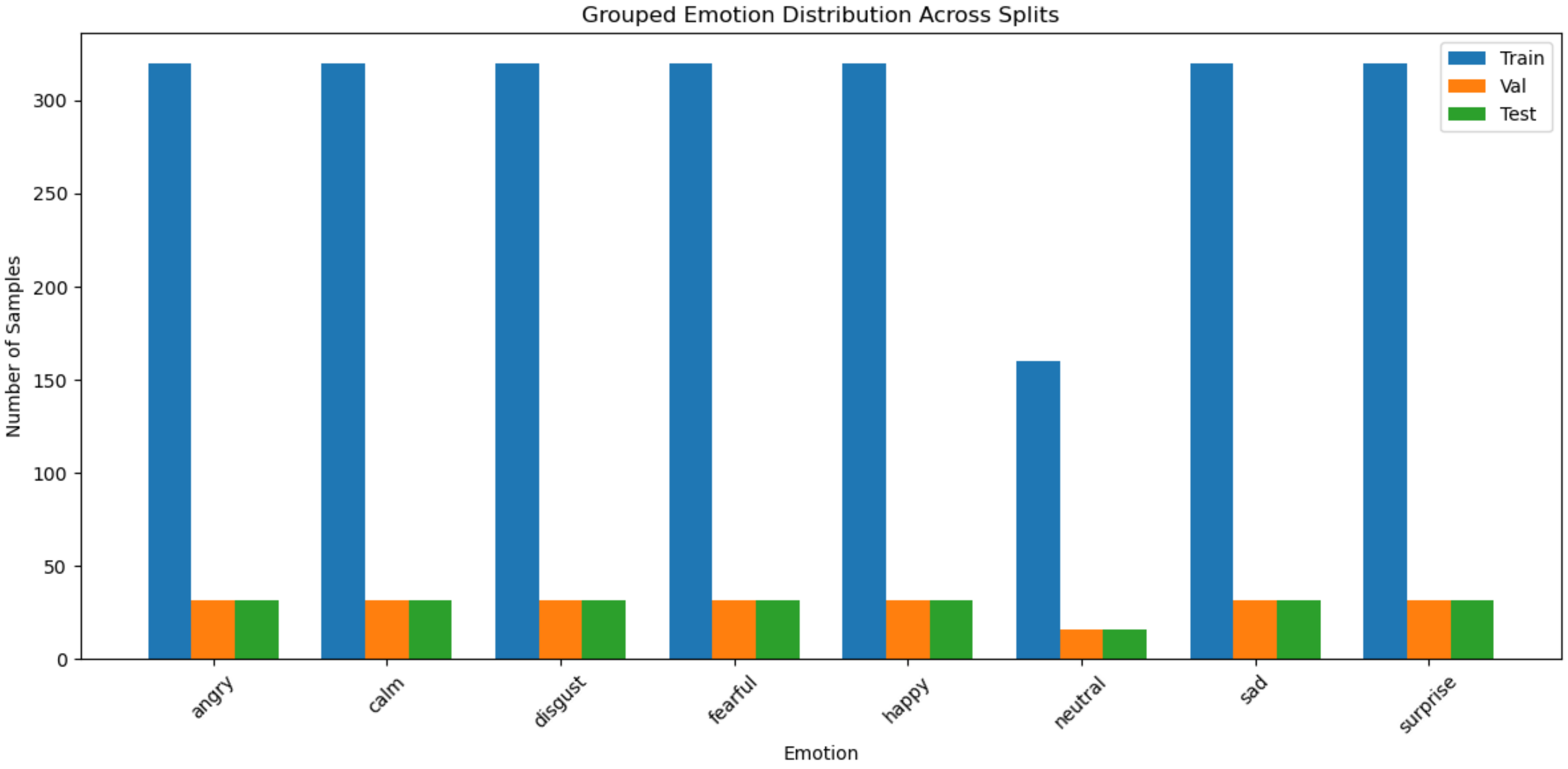
- Input: 4D tensor from video: T (frames) × C (channels) × H × W
- Output: Predicted emotion class (e.g., happy, sad, etc.)
- Goal: Test deep learning models to improve accuracy
- Focus: Visual features only

Dataset Overview

- Ryerson Audio-Visual Database of Emotional Speech and Song
- Emotions (8): calm, happy, sad, angry, fearful, surprise, disgust, neutral
- Each emotions has 2 emotional intensity: Normal and Strong
- Except for Neutral, it has just one emotional intensity. That’s why the number of samples are half of the other emptions.

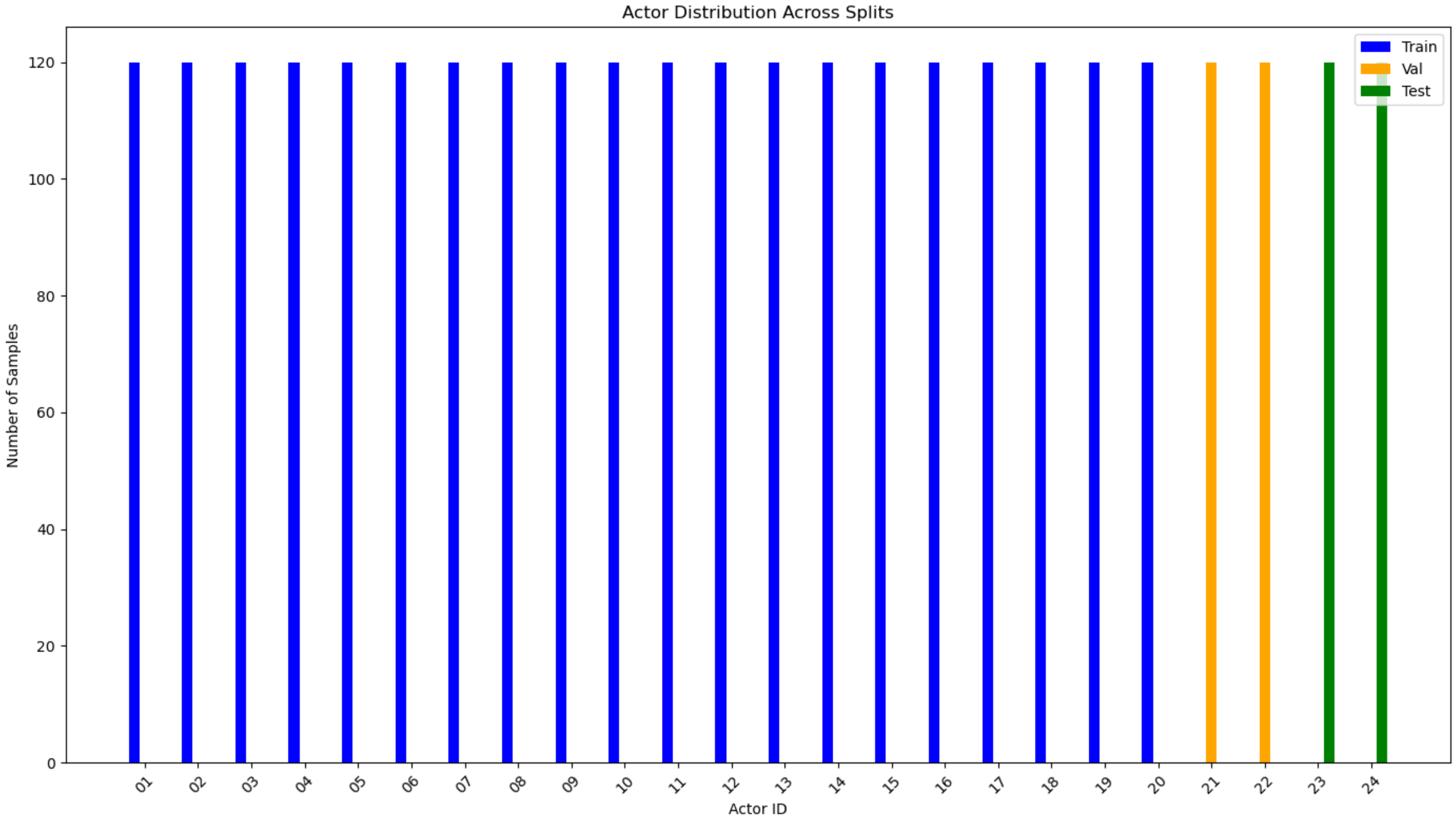
Dataset Stats

- 1,400+ clips, 24 actors, 1280×720 resolution
- Train/Val/Test split: 80/10/10
- For Train we have 320 samples per emotion (except for neutral which has half 160)
- Total Train Samples = 320*7 + 160 = 2400
- For Validation and Test we have 32 samples per emotion (neutral has 16) and 240 in total each split



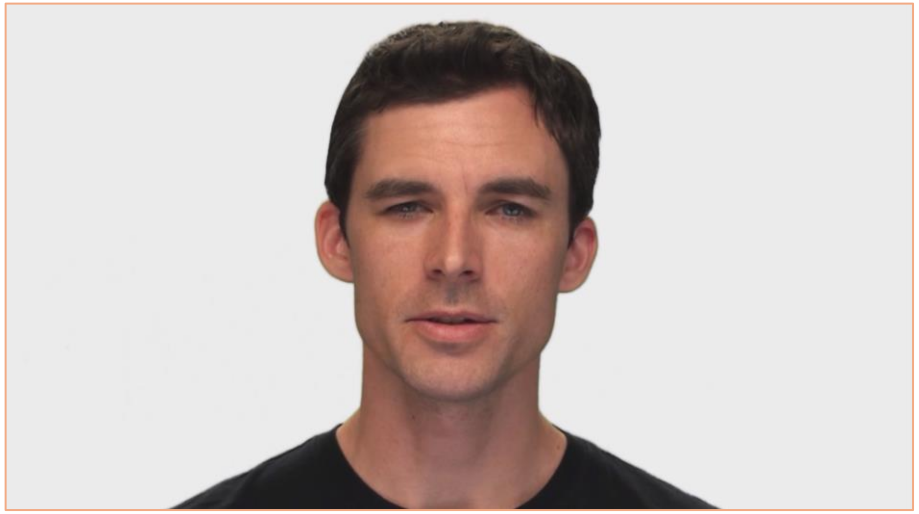
Dataset Stats

- 1,400+ clips, 24 actors, 1280×720 resolution
- Train/Val/Test split: 80/10/10
- Train: Actor 1 – 20
- Val: Actor 21, 22
- Test: Actor 23, 24
- Odd: Male, Even: Female
- Per actor 120 Samples



Data Preprocessing

- Unzip each video file
- Extract the emotion class and actor data
- Read video and extract frames
- Intelligent resizing
  - Resize to shorter size (256 by 256) + CenterCrop(224 by 224)
- Extracting 16 frames from each video in the dataset



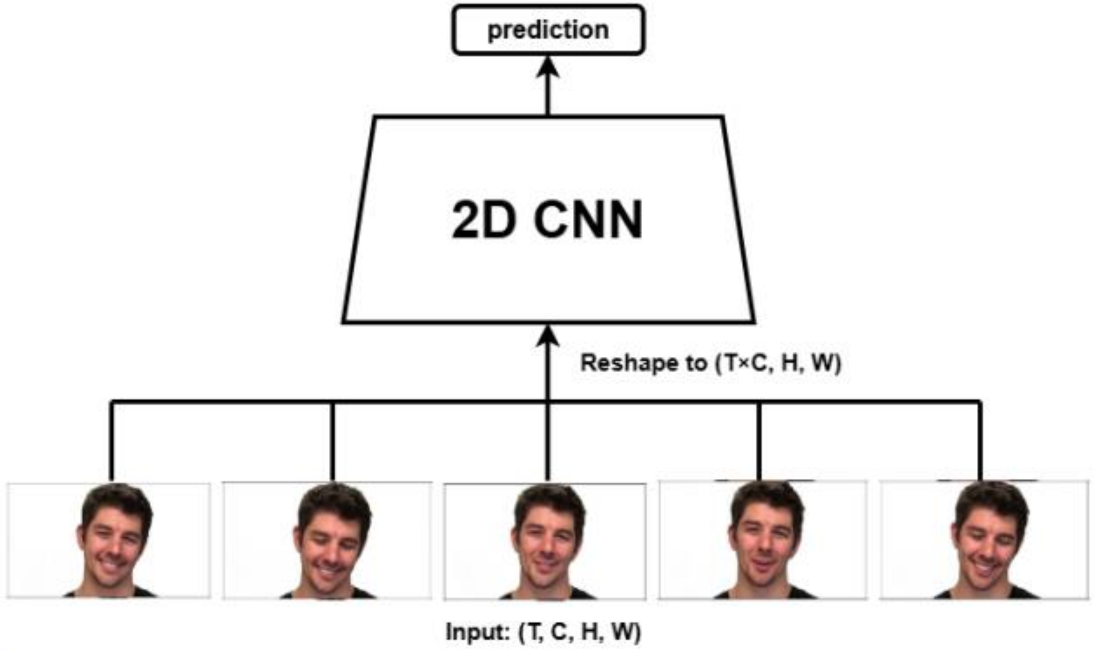
Input Frame

Naive resizing

Intelligent resizing

Early Fusion Model

- Reshape 4D input to 3D by merging T and C → use 2D CNN
- Simple architecture, quick to train
- Captures very shallow temporal features
- Performs moderately on accuracy
- Used as baseline



Early Fusion Model Architecture 1 (2d CNN, not pretrained)

- **Input Shape:** A clip of shape `(B, T, C, H, W)` — batch of `T` RGB frames.
- **Early Fusion:** Frames are fused by reshaping to `(B, T*C, H, W)` before passing through the CNN.
- **Convolutional Layers:**
    - `Conv2d` layers with increasing filters: 64 → 128 → 256 → 512.
    - Each followed by `BatchNorm2d` and ReLU activation.
- **Global Average Pooling:** Reduces spatial dimensions to 1×1.
- **Fully Connected Layers:**
    - `512 → 1024 → num_classes` (with dropout and ReLU in between).
- **Output:** Final logits for classification across `num_classes`.

Early Fusion Model Architecture 1 (2d CNN, not pretrained) [Results]

Table: Test accuracy of the early fusion model (2d CNN)

Frames	6	10	16
No Augment	16.67%	21.67%	21.67%
Augment	28.33%	26.67%	28.33%

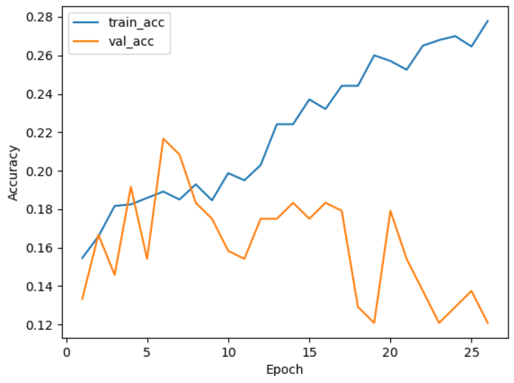
Early Fusion Model Architecture 1 (2d CNN, not pretrained) [Results]

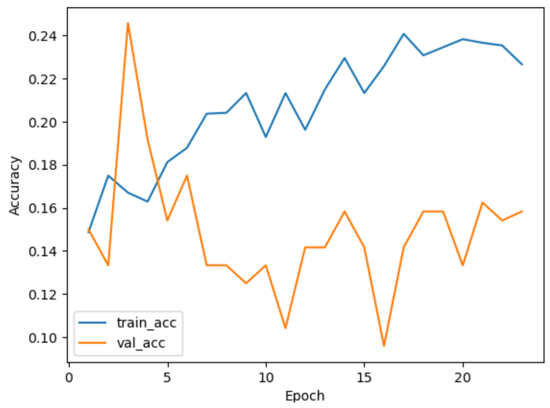
Frames 6

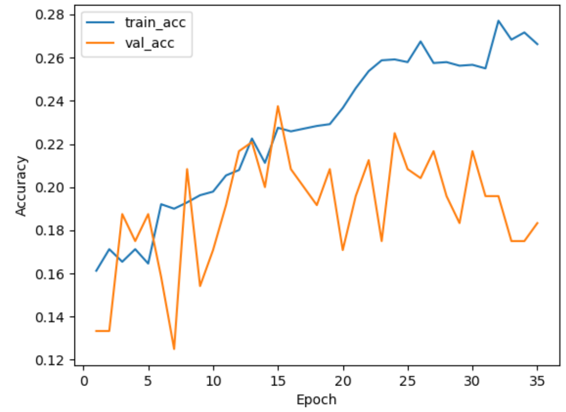
Frames 10

Frames 16

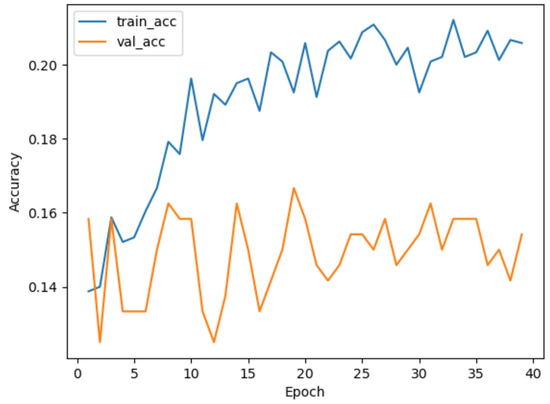
No Augment

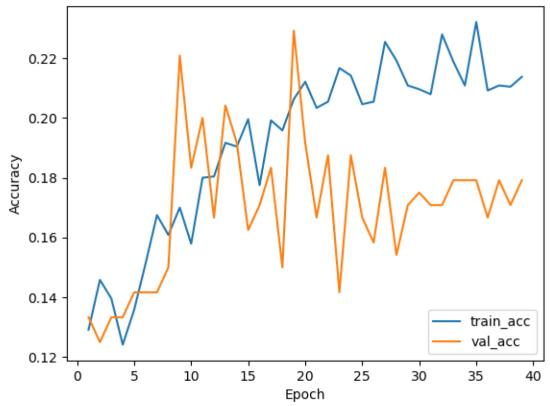


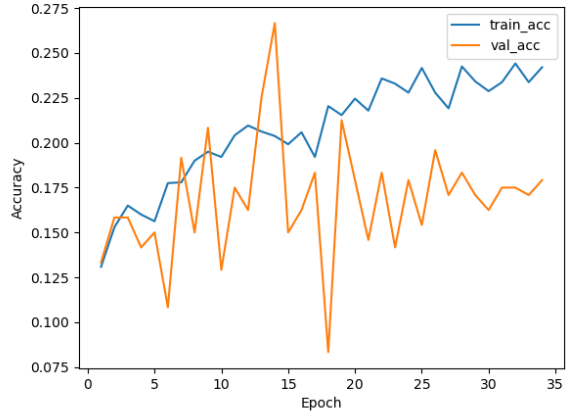




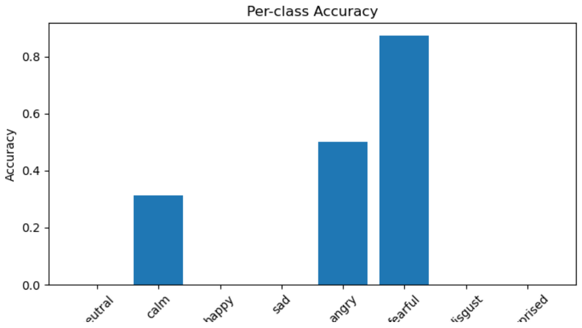
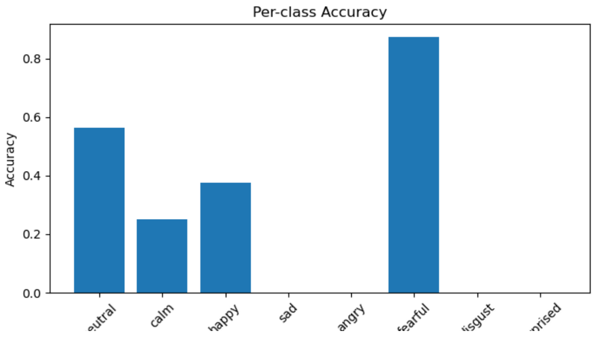
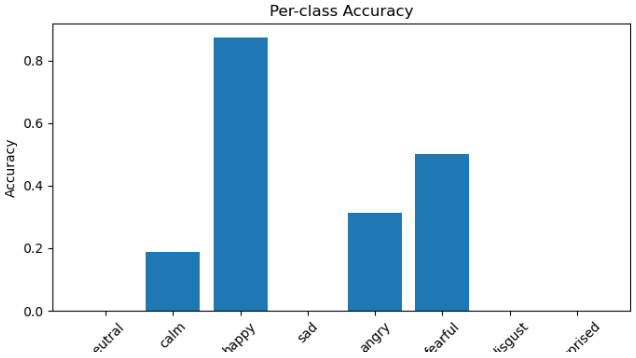
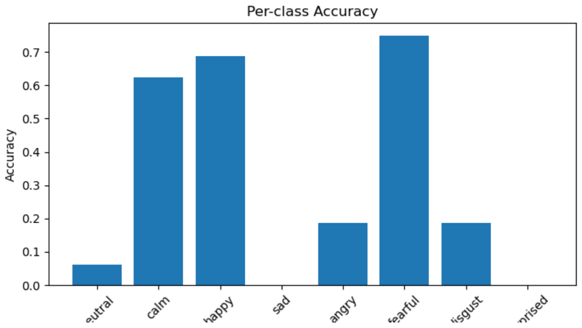
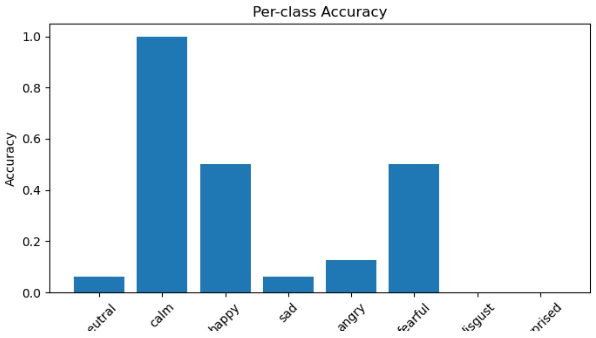
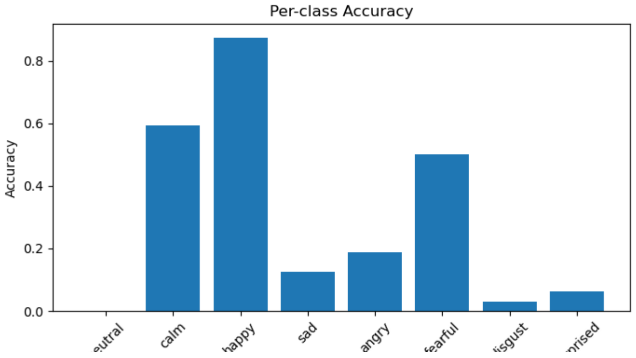
Augment







Early Fusion Model Architecture 1 (2d CNN, not pretrained) [Results]

	Frames 6	Frames 10	Frames 16
No Augment			
Augment			

Early Fusion Model Architecture 2 (2d Resnet, pretrained)

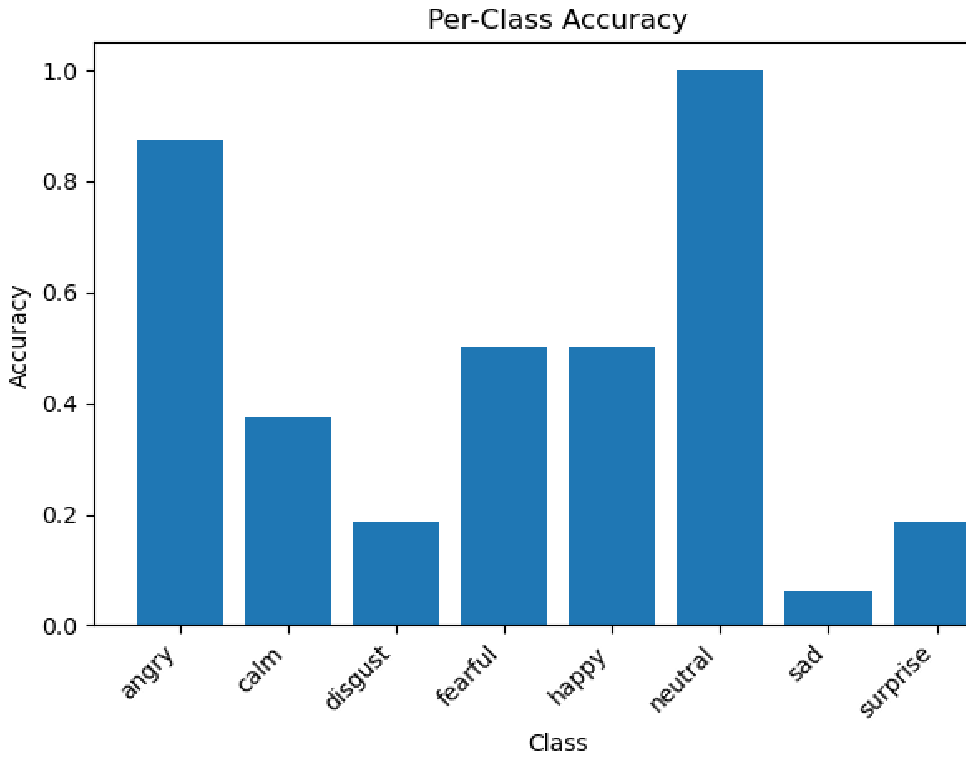
- **Input Shape:** `(B, 3*num_frames, H, W)` — early-fused RGB frames stacked along the channel dimension.
- **Backbone:** Modified **ResNet-18**:
  - Replaces the first `conv1` layer with a `Conv2d` accepting `3*num_frames` input channels instead of 3.
  - Uses all layers from ResNet-18 **except** the original `conv1` and final `fc` layer.
- **Feature Extractor Output:** 512-dimensional feature vector after global average pooling.
- **Final Layer:** A custom `Linear(512, num_classes)` layer for classification.
- **Pretrained Weights:** Initializes from ImageNet pretrained ResNet-18 (except the replaced layers).

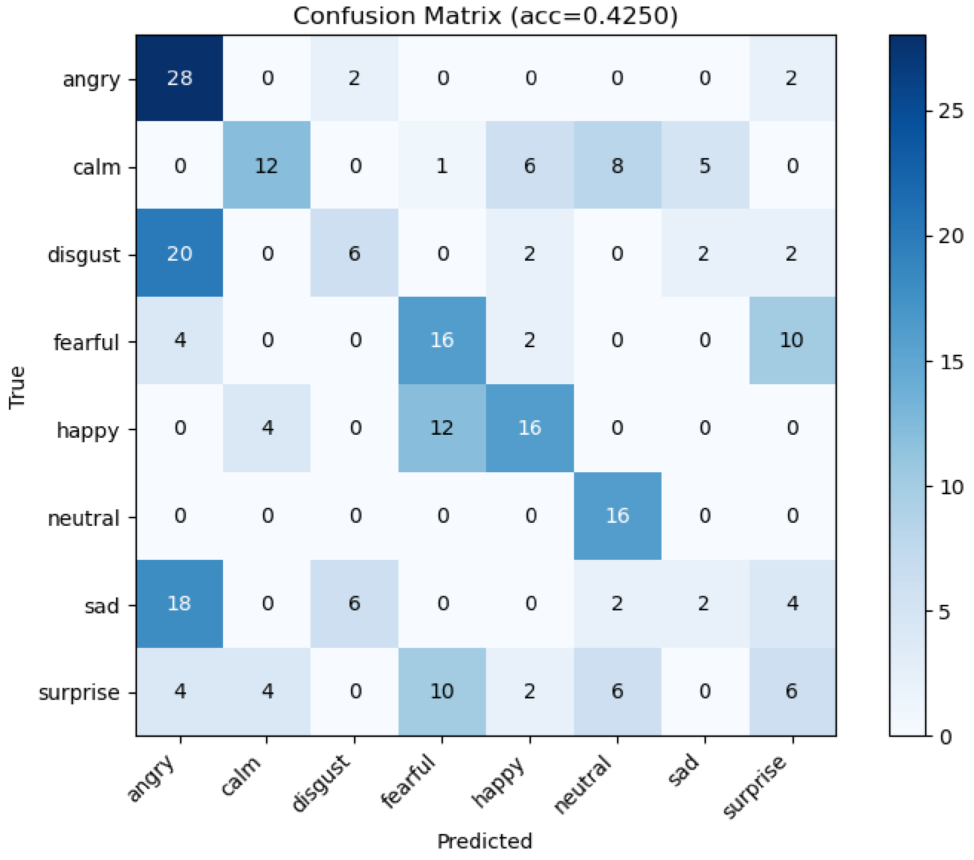
Early Fusion Model Architecture 2 (2d Resnet, pretrained) [Results]

Frames 10

Augment

➢ Test Accuracy: 42.5%



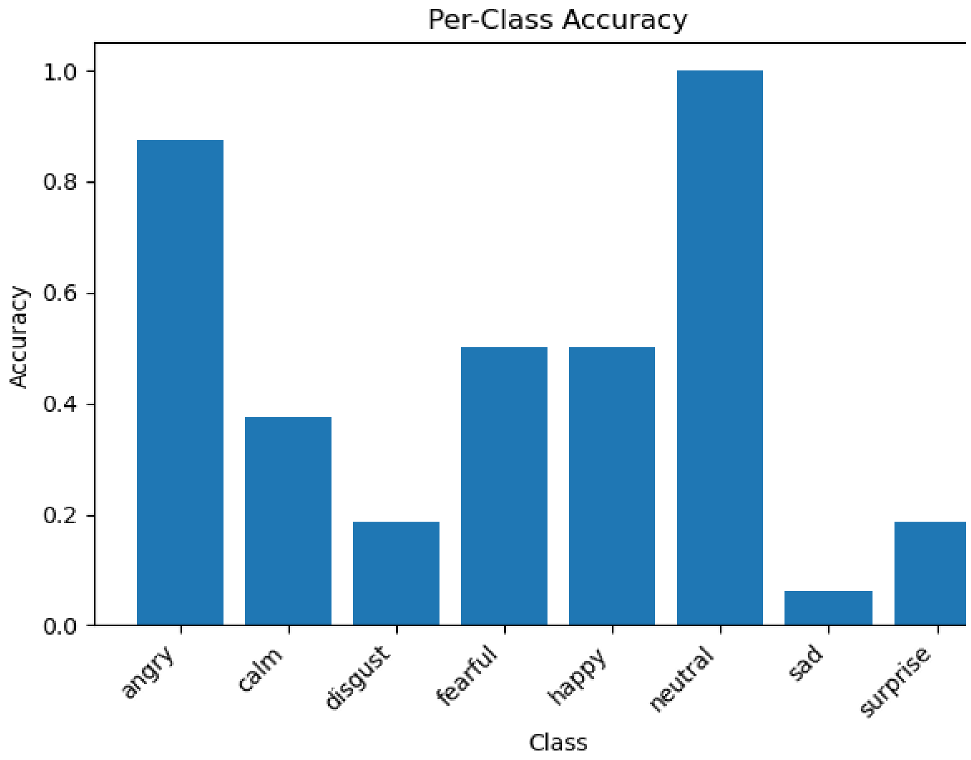


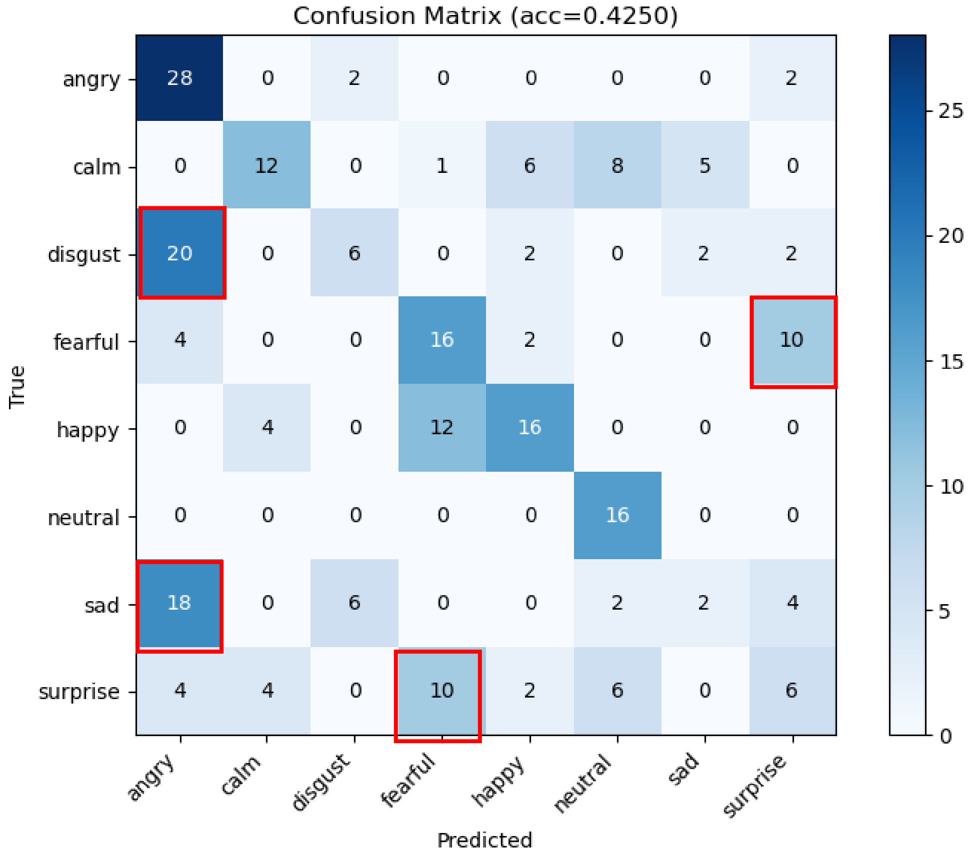
Early Fusion Model Architecture 2 (2d Resnet, pretrained) [Results]

Frames 10

Augment

- Test Accuracy: 42.5%

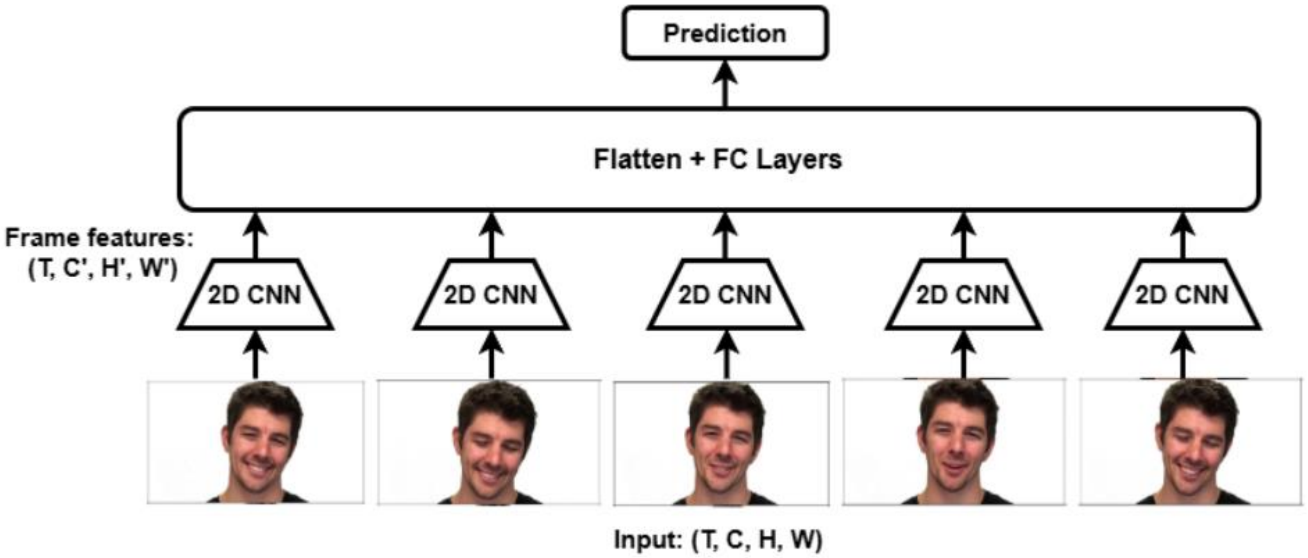




True \ Predicted	angry	calm	disgust	fearful	happy	neutral	sad	surprise
angry	28	0	2	0	0	0	0	2
calm	0	12	0	1	6	8	5	0
disgust	20	0	6	0	2	0	2	2
fearful	4	0	0	16	2	0	0	10
happy	0	4	0	12	16	0	0	0
neutral	0	0	0	0	0	16	0	0
sad	18	0	6	0	0	2	2	4
surprise	4	4	0	10	2	6	0	6

Late Fusion Model

- ➢ Each frame goes through 2D CNN → feature extraction
- ➢ Features concatenated and passed through FC layers
- ➢ Better modeling of temporal features than early fusion
- ➢ More flexible to noise or irrelevant frames
- ➢ This model achieved better generalization.



Late Fusion Model Architecture (2d Resnet, pretrained)

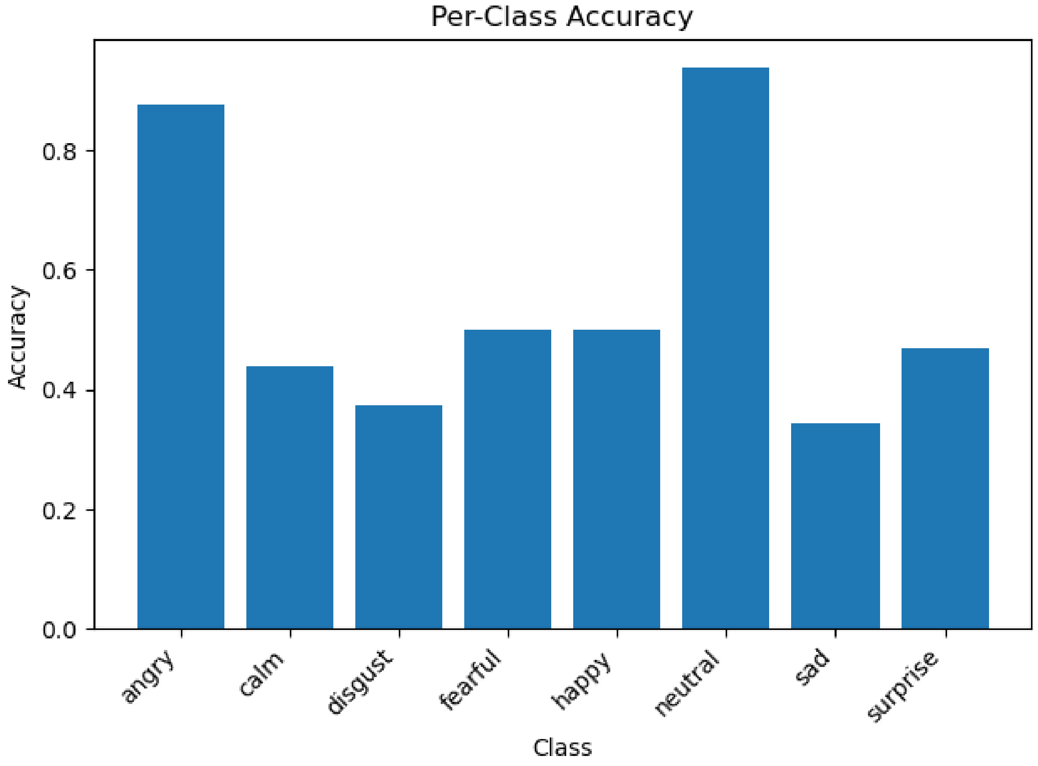
- **Input Shape:** `(B, T, C, H, W)` — a sequence of `T` RGB frames per sample.
- **Frame-Level Feature Extraction:**
  - Each frame is passed individually through a shared **ResNet-18** backbone (excluding the final `fc` layer).
  - Backbone outputs a 512-dimensional feature per frame.
- **Late Fusion:**
  - Frame features of shape `(B, T, 512)` are **averaged across time** to form a single vector of shape `(B, 512)`.
- **Classification Layer:** A `Linear(512, num_classes)` layer maps the fused features to output logits.
- **Pretrained Weights:** Uses ImageNet-pretrained ResNet-18 for frame-level encoding.

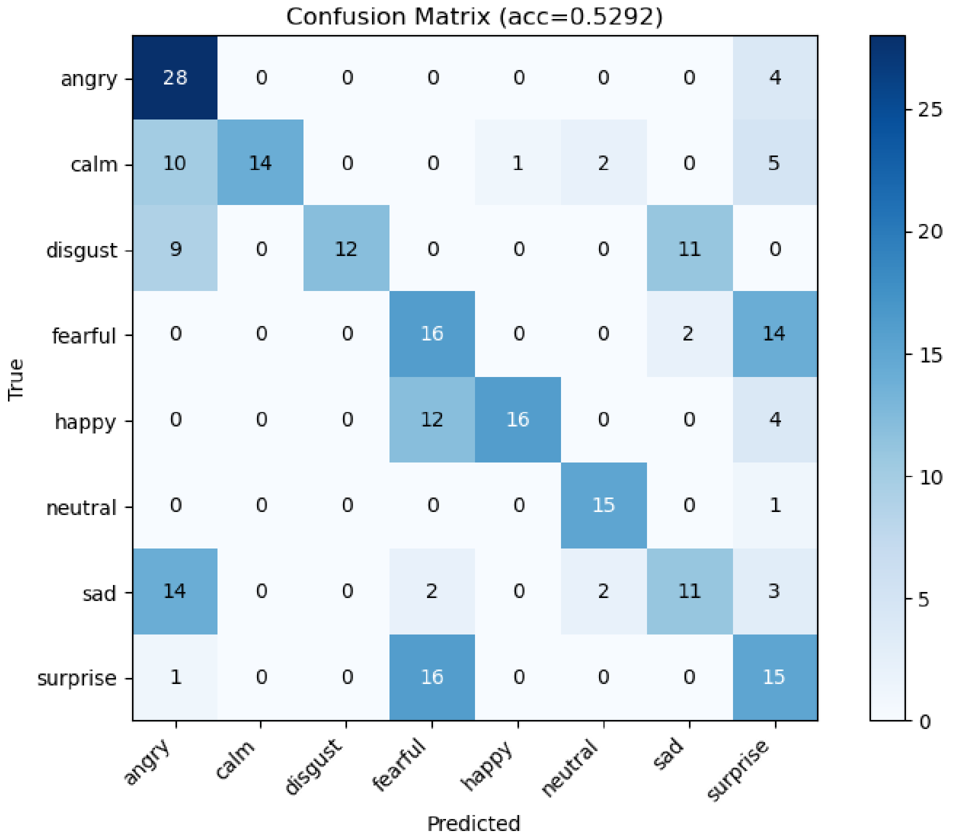
Late Fusion Model Architecture (2d Resnet, pretrained) [Results]

Frames 10

Augment

- Test Accuracy: 52.9%

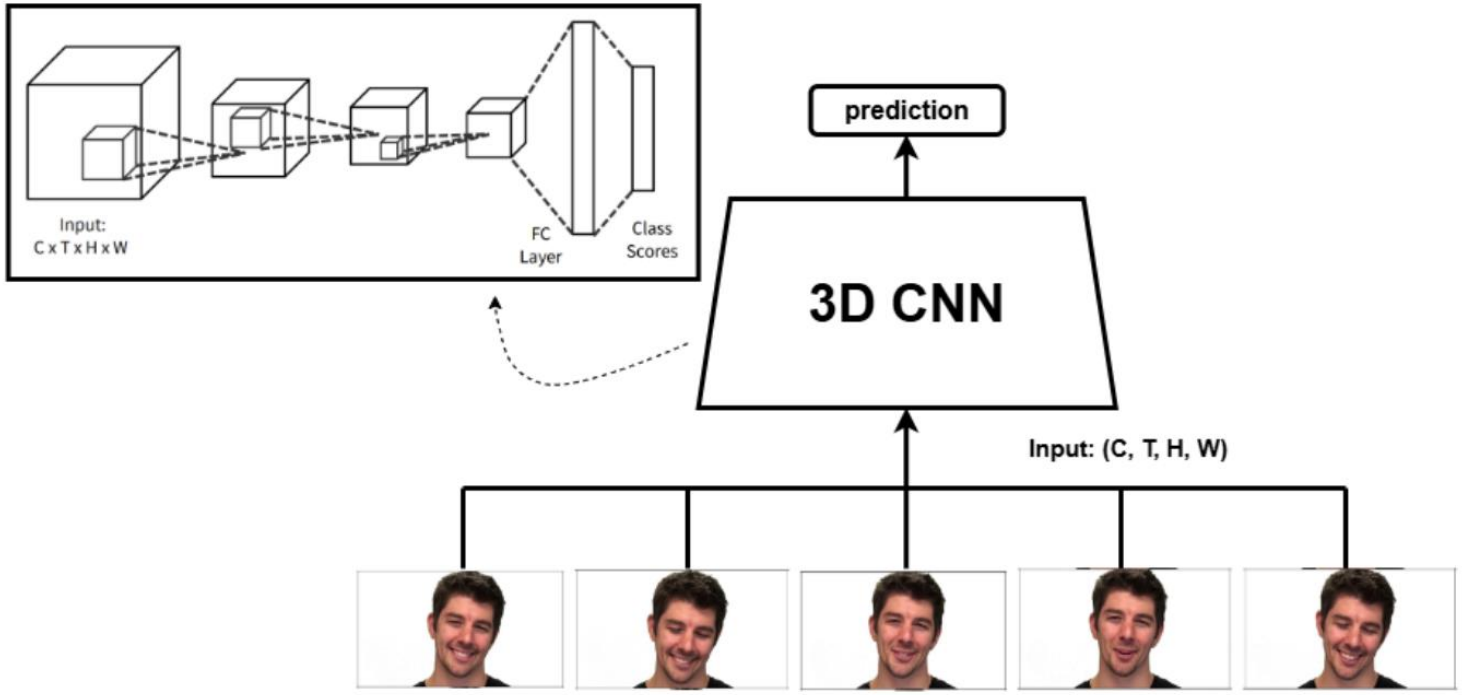




	angry	calm	disgust	fearful	happy	neutral	sad	surprise
angry	28	0	0	0	0	0	0	4
calm	10	14	0	0	1	2	0	5
disgust	9	0	12	0	0	0	11	0
fearful	0	0	0	16	0	0	2	14
happy	0	0	0	12	16	0	0	4
neutral	0	0	0	0	0	15	0	1
sad	14	0	0	2	0	2	11	3
surprise	1	0	0	16	0	0	0	15

3D CNN Model

- Uses 3D kernels to capture spatial & temporal features
- No reshaping or frame-wise separation
- Fully leverages video dynamics
- Visual: 3D CNN diagram
- This model outperformed better than early fusion and late fusion.



3d CNN Model Architecture (3d Resnet)

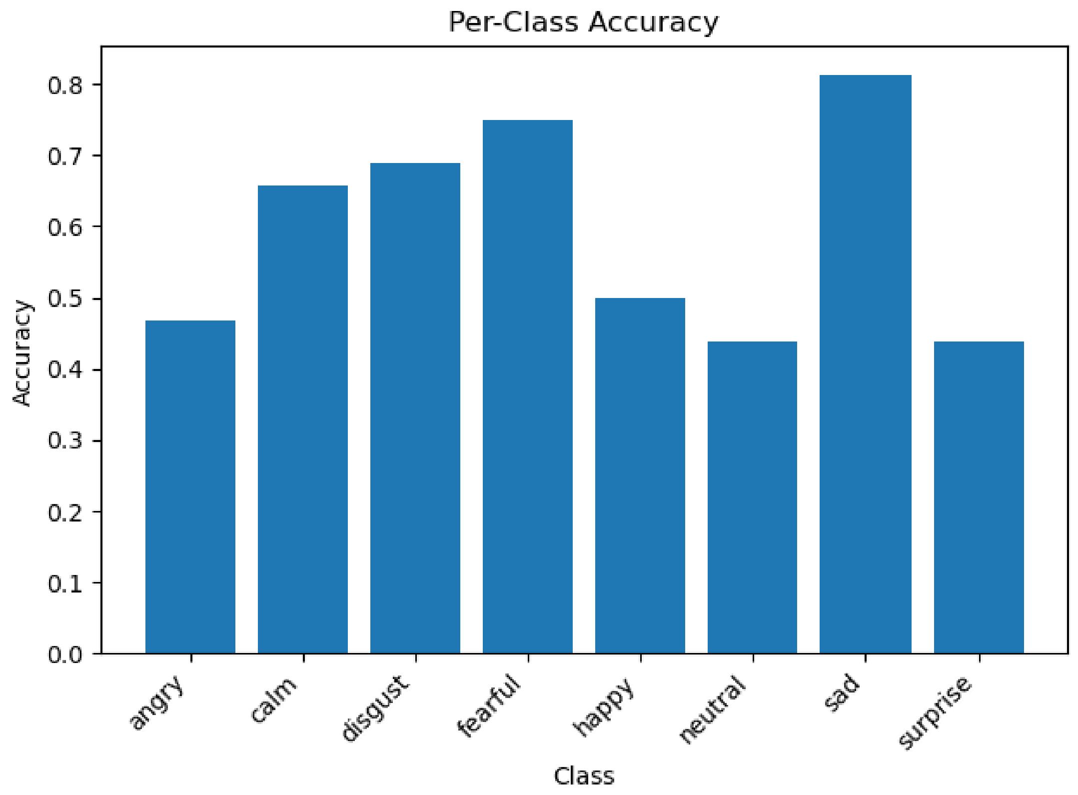
- **Input Shape:** `(B, C, T, H, W)` — batch of video clips with `T` RGB frames.
- **Architecture:** Uses **3D convolutions** (spatiotemporal) with the **pretrained ResNet-18 (r3d_18)** from `torchvision.models.video`.
    - Processes spatial and temporal dimensions **jointly**.
    - Designed to capture motion patterns and appearance features simultaneously.
- **Pretrained Weights:** Initialized from a model pretrained on **Kinetics-400**.
- **Final Layer:** The default classification layer is replaced with a custom `Linear(in_features=512, out_features=num_classes)` layer.

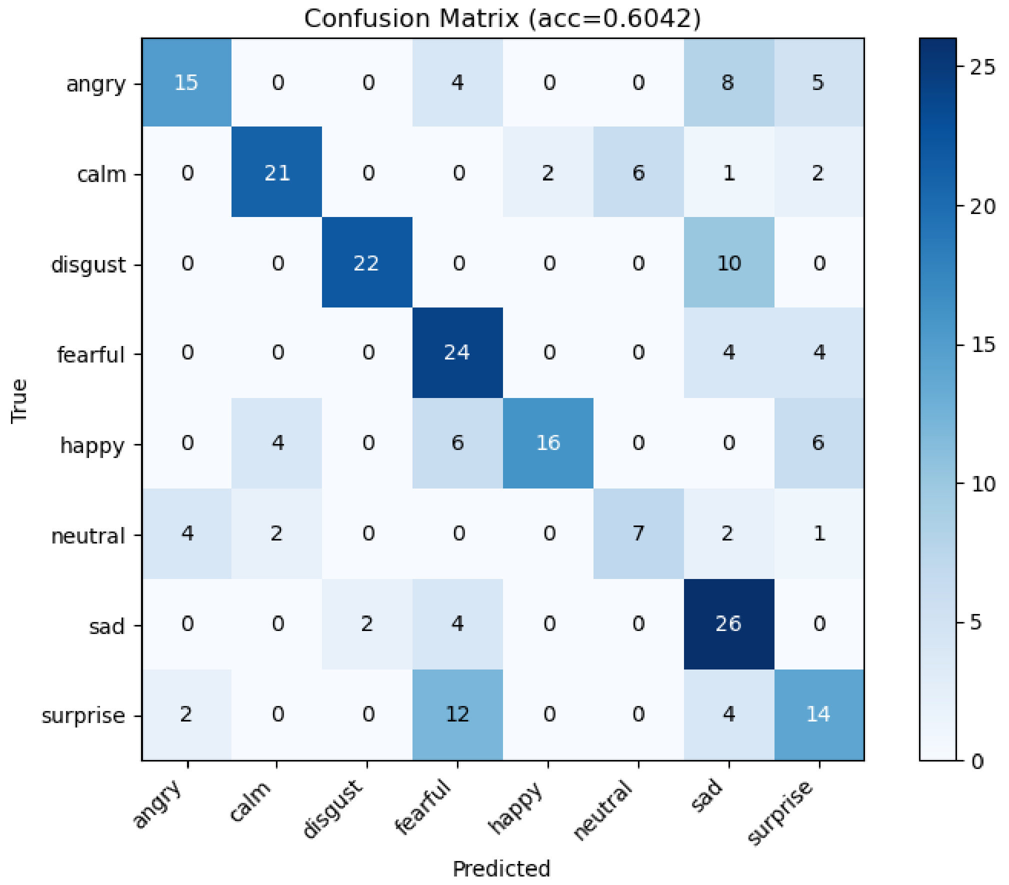
3d CNN Model Architecture (3d Resnet) [Results]

Frames 10

Augment

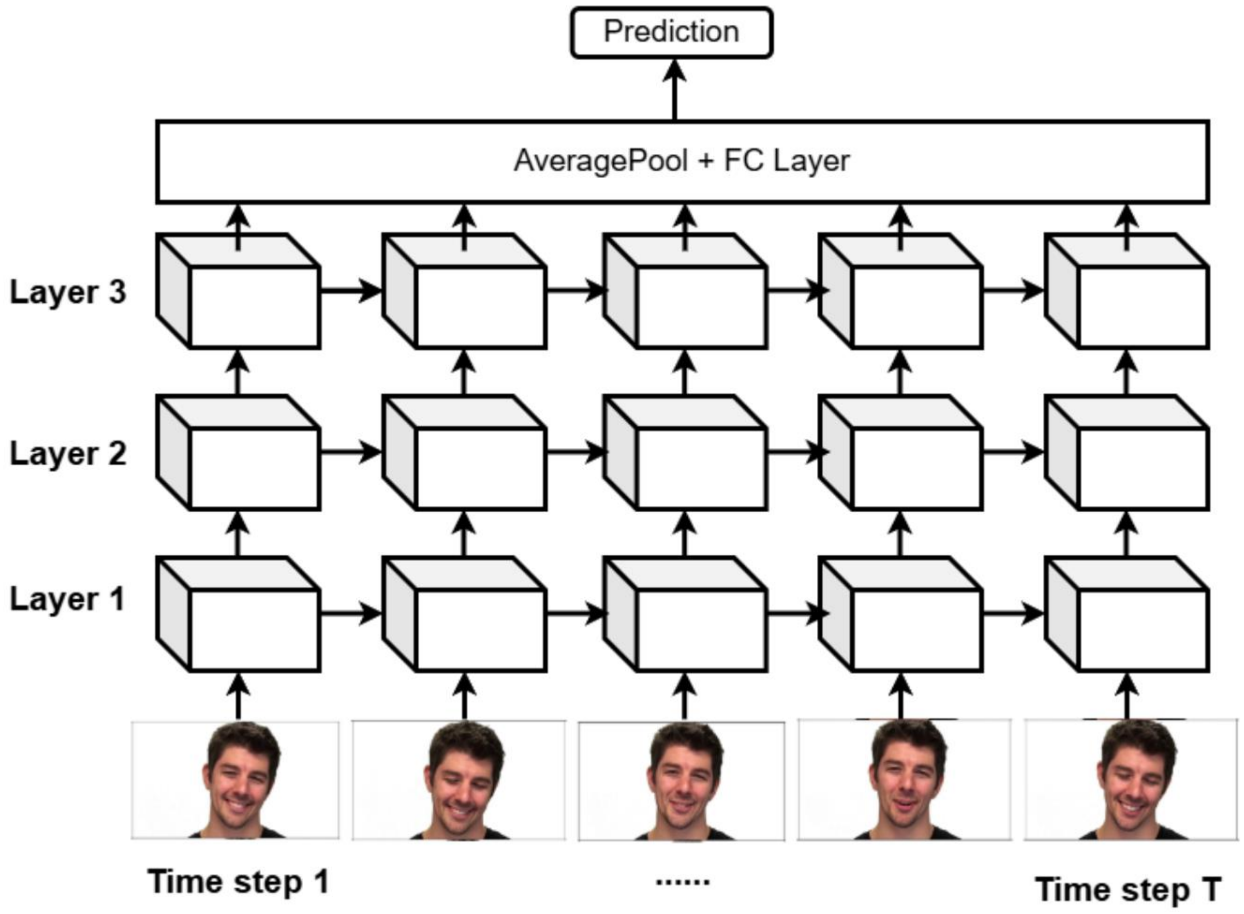
➢ Test Accuracy: 60.4%





CNN-RNN Model

- Uses 2D CNNs + LSTMs per layer
- Takes input from previous time steps and layers
- Targets long-range temporal dependencies



CNN (Resnet) - RNN (LSTM) Model Architecture

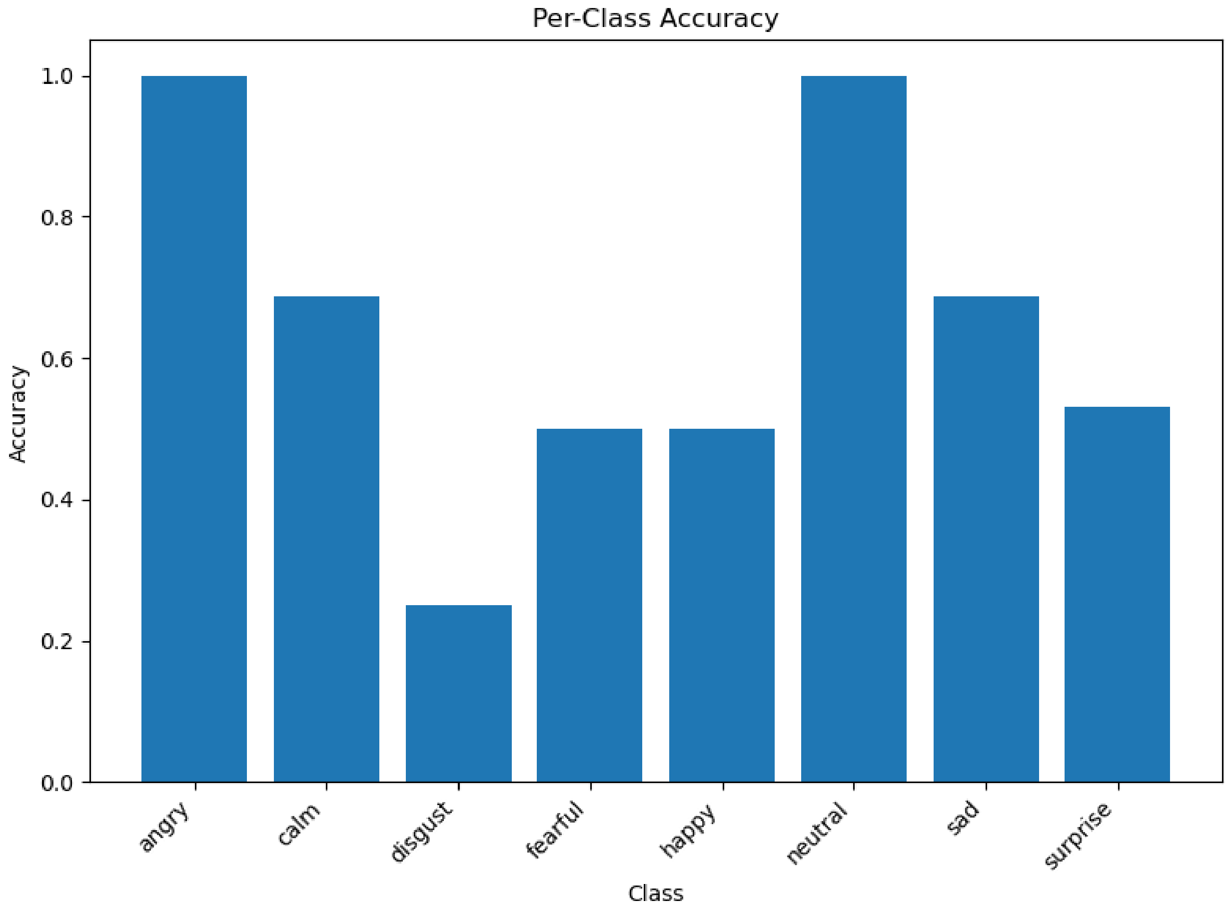
- **Input Shape:** `(B, T, C, H, W)` — a sequence of `T` RGB frames per sample.
- **Frame Feature Extraction:**
  - Each frame is passed through a shared **ResNet-18** backbone (excluding the final `fc` layer).
  - Produces 512-dimensional feature vectors for each frame → resulting in a sequence of shape `(B, T, 512)`.
- **Temporal Modeling:**
  - A single-layer **LSTM** processes the frame-level features across time.
  - The final hidden state (from the last time step) represents the entire sequence.
- **Classification Layer:** A `Linear(hidden_size, num_classes)` layer maps the LSTM output to class logits.
- **Pretrained Weights:** ResNet-18 is pretrained on ImageNet.

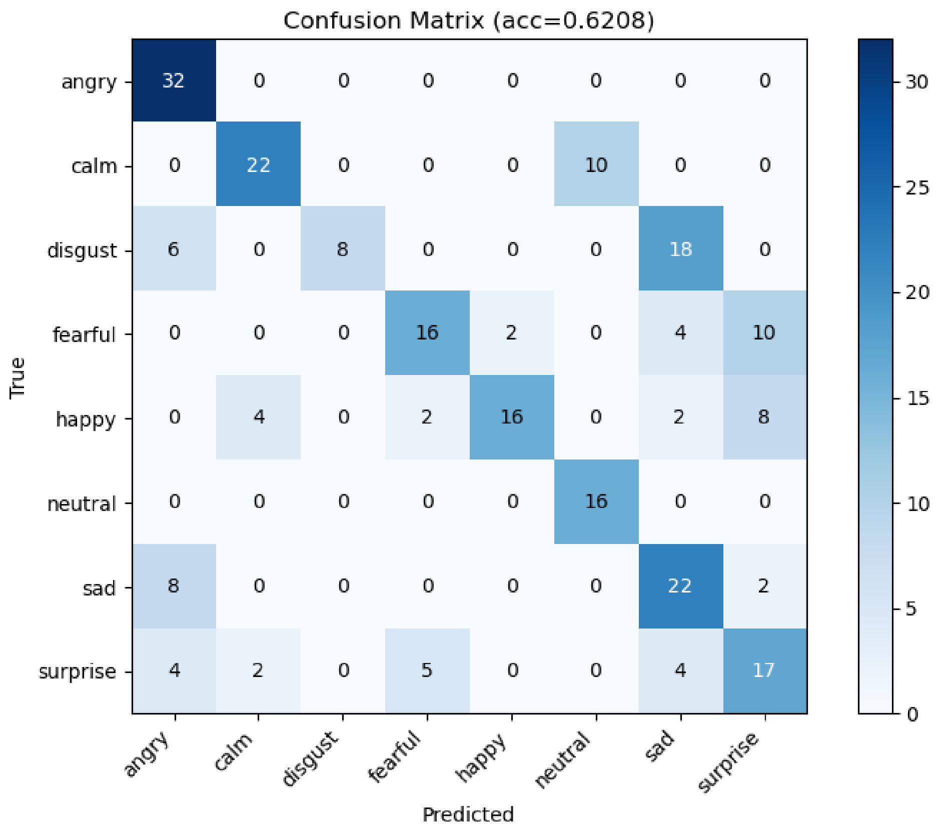
CNN (Resnet) - RNN (LSTM) Model Architecture [Results]

Frames 10

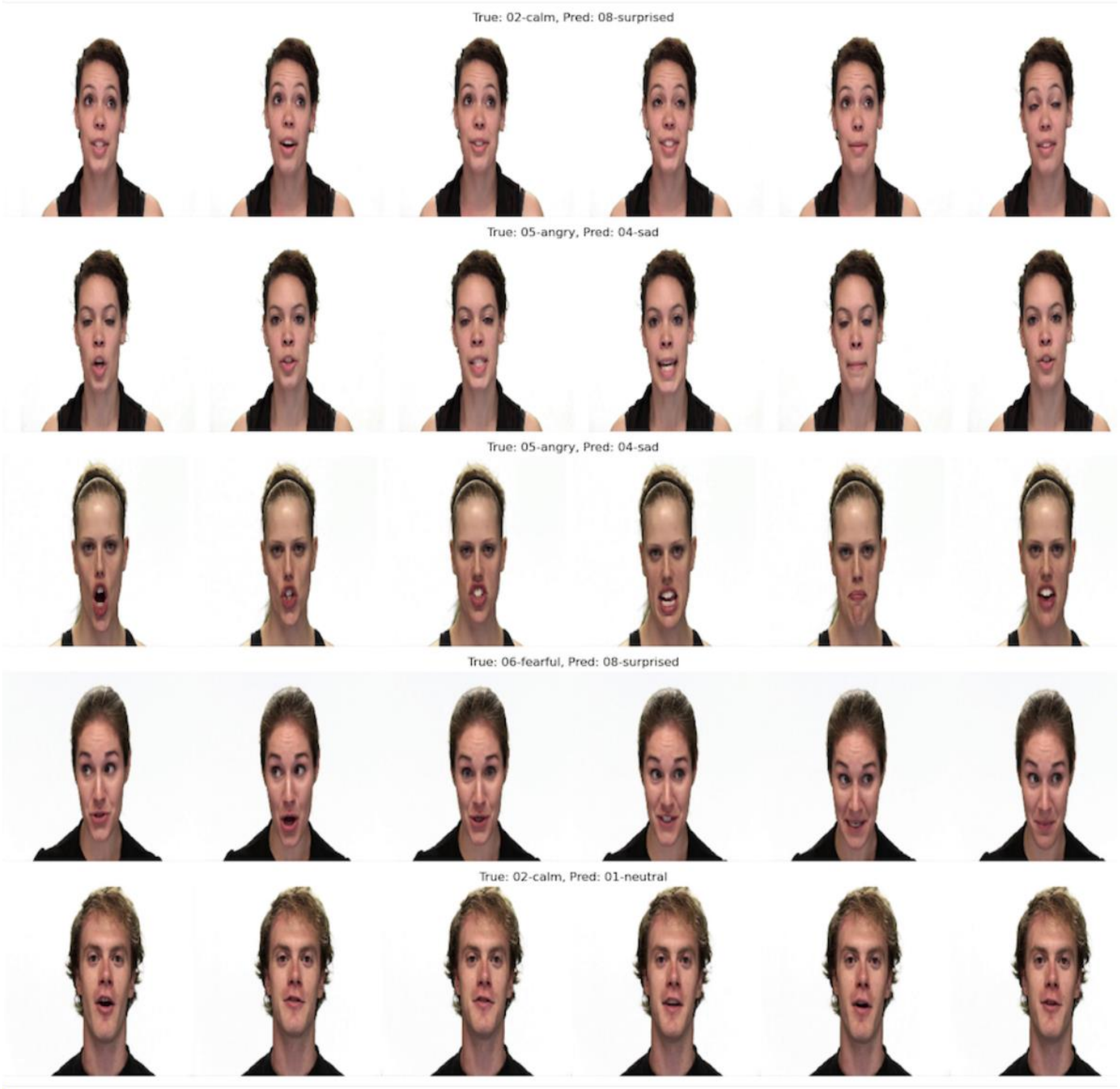
Augment

- Test Accuracy: 62.08%





Misclassified Classes



Results Comparison

Table: Comparison of the best accuracies of the models

Model	Accuracy (Frames 10)
Early Fusion Model Architecture 2 (2d Resnet, pretrained)	42.5%
Late Fusion Model Architecture (2d Resnet, pretrained)	52.9%
3d CNN Model Architecture (3d Resnet)	60.4%
CNN (Resnet) - RNN (LSTM)	62.08%

Conclusion

- CNN-RNN or 3d CNN model is best for emotion recognition in videos
- Downsizing images helps both speed and accuracy
- Sampling more frames gives better representation

Questions?