**AUDIO-VIDEO CLASSIFICATION-** Sanchari Guha Niyogi

**Objectives**: Analyze the video dataset and generate metadata dividing them into subsequent classes and subclasses.

We will implement multi label classification where the labels can be marked as Keywords like ‘Cancer’, ‘Deep Learning’ etc. along with Video Quality, Topic, target etc.

**Algorithm**:

For Video Classification:

Datasets Used: YouTube Videos

For training:

1. Read all the frames in the video dataset and select frames with a regular interval between them.
2. Write them into a directory structure
3. Divide them into training, testing and validation datasets
4. Pass them onto a CNN after reshaping them into desired dimensions.
5. Define the architecture using ReLu and softmax functions. Also add Dropout layers to prevent overfitting.
6. Extract the dense layer and concatenate to the top layer.
7. Save the weights of the model to minimize extra load of retraining.

After Training we will test our model. We are going to use Rolling prediction averaging .

1. Load the model.
2. Set the mean subtraction value.
3. Initialize the object using framecapture and loop over frames.
4. Use the VideoCapture class from OpenCV.
5. Grab a frame and set the dimensions.
6. Perform preprocessing (dimension scaling, color channel adjustment, mean subtraction etc.)
7. Perform Prediction averaging resulting in a class label. In other words, we get a class label with the largest corresponding probability across the average predictions.
8. Indicate the prediction on the output frame.
9. In case of flickering adjust the frame size to smoothen results.

For Audio classification:

1. Use a suitable API for speech to text conversion.
2. We can use a library called RAKE (Rapid automatic Keyword Extraction)
3. Perform preprocessing before training.
4. Use any pretrained model e.g. BERT to train the classification model.
5. Check the classification accuracy.

**Tools**: Keras , Tensorflow , OpenCV, Google Cloud Natural Language API

**Probable Challenges :**

1. Backend errors
2. Computational speed
3. System Compatibility

**Implementation in the existing system:** It is vital to have a good system in place. CNNs are compatible with mobile devices where memory and computational power are limited. Moreover they require much lesser preprocessing as compared to other classification algorithms.