Pakistan Journal of Engineering and Technology
ISSN (Print): 2664-2042; ISSN (Online): 2664-2050
Received: 2024-12-20
Accepted: 2025-05-12
https://doi.org/10.51846/vol8iss1pp32-39

PAKJET
Pakistan Journal of Engineering & Technology

Article

Analyzing Expressions to Curate Feeds: A Real-Time Mood-Based Reels Algorithm

Isha Bhutto and Asad Ali Jatoi *

Department of Software Engineering, Mehran University of Engineering and Technology, Jamshoro 76062, Pakistan
* Correspondence: Asad Ali Jatoi (17sw45@students.muet.edu.pk)

Abstract: As technology evolves, the demand for customised and personalised experiences is increasing in daily life. This tendency is primarily noticeable in the field of social media platforms, where user satisfaction is pivoted on the content that relates well with the individual's preferences and interests. Meeting these demands requires emphasis on understanding and addressing user-specific needs and requirements. Vertical videos and reels are rising in popularity and growing rapidly nowadays at a faster rate due to their new style of representation. A huge random cluster of reels makes it challenging for users to find interesting and relevant reels that align with their current mood hence to keep them engaged on the platform. To overcome this lack, this research presents a solution, namely Reel Mood, an innovative algorithm that employs the latest facial detection technology with Flutter. It has been designed as an initiative that will reshape recommendation algorithms. The research detects the user's emotion through facial expressions using the device's camera and captures images throughout. From those images, their mood is analysed, and based on their current emotion, our Reel Mood application recommends reels tailored as per their mood. The facial expression machine learning model integrates smoothly into the Flutter application, featuring an engaging user interface: recognising facial expressions and recommending reels based on their current mood.

Keywords: AI-based Personalization, Convolutional Neural Network, Facial Expression Recognition, Flutter Tensor Flow Lite, Keras

1. Introduction

1.1. Background and Motivation

In this digital era, users have personal preferences and want to customize everything according to their expectations especially in terms of social media platforms. Everyone serves time for seeking sources of happiness with the help of social media apps. Nowadays, short videos such as Reels have become more addictive and have high watch hours as they convey information in less time. HypeAuditor team analysed 77.6 million Instagram posts throughout July 2022 with more reach and engagement were reels reels-based [1]. Mood plays a great role in efficient content recommendation in the way that relates to the user choice. Either the user is happy, sad, or angry; the user wants to customize content according to the current situation unlike traditional content recommendation systems approach which gives recommendations based on user engagement (like, share, and comment), past preferences, and searching history. There are many systems providing recommendations but not following to get the real-time requirement of the user, which leads to frustration and disengagement. Due to a lack of the user's emotional support and awareness during the reel recommendation process, it could lead to irrelevant reel suggestions, creating a gap between the user and the reels displayed. Eventually, this might cause a reduction in overall application engagement, underlying the need for a more adaptive and emotion-sensitive reel recommendation system.

Therefore, we developed an application, called ReelMood which is designed to fill the gap by customising the content according to the current mood to increase engagement. ReelMood ensures to provide related content in a user-friendly interface on the basis of user facial expressions and emotions. The algorithm used in ReelMood tracks the user's emotional state throughout their activity on the app rather than just relying on past preferences and past engagement areas of consumed content of reels. ReelMood improves the appropriate content experience and guarantees that every reel is aligned to the user's current emotional state. Adapting reel content recommendations with respect to the user's mood gives flexible, dynamic, and personalised content exploration.

1.2. Literature Review

Emotions, whether expressed through words or facial expressions, determine a great deal about a person's point of view. Today, emotion detection—whether verbal or visual—plays a crucial role in revealing a person's sentiments towards a product, idea or form of entertainment. Shahid Salim et. al. [2] highlighted the importance of emotions conveyed through written reviews in guiding a fellow customer as well the manufactures for product improvements and shaping effective marketing strategies.

However, the visual emotions can reshape the future of forms of entertainment unlike the traditional reel recommendation systems which deals with the factors like previous liked history, relevant user's liked reels (collaborative

filtering system), most interacted with reels, web-sessions records and user search-history based reels etc. ReelMood on the other hand adds-on emotions detection with existing traditional factors to achieve one-step ahead accuracy. Traditional factors are key factors but those ignore the current emotional state of the user which outdates the algorithm's capability — as a user is not always in same emotional state. Emotions develop a physiological awakening condition in a person that is personal and private [3]. Detecting facial movements could be challenging as they are very minimal and lead to differences. To achieve this, some technologies, such as deep learning and machine learning, have been used and have gained positive outcomes [4].

Almost every social media application has their own algorithm which recommends video-content to users. Such as reels, a built-in feature enclosed by the Instagram platform, serve as a repository of user-generated media (UGM), mainly composed of short videos varying between 15 and 60 seconds in length [5]. Instagram's algorithm for recommendation suggests reels based not merely from the accounts the user follows but also delivers reels from the suggested accounts. It considers multiple key factors to define which reels are displayed to the user which includes, confirming that the platform remains interactive and meaningful.

Like Instagram, which has reels to showcase personal experiences and creativity, Facebook has a feature called Shorts. Reels and shorts have the potential to record experiences, exchange ideas, and showcase creativity with everyone. Essentially, the content is showcased as shorts influenced by diverse engagement parameters such as likes, shares, comments, and search history, which represent the user's participation [6].

A mood-based music recommendation system [4] uses machine learning to detect the current emotion from facial expressions and recommend music based on the recognized emotion. It also provides the feature to put an emoji to indicate the user's current mood so that songs can be filtered using that emoji. This application works on happy, sad, angry, surprised, and neutral emotions. It gives 75% accuracy and can be further used to recommend videos or movies [4]. Similar research [7], where researchers have used advanced deep learning techniques like Convolutional Neural Network on three different models including VGG-16, ResNet-50 and Inception -V3—on renowned facial images datasets (FER-2013 and CK+). The research has acquired 97.93% accuracy on one dataset and 92.91% on other. These strong researches have laid the foundation to our research incorporating emotions detection while watching reels.

The motive of this research is to develop an algorithm that can be integrated into an application or website without requiring model-generation or other preparatory steps. In Pakistan, the proportion of people having application server skillset is notably low—only 6.7%, as summarized by Bilal Raza et. al. [8]. Considering this dire gap, simplifying automations like ReelMode can help the broader range of developers to access state-of-the-art features simply by connecting to an API.

1.3. Contribution

The problem with the above video recommendations systems is for cold users as they have no preferences in the beginning [9] hence no data for algorithm to start working with. Some social media applications, therefore, initially start asking user's preferences from a set of pre-defined entertainment categories such as Sports, Arts, Music, Lifestyle, Travel etc. Which basically lays a foundation for algorithm to start with.

A research study portrayed the TikTok (short-video platform) recommendation algorithm as 'knowing me better than I know my- self' [10]. Considering the quoted statement and now adding-on emotions detection with that will certainly skyrocket recommendation algorithm's capabilities to another level. Our idea was to embed traditional recommending factors with facial expressions detection to curate users' feeds with more accuracy and relevancy.

2. Materials, Tools and Technologies

To bring the idea of 'Analyzing Expressions to Curate Feeds' to reality, different tools and technologies have been integrated into one application. Flutter seemed suitable opt for application development due to Flutter's Write Ones Run Everywhere (WORE) and embedded widget system nature. Following are the frontend, backend and model development technologies.

2.1. Frontend Material and Technology

Flutter application was developed to showcase the model's proper working for which Dart and Flutter's built-in widgets were used. Flutter has its own wide range of user-interface (UI) components.

2.1.1. Flutter Built-in Widgets

Widget based UI, also known as, Declarative UI method involves using widgets for everything. Widgets are either stateless or stateful in Flutter. An application undergoes both as on a simple click of a button or just a scroll states of components tend to change. To portray the working of model, we developed a social media application which shows reels, but the catch is it will first identify one's mood based on their facial expressions. Whole social media application is built on Flutter widgets and Dart.

2.1.2. Dart Language

Dart is Flutter's programming language which serves a vital role in developing its frontend. Dart is diverse framework of Flutter. Dart assets in deciding what action to be performed based on a conditional change, alongside with declaring and organizing widgets. We can say Dart is somewhat similar to JavaScript's role in HTML with some critical distinctions.

Below attached is the Fig. 1 showing UI developed using widgets and Dart.

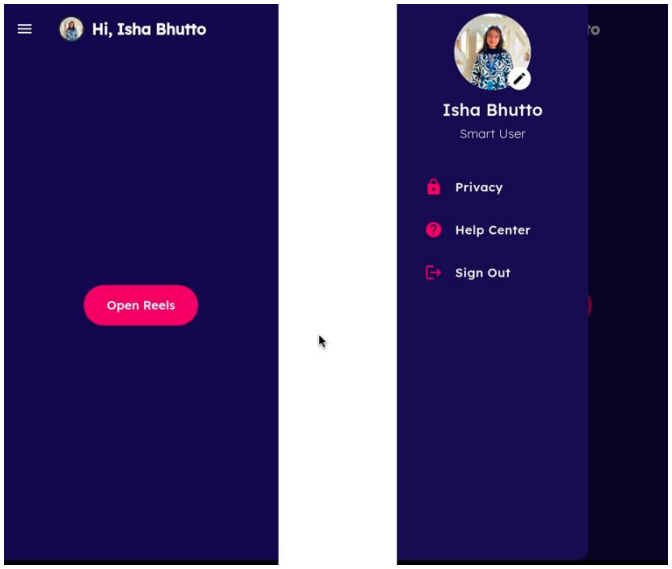

Figure 1. User interface of Flutter application.

2.2. Backend Material and Technology

For backend, Flutter and Firebase were used. Flutter mainly alongside with Dart to implement the business logic part of the application as stated above. Firestore was used at multiple instances defined below.

2.2.1. Firebase Firestore

The backend of this application was built with Firebase database that provides real-time database functions allowing cloud storing options. It gives security, some free storage, smooth retrieval and insertion and handles real time changes efficiently that help in retrieval and management of mood-based reels.

Moreover, Firebase Authentication was used to have signup and sign in functionality in the application. Firebase provides its own authentication system, easy and reliable.

Firebase Firestore was used to serve as a pool of data (see Fig. 2). It is a NoSQL (No Structure Query Language) database system. The database stored wide variety of reels in different categories based on emotions. Our model encompasses Angry, Happy, Sad, Surprised and Neutral emotions detection. Based on each emotion different directories were created which assist to show particular set of reels based on current mood. This pool is expandable and can have more emotions when implemented with detection algorithm.

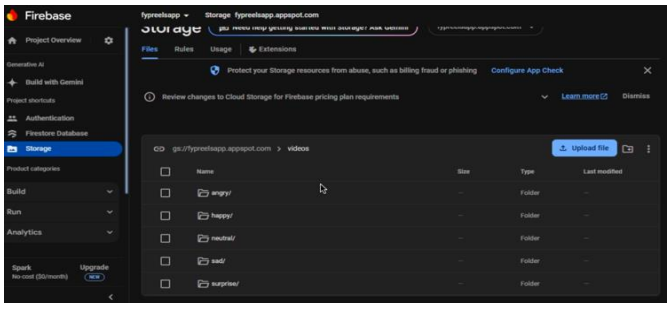

Figure 2. Pool of reels stored over Firestore with respect to different emotions.

2.3. Model Development Material and Technology

Model development phase undergone data collection, data preprocessing, algorithm selection, model training and model testing phases. The key tools used are stated below.

2.3.1. Kaggle and Instagram

Kaggle is world renown dataset platform. Kaggle is widely used to develop and refine machine learning models with the help of datasets. It is community-driven approach for collaboration. Our model's capabilities were enhanced by incorporating carefully selected datasets. To classify emotions from different facial expressions, wide images of people with different emotions were gathered in form of datasets from Kaggle. Model is furthermore tested using some different images dataset again collected from Kaggle.

To show reels based on a specific emotion, we needed a lot of classified reels. Instagram was used to meet this requirement. The videos we are showcasing in our app are collected from Instagram reels. Instagram provides random reels but those were classified into different categories based on our selective set of emotions.

2.3.2. GoogleColab and Python

A model is to be trained using Python as programming language and is done on tool called Google Colab. The development of the emotion detection model is based on Python. It gives a great set of libraries and dependencies to develop machine learning models with ease of understandable writing code and complex calculations. Python, with smallest code snipped, helped in converting standard format of model: H5, which is used to run model over web servers, into TFLite format which is mobile application compatible Tensor Flow extension (see Fig. 3).

```
from keras.models import load_model
model = load_model("/content/model.h5")

TF_LITE_MODEL_FILE_NAME = "tflite_model.tflite"
tf_lite_converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = tf_lite_converter.convert()
tflite_model_name = TF_LITE_MODEL_FILE_NAME
open(tflite_model_name, "wb").write(tflite_model)
convert_bytes(get_file_size(TF_LITE_MODEL_FILE_NAME), "KB")

# Convert the model.
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
# or using another method

# Save the model.
with open('tflite_model_another.tflite', 'wb') as f:
```

Figure 3. Python code snippet to convert H5 extension to TFLite.

2.3.3. Keras and TFLite

The model is developed using Keras. The Keras library is built on top of TensorFlow that provides tools for deep learning model development such as preprocessing image module, providing functions to load, preprocess, and augment image data for training [11]. The layering module includes core layers like Conv2D, Batch Normalization, and Activation for

constructing Convolutional Neural Network (CNN) by using the Sequential model as it works on a linear stack just like one input will produce one output.

The ImageDataGenerator class in Keras loads and preprocesses images from the dataset. The ImageDataGenerator class in Keras loads and preprocesses images from the dataset. Due to the ease of use and effectiveness of grayscale, the target size is set to 48x48 pixels, and colour mode is set to grayscale, which is a usual procedure in emotion recognition.

3. Methodology and Implementation

This section comprehensively explains the methodology adopted and its implementation including model training with convolutional neural network, model connectivity with mobile application using Keras and TFLite, and mobile application development in Flutter. The following sections outlines the systematic approach and technical steps taken to develop the proposed application.

3.1. Dataset Preparation and Preprocessing

Multiple image datasets were obtained from open-source Kaggle repositories which helped later-on in model training and testing. The dataset collected was raw in nature hence adaptive measures were taken to categories dataset for model testing later-on.

3.2. Reels Categorization over Firebase

Reels pool was created over Firebase database incorporating five normal moods a user can have, including Sad, Happy, Angry, Surprised and Neutral. Reels obtained from Instagram were vague. Manual categorization was adopted to sort and then store reels in specified mood-based repositories. Model generates output of the detected emotion. The output is then used to show particular set of reels. If detected emotion is Happy, Happy repository will be selected as root directory and user will be seeing reels from that category only. Reels which are shown once will be marked as seen reels and those will not be appearing again as to prevent user from falling into the state of boredom.

This fetch videos function retrieves video URLs for a particular emotion category folder. It collects all video records and sequentially collects their downloadable URLs, storing them into the variable videoURLs. Once all the URLs are fetched, it refreshes the state of videos with the fetched URLs and turns off the loading indicator by toggling loading off. The code snippet to fetch videos from Firebase is shown in Fig. 4.

This cloud storage solution provides robust, scalable storage to integrate with Firebase Firestore that allows efficient retrieval of reel content without disturbing the app performance. This structural organization helps in providing smooth user experience by loading relevant reels quickly after mood detection.

```
Future<void> _fetchVideos(String detectedEmotion) async {
    try {
      final ListResult result = await FirebaseStorage.instance
          .ref('videos/$detectedEmotion')
          .listAll();
      final List<String> videoUrls = [];

      for (final Reference ref in result.items) {
        final String url = await ref.getDownloadURL();
        videoUrls.add(url);
      }
      // Update state after fetching all URLs
      setState(() {
        videos = videoUrls;
        isLoading = false;
      });
    } catch (e) {
      print("Error fetching video URLs: $e");
      setState(() {
        isLoading = false;
      });
    }
  }
```

Figure 4. Code snippet to fetch videos from Firebase.

3.3. Model Design and Training

Keras is used for building and preprocessing the model. Preprocesses images are loaded from the dataset. The images colour mode is set to grayscale, standard colour mode for emotion recognition. Converting colour mode reduces the computational complexities as grayscale mode has one channel as compared to colour image having three channels (Red, Green and Blue also known as RGB) [12]. So, if an image is of size 48×48 its colour model pixel values will be 6,912 while grayscale image of same size will have 2,304-pixel values. Based on below formula.

$$Total\ Pixels = Height \times Width \times Channels \quad (1)$$

This reduction in complexity makes Convolutional Neural Network (CNN) algorithm extraction feature faster and less memory intensive. The model initiates with an input layer of 48×48×1, corresponding to the target image size and grayscale channel. CNN is ideal algorithm for facial recognition due to its ability to learn spatial hierarchy features of an image automatically. CNN works on convolutional layers which extract geometrical features for instance edges, corners or patterns from provided image. Facial expressions are read from various facial parts including mouth, eyebrows, eyes etc. CNN, instead of computing every pixel separately uses shared weights. CNN's MaxPooling layer comes handy dealing with different variational facial features. Which guarantees that the model is identifying similar feature i.e. smile for happy mood, furrowed brow for surprised emotion even if the image is tilted.

CNN works on different layers: Lower Layer to learn edges or corners, Intermediate Layer to learn facial parts like lips, eyes or eyebrows and Deeper Layer merges other layers' features to learn pattern such as happy face, sad face or surprised face.

Input layer holds the raw image as input. Convolutional layers apply kernels to filter the image as to extract features. Complex features are identified using Rectified Linear Unit (ReLU). Softmax layer, also known as Output layer, calculates the probability of each emotion and assures those all sum up to 1. For instance, a raw image is passed throughout the layers, in final vector of raw scores from different layers are collected for each classification class. These outputs are called logits. Set of logits is collection of vector scores, in our model's case it would be vector score of happy, sad, surprised, angry and neutral.

$$Logit = [happy, sad, angry, surprise, neutral] \quad (2)$$

If the scores are real numbers consisting of negative integers and larger values then exponential of scores will be taken to reduce the smaller ones (negatives) and to amplify larger ones (positive).

$$Logits = [e^{happy}, e^{sad}, e^{angry}, e^{surprise}, e^{neutral}] \quad (3)$$

The final output required has to show highest probability of one class. Hence each exponential score will be divided to sum of all exponential scores, this step is called Softmax normalizing. This way the final set will be sum to 1. Considering the output to be below set.

$$Normalised\ Set = [0.53, 0.01, 0.05, 0.23, 0.18] \quad (4)$$

The values sum to 1 and indicate the probability of each class. Happy: 53%, Sad: 1%, Angry: 5%, Surprise: 23% and Neutral: 18%.

The higher probability from normalised set is of Happy class, therefore; CNN will classify this picture into Happy class.

The CNN 5 output classes are performed in training this model. The model consists of four convolutional layers for feature extraction and every layer is followed by batch normalization, ReLU activation, dropout (to reduce overfitting when new data is entered) and max pooling. Dropout percentage is set to 0.25 which is 25% indicating that out of 100, 25 neurons will be deactivated randomly on every training step, rest of 75 will be utilised to generate the output. The classification of final output is proceeded with the help of softmax. The model is compiled with the Adaptive Moment Estimation (Adam) optimizer which handles the complexities of training CNN based model. It adapts the overtime changing in model's learning and as the dataset is composed of pictures, it is relatively large dataset. With Adam, efficient learning is assured. First to fully connected CNN model layer code snippet. The code for first to fully connected CNN model layer is displayed in Fig. 5.

```
from keras.optimizers import Adam, SGD, RMSprop
no_of_classes = 5
model = Sequential()

#1st CNN layer
model.add(Conv2D(64,(3,3),padding='same',input_shape=(48,48,1))) #Grayscale Channel
model.add(BatchNormalization())
model.add(Activation( 'relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout (0.25))

#2nd CNN layer
model.add(Conv2D(128,(5,5),padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

#3rd CNN layer
model.add(Conv2D(512,(3,3),padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

#4th CNN layer
model.add(Conv2D(512,(3,3),padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
model.add(Flatten())

#Fully connected 1st layer
model.add(Dense(256))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

# Fully connected layer 2nd layer
model.add(Dense(512))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

model.add(Dense(no_of_classes, activation='softmax'))
opt = Adam(lr = 0.0001)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

Figure 5. First to fully connected CNN model layer code snippet.

3.4. Model Conversion to TFLite

CNN algorithm applied Keras library model comes with a HDF5 format (h5 extension). H5 format stores model architecture, structured unstructured data and metadata. Keras and TensorFlow library trained models are saved with this extension to avoid retraining every time to make predictions. H5 holds the preset of our model but challenge is to integrate h5 extension with flutter application.

Flutter is a mobile application development framework by Google. To load created model in Flutter application, *TFlite_Flutter* plugin was used. H5 is converted to TensorFlow Lite format which is compatible with Flutter. The conversion was made using below snippet shown in Fig. 6 in Python.

```
from keras.models import load_model
import tensorflow as tf

# Loading Keras model
model = tf.keras.models.load_model('/content/model.h5')


# Converting the model to TensorFlow Lite format
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Saving the converted model
with open('model.tflite', 'wb') as f:
```

Figure 6. Code snippet to convert format.

The .convert() method transforms .h5 into the TFLite format. The converted model is saved and the details about the tensor shape and type is shown at the end.

3.5. Flutter Application Development and Model Loading

Basic Flutter application was developed to test the algorithm's working in reference to our idea. Currently the application has a button on the top which opens user's phone camera to click picture and detect the emotion. Picture clicked is stored on local storage of the phone in order to keep the clicked picture private. After the emotion is analyzed, it shows reels based on emotion. The application has basic features of authentication and reels showing only as it is a preceding product. To make it all work first UI of the application was build using Flutter widgets and dart as explained in frontend section then the model converted from h5 extension to TFLite is loaded in the flutter application (see Fig. 7). Face detection screen of application is shown in Fig. 8.

```dart
Future _tfLteInit() async {
  Tflite.close();
  try {
    String res = (await Tflite.loadModel(
      model: "assets/emotion_detection.tflite",
      labels: "assets/labels.txt",
    ))!;
    print(res);
  } catch (PlatformException) {
    print('Failed to load model.');
  }
}
```

Figure 7. Code snippet to load model in Flutter.

In flutter application, the top has camera icon that opens screen where currently we can open camera or upload a picture which is then classifying the mood. To make application testing easier 'select from gallery' option was availed.

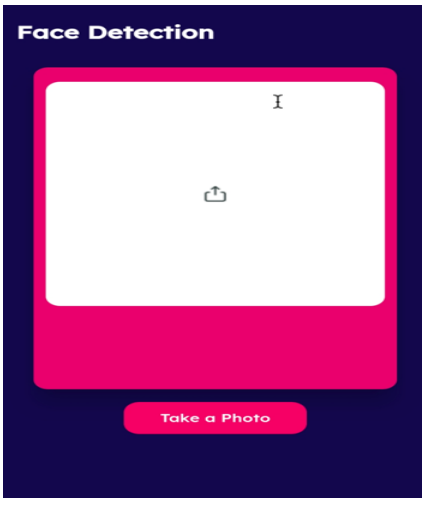

Figure 8. Face detection screen of the application.

When a picture is taken it detects the mood and showcases reels from that category of mood (see Fig. 9). After the picture is taken, simultaneously facial expressions are detected. Code snippet to detect emotions from picture is shown in Fig. 10.

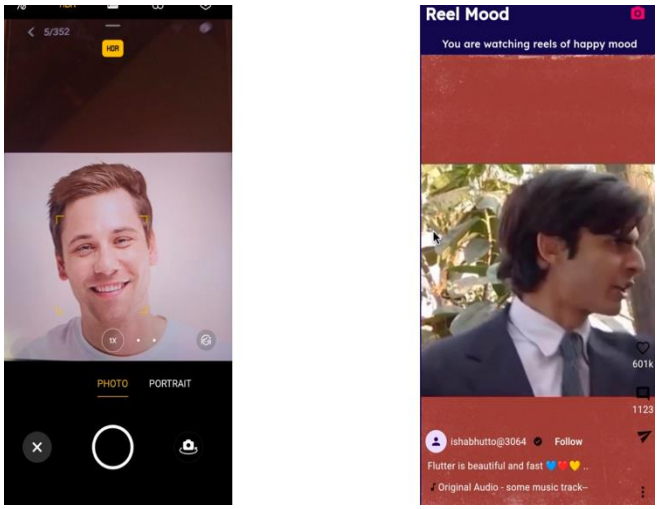

Figure 9. Detection and recommendation.

```dart
pickImageCamera() async {
    final ImagePicker picker = ImagePicker();
    // Pick an image.
    final XFile? image = await picker.pickImage(source: ImageSource.camera);
    if (image == null) return;
    var imageMap = File(image.path);
    setState(() {
      filePath = imageMap;
    });
    var recognitions = await Tflite.runModelOnImage(
        path: image.path, // required
        imageMean: 0.0, // defaults to 117.0
        imageStd: 255.0, // defaults to 1.0
        numResults: 2, // defaults to 5
        threshold: 0.2, // defaults to 0.1
        asynch: true // defaults to true
        );
    if (recognitions == null) {
      print("recognitions is Null");
      return;
    }
    print(recognitions.toString());
    setState(() {
      label = recognitions[0]['label'].toString();
    });
    // Return the detected Label when an image is processed
    Navigator.pop(context, label);
  }
```

Figure 10. Code snippet to detect emotions from picture then print.

3.6. Integration of Emotion Detection with Feed

After emotion is detected, the reels are shown based on classified emotion. The recommendation of reels is coming from the repository of videos stored over Firestore. Following Flutter code shown in Fig. 11 obtains reels.

The algorithm is capable of detecting 5 different moods and showing reels.

```
    final detectedEmotion = await Navigator.push(
      context,
      MaterialPageRoute(builder: (context) => const CameraPage()),
    );

    print("Detected Emotion: $detectedEmotion");

    if (detectedEmotion != null &&
        availableCategories.contains(detectedEmotion)) {
      setState(() {
        isLoading = true;
        videos.clear();
        this.detectedEmotion = detectedEmotion; // Store detected emotion
      });
      _fetchVideos(detectedEmotion);
    } else {
      setState(() {
        videos = [];
        isLoading = false;
      });
      print("No videos available for the detected emotion.");
    }
  }
```

Figure 11. Code snippet to show reels based on detected emotion.

4. Results

Effective training and fine-tuning are achieved by setting the Adam optimizer with a learning rate of 0.0001. The model was trained over 48 epochs, using *fit_generator* to manage large image batches. To enhance generalization, the training set was augmented, and the test set was used to monitor validation performance. Moreover, the performance of the model is evaluated on the basis of Loss Curve and Accuracy Curve. Loss curve to display training and validation loss and Accuracy curve to show improvements in training and validation accuracy over epochs, highlighting convergence behaviour and final accuracy. Training and validation accuracy curves are shown in Fig. 12.

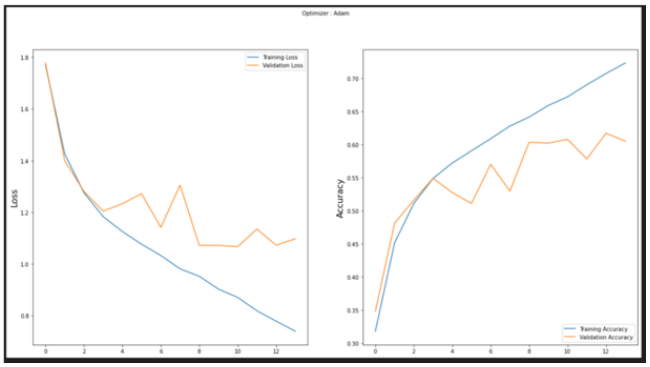

Figure 12. Training and validation accuracy curve.

Furthermore, a dataset of 1200 images mixed with different emotions was taken to test the accuracy and precision of model against each category. Generated data was plotted on a confusion matrix where x-axis represent the actual expression in the image and y-axis represents the predicted expression (see Fig. 13). For instance, in the dataset of images labeled with 'Happy' expression, the model correctly identified the expression 207 times. However, it also misclassified it as 'Sad', 'Angry', 'Surprised' and 'Neutral' in 07, 03, 22 and 13 instances.

	Happy	Sad	Angry	Surprised	Neutral
Happy	207	07	03	22	13
Sad	04	196	30	09	05
Angry	03	22	198	15	11
Surprised	05	05	10	184	06
Neutral	09	08	23	17	188

Figure 13. Confusion Matrix where x-axis represents actual expression and y-axis represents predicted expression.

To calculate Precision, Accuracy, Recall and Overall Accuracy. Following formulas are used.

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

$$Overall\ Accuracy = \frac{Total\ Correct\ Predictions}{Total\ Predictions} \tag{8}$$

where TP is True Positive, number of times when emotion was predicted correctly. TN is True Negative, number of times when emotion was correctly predicted in negative class. FP or False Positive, when incorrectly predicted as a positive but actual was negative. FN or False Negative is when incorrectly predicted as negative which was actually positive. The data obtained was passed to the Confusion matrix online calculator by reference [13] as shown in Fig. 14.

Draw confusion matrix for 5 classes.

		Truth data						
		Class 1 Happy	Class 2 Sad	Class 3 Angry	Class 4 Surprised	Class 5 Neutral	Classification overall	User's accuracy (Precision)
Happy	Class 1	207	07	03	022	013	252	82.143%
Sad	Class 2	04	0196	30	09	05	244	80.328%
Angry	Class 3	03	022	198	015	011	249	79.518%
Surprised	Class 4	05	05	10	184	6	210	87.619%
Neutral	Class 5	09	08	023	017	188	245	76.735%
	Truth overall	228	238	264	247	223	1200	
	Producer's accuracy (Recall)	90.789%	82.353%	75%	74.494%	84.305%		

Overall accuracy (OA): 81.083%

Kappa[1]: 0.764

Figure 14. Calculation of accuracy, precision, recall, and overall accuracy.

The overall accuracy generated for our model is 81%. Which is initially better but in future more datasets can be pushed to train model to exceed current accuracy.

5. Discussion

The proposed Flutter application integrated with CNN based emotion detection model presented effective mood-based personalized reels. Our findings indicate content over social medias can be more personalized with user emotions. Result generated by our model is coherent with prior research on emotion detection and personalized reel system. Unlike traditional approaches, which focus on sessions-data, search history and collaborative learning, our model shows real-time reels recommendation with emotions, portraying the potential of practicality integration to user-centric applications.

However, one big challenge remains, the continuous detection of emotions throughout the watch time. Currently the application is asking user to allow it to take a picture and thus detects the emotion. Application does not keep the data; it uses image to detect emotion and then the picture vanishes. So, the user will be seeing one-emotion-based reels unless a new picture for mood detection is taken. This must be automated in future work.

We are aiming to implement continuous detection and trinity of reels strategy in future. In trinity of reels, we will be keeping three videos in-lined according to user's detected mood. For instance, if the detected mood is Sad, we will keep three reels in our list. First shown, then, when we are showing the second reel, application will start detecting the face again without user having to press capture button again. So, this time the mood may have changed, we will store the new detected emotion and again make a trinity of reels ordered in our feed. Thus, the third reel is shown, and next reel would be the first reel of trinities from the newly detected emotion. This process will keep repeating hence keeping user's attention intact throughout the application.

6. Conclusion

Our study illustrates that personalization of content can be furthermore enhanced on social media with the integration of emotions detection. It can be challenging to uphold user privacy policies with continuous detection of expressions but when open-sourced, user can see that the model is deleting the captured content and just holding the textual output from it. From our research, it is observed that emotion-driven content delivery improves user experience. Future work in this area could help us expand our vision by applying this framework from entertainment to therapeutic innovations. This research lays the foundation for an intelligent digital ecosystem.

Funding: This research received no external funding.

Data Availability Statement: The datasets used during the current study are available from the corresponding author on reasonable request.

Ethical Statement: This research adheres to the ethical guidelines established by the Committee on Publication Ethics (COPE). All procedures and methodologies used in this study comply with COPE standards, ensuring transparency, integrity, and respect for all participants involved.

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] "Social Media Today," [Online]. Available: https://hypeauditor.com/blog/how-different-types-of-content-perform-on-instagram/. [Accessed 24 11 2024].

[2] S. Salim, Z. Iqbal and J. Iqbal, "Emotion Classification through Product Consumer Reviews," *Pakistan Journal of Engineering and Technology, PakJET,* vol. 4, no. 4, pp. 35-40, 2021.

[3] M. Borgaonkar, V. Babanne, M. Katta and P. Kudale, "Emotion based personalized recommendation system," *International Research Journal of Engineering and Technology (IRJET),* vol. 7, no. 8, pp. 701-705, 08 2020.

[4] A. Mahadik, S. Milgir, J. Patel, V. Kavathekar and V. B. Jagan, "Mood based music recommendation system," *International Journal of Engineering Research and Technology,* vol. 10, no. 06, pp. 553-559, June 2021.

[5] D. Menon, "Factors influencing Instagram Reels usage behaviours: An examination of motives, contextual age and narcissism," *Telematics and Informatics Reports,* vol. 5, 2022.

[6] V. Saminathan, "A Critical Review on Reels and Shorts of Facebook and YouTube," *International Journal of Innovative Research and Creative Technology,* vol. 8, no. 4, pp. 33-39, 2022.

[7] N. Bukhari, S. Hussain, M. Ayoub, Y. Yu and A. Khan, "A Deep Learning-based Framework for Emotion Recognition using Facial Expression," *Pakistan Journal of Engineering and Technology, PakJET,* vol. 5, no. 3, pp. 51-57, 2022.

[8] B. Raza, N. Fatima, S. Nazir and B. Amin, "Skills Set Required for Web Developers in Pakistan," *Pakistan Journal of Engineering and Technology, PakJET,* vol. 6, pp. 86-91, 2023.

[9] M. Saraswat, S. Chakraverty and A. Kala, "Analyzing emotion based movie recommender system using fuzzy emotion features," *International Journal of Information Technology,* vol. 12, pp. 467-472, 2020.

[10] A. Y. Lee, H. Mieczkowski, N. B. Ellison and J. T. Hancock, "The Algorithmic Crystal: Conceptualizing the Self through Algorithmic Personalization on TikTok," *Proceedings of the ACM on Human-Computer Interaction,* pp. 1-22, 2022.

[11] "Keras Tutorial," Javatpoint, [Online]. Available: https://www.javatpoint.com/keras. [Accessed 24 11 2024].

[12] R. C. Gonzalez and R. E. Woods, in *Digital Image Processing*, Pearson Prentice Hall.

[13] M. Vanetti, "Confusion matrix online calculator," 2007. [Online]. Available: https://marcovanetti.com/pages/cfmatrix/?noc=5. [Accessed 18 12 2024].

This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.