M2: Cue-Cetera Design Prototype Report v1.1

Evidence of Soundness

Theoretical Background

The main problem that Cue-Cetera has aimed to tackle is the issue of neurodivergent individuals and children missing important visual cues that would add context to a conversation. Marco et. al. made a study regarding the decreased ability in sensory processing in autistic individuals [1]. Furthermore, Sasson et. al. conducted a study that found that neurotypical people tend to be less willing to communicate with those with autism due to this issue [2]. One way to help these individuals better identify these visual cues is by using something highly accessible to a wide variety of people and also simple to interact with an app. Evan Brown goes into depth on how the use of visual support can help neurodivergent children better identify things that their peers are more adept at picking up such as societally proper behaviors [3]. Cue-Cetera is derived from this example of the use of visual support as it is an app that can identify these different social cues and classify them in an easy-to-understand way.

Apart from continuing to optimize our machine learning model, we plan on using facial detection in each frame captured from the user's video to improve the data being applied to the model. According to the journal *Automatic Facial Expression Recognition based on MobileNetV2 in Real-Time* [12], having an excessive amount of background and non-face areas in the images being applied to a model can decrease the accuracy of the prediction. To avoid this issue, we will use facial detection to create a bounding box around the face and apply that data to the model to get the predicted emotion label.

Prior Art

A related example of classifying social cues is a research project by the Massachusetts Institute of Technology (MIT) in which they had a model to detect social cues based on children's voices using their tone, speech patterns, pitch, and energy [4]. Another example of this would be a study that looked into modeling visual non-verbal social cues using social signal processing (SSP), in which they strived to recognize and understand human social interaction by analyzing their sensed social cues [5]. The initial idea of a visual cue classifier was drawn from the first MIT example to expand more on the audio cues and dive deeper into visual cues. Furthermore, some inspiration was derived from the study of SSP as we made it possible for observed visual cues to be classified in the model and further organized as being positive or negative.

Empirical Evidence

To create a machine learning model that is able to classify the image into one of 28 categories of emotions, we experimented with different convolutional neural networks (CNN) like MobileNetV2 and also created one using Sequential API. Since our dataset was made from two different datasets that had a variety of images of people captured in different ways, we had to experiment a lot with the hyperparameters of each model. Our goal was to find which CNN was most effective in the test set in terms of accuracy.

When working with the MobileNetV2, one of the main challenges was finding which optimizer is the most effective in terms of processing speeds and accuracy. In order to test out several parameters at once, we ran different models, some with frozen layers and others without. We also made sure to try different optimizers for each that were appropriate to our respective goal, like Adam, Nadam, AdamW, and RMSprop. Each model had about 25 epochs and an early stopping of 5 in order to minimize unnecessary training that wouldn't improve the accuracy of the model. According to the article *Optimization Algorithms in Neural Networks* [14], the Adam optimizers and RMSprop algorithm essentially are the same except RMSprop also uses some logic from the stochastic gradient descent algorithm. They also state that Adam is considered the most effective algorithm among the others. Additionally, we made sure to use regularizers to avoid overfitting. The reason we decided to experiment with MobileNetV2 and CNN was because of Lida Hu's journal on Automatic Facial Expression Recognition based on MobileNetV2 in Real-Time [12], which stated that when working with a classification task involving facial expression recognition, it is more beneficial to use CNN and MobileNetV2 to increase the accuracy and speed of the predictions. CNN's are also able to accurately recognize subtle facial expressions, making it optimal for our use case [13]. For the creation of the CNN, we used the sequential API to create the layers, following the general structure displayed in Figure 1 below:

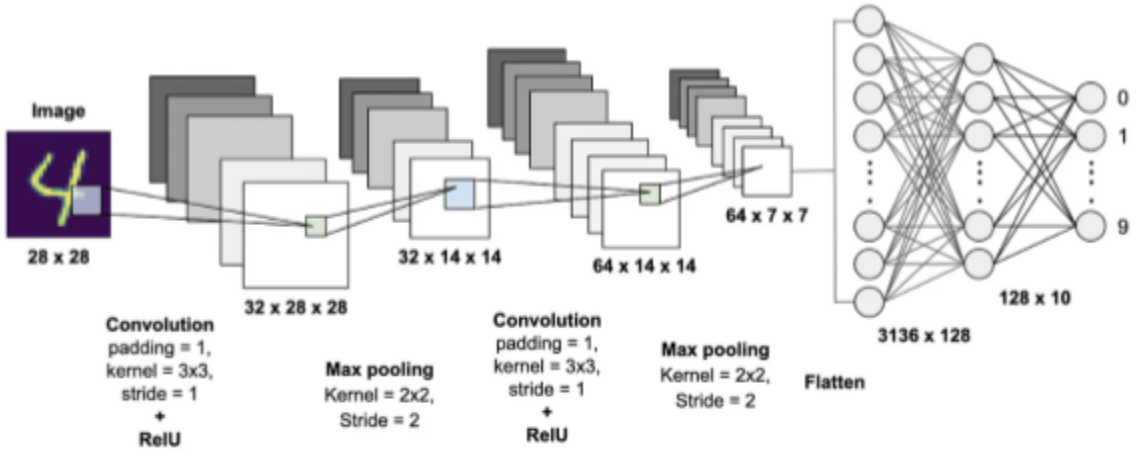


Fig.1. Convolutional Neural Network for image processing [15]

For our prototype model, we repeated this process 4 times and, with a continued dropout layer through each iteration to prevent overfitting. Lida Hu's journal also stated that non-face areas in the images can affect the accuracy of the prediction, so we made sure to crop the images in the EMOTIC dataset to their bounding box of where it was being analyzed to minimize background noise in each.

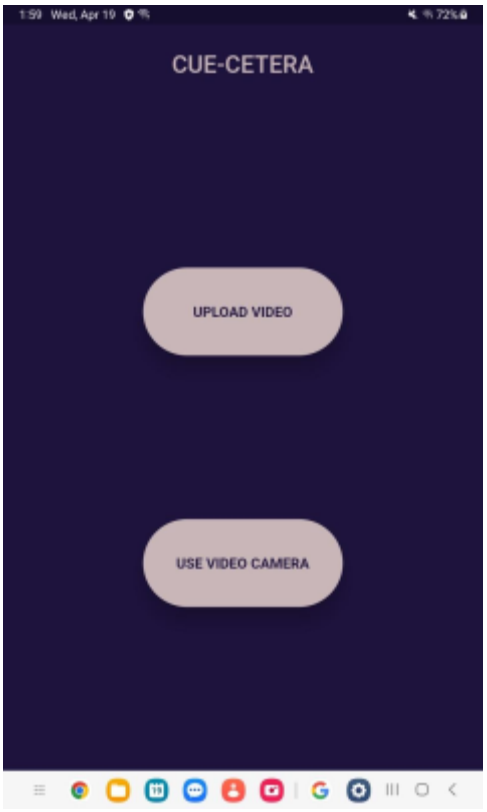
After running a variation of models, each with their own set of hyperparameters, we found that out of all the optimizers, the Adam algorithm performed the best in terms of accuracy. However, it made the model run slower compared to the other optimizers. Also, we found that our CNN model performed better during testing than the models made using MobileNetV2, but it also took around 23 minutes to run each epoch, so extensive training wasn't possible without letting it run for days. Overall, we plan on continuing to use Adam for the optimizer, but altering the hyperparameters, such as the learning rate and regularization to improve the performance. Since we recently got access to HiPerGator, we will use it to further speed up the training process in the future.

External Interfaces

Presentation

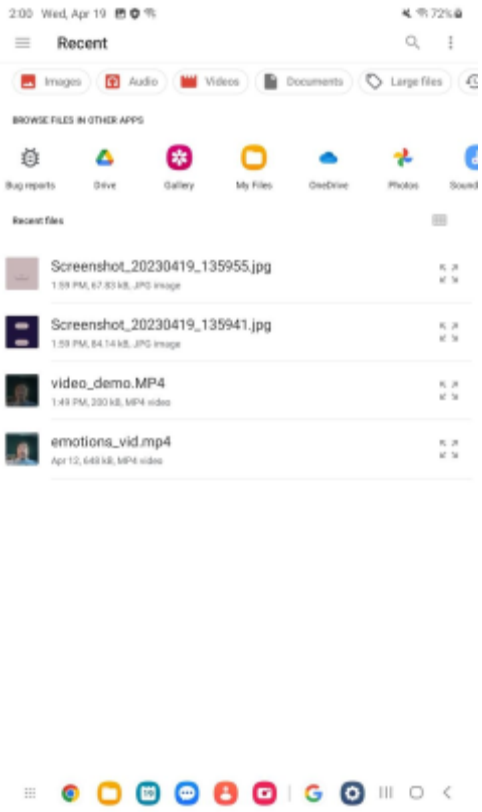
The two main options the user has in the application is “Upload a Video” and “Record a Video”. These are highlighted through the use of buttons that are easy to identify due to their color contrast from the background. In the future, we plan on continuing to fix the aesthetics of the user interface. We also plan on making it more accessible by including a text-to-speech option in the settings for those who are visually impaired. Below are the two use-cases in our current state of the project.

Scenario 1:

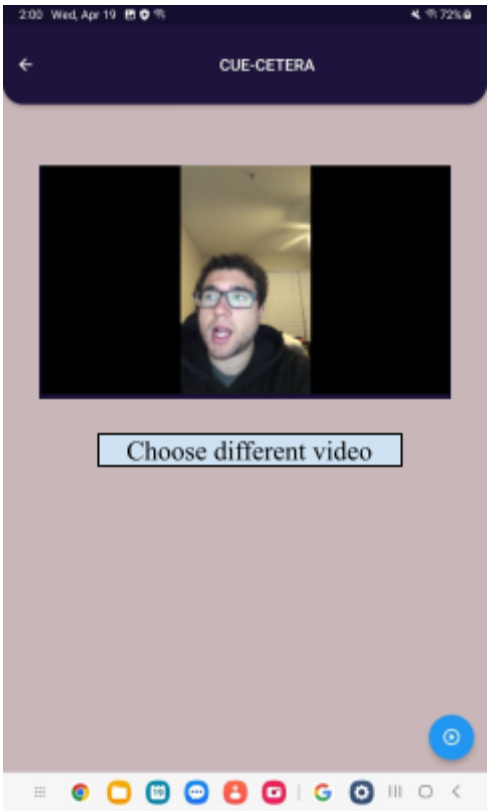


Main Menu page

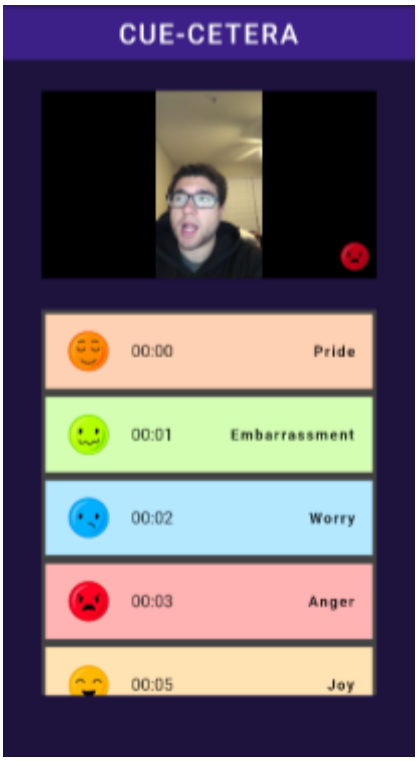
Upload video button goes to select page



Select button click opens up local gallery

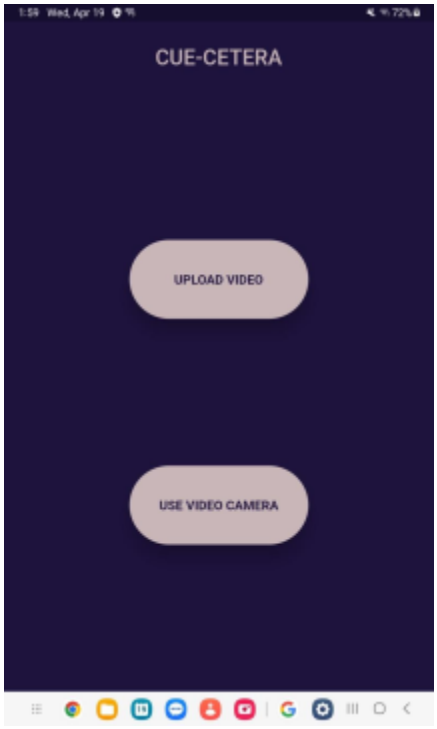


(Not implemented yet) Once video is selected, user can view the unclassified video. User can then either select a different video, or choose to classify the current one.

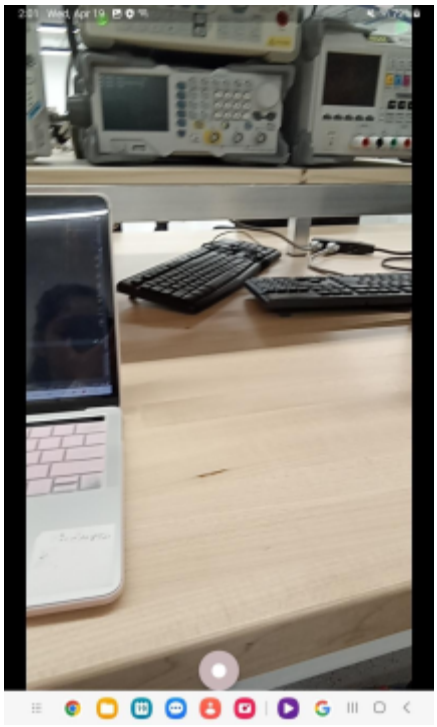


Classified video is displayed with navigational timestamps provided and informational image on the bottom right of each frame.

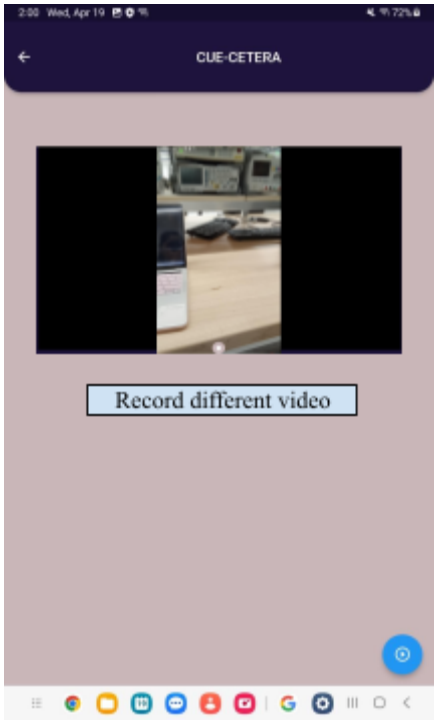
Scenario 2:



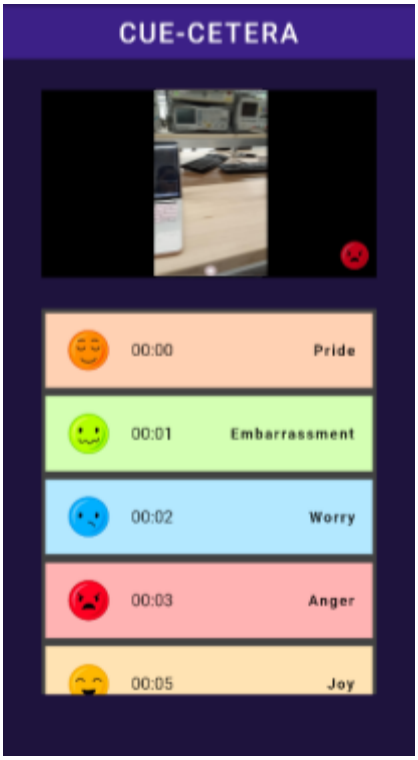
Main Menu page



Use video camera button click opens camera to record video



(Not implemented yet) Once video is recorded, user can view the unclassified video. User can then either record a different video, or choose to classify the current one.



Classified video is displayed with navigational timestamps provided and informational image on the bottom right of each frame.

Perception

For the in-progress part, we are working on temporarily storing the user's recorded or selected video in RAM for access in the Confirmation and Results Display pages, and properly retrieving frame classification data from our back-end. This data should then be organized in such a way that our Results Display page is able to work properly. As of right now, the Results Display page only shows a pre-loaded video that uses a hard-coded, arbitrary timestamp list for navigation and classification.

It is also our intention to minimize our classifications to being either positive, negative, or neutral/default. This will be done by grouping the 26 output emotions from our machine learning program into those three groups, and shown using a blue thumbs up (rather than green for those who are color blind), a red thumbs down, or no image.

- Interaction with interface: Tactile, sound when button is clicked
- Each page on the app is labeled. Errors are printed on the screen. Can navigate to the previous page with the back button.
- On the Results Display page, the user can navigate to a given timestamp using the auto-generated timestamp buttons on the bottom of the screen, which indicate their corresponding time, and emotion.

Usability

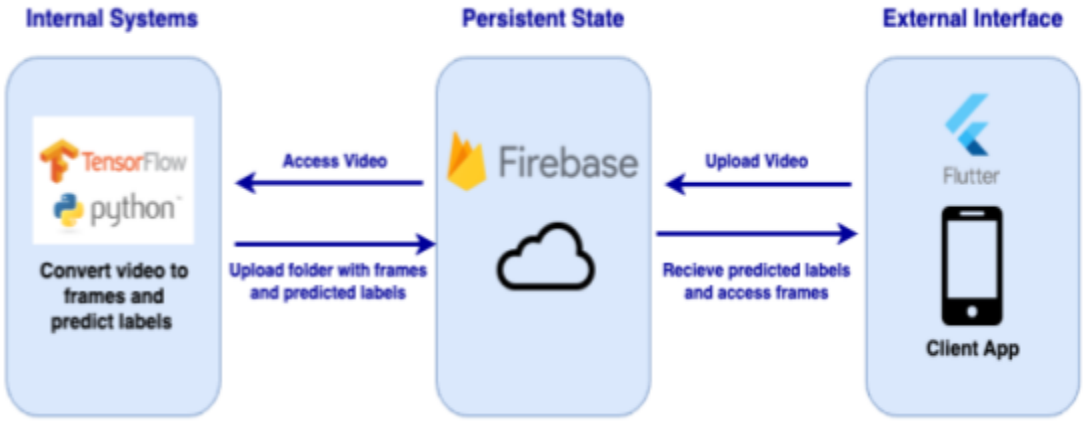
Since the buttons are labeled with their corresponding actions, the application reacts predictively to the user interaction. Every video uploaded or taken by the user is converted to frames at a rate of 10 fps, preprocessed and then applied to the machine learning model. So if a user were to upload the same video two times in a row, it would yield the same labels and frames, which are sent to the database for the front end to access.

Internal Systems

Component Architecture

- We are using Android Studio with Flutter plugin to build the front-end of the app. The main language being used is Dart. The libraries/packages being used are: flutter, file_picker, video player, firebase_core, firebase_database, firebase_storage, firebase_options, camera, flutter_spinkit, and chewie . We are also using cupertino_icons for the icons such as the back button and play.
- The UI design of each page is set up using widgets that are customized with buttons and features using the classes in the flutter library. Each page is also written using a class function. The homepage has two buttons, one for video upload and the other to take a video. When either button is pressed, it navigates the route to the corresponding page (calls the corresponding class). For the video upload page, there is one button, which when clicked uses functions from the files_picker package to access the files on the device and prompt the user to select the desired one. The path of the file is then used to play the video using the video_player package in tandem with the chewie package.
- For the record video option on the home page, when the button is clicked, the camera package is used to select the first available camera and preview the camera on the app. There is also a button to record and stop the video. When the video is stopped, the path of the video is sent to the video player.
- On the Results Display page, the video is available for viewing using the video_player and chewie packages. As the video plays, a search is performed using the video's current time to find the timestamp window the user is currently viewing. The appropriate symbol for the current frame is then displayed based on the results of that search.

Communication Mechanisms



We use Firebase to store the video the user uploads. This allows the app server to access the video in order to convert it into frames and apply the machine learning model to the set of images. In doing so, it creates a folder of the frames extracted from the video, with their corresponding timeframes, along with an array of labels that map to each frame. The folder and array are then uploaded to Firebase so it could be accessed by the front end. The front-end will then use these classified frames, and the fact that there are 10 frames per second, to generate a list of classified timestamps for use in the Result Display page with the original video.

Resilience

The code below shows error handling in the front end to limit the video file size to 50 MB. This is applied when the user can upload a video from their local gallery. If the picked video file size is more than 50 MB, it shows an error on the screen saying "File size cannot exceed 50 MB" and the user can go back to upload another appropriate size video.

```
if (picked != null) {
  print(picked.files.first.size/(1024*1024));
  if(picked.files.first.size/(1024*1024) > 50){
    print('File size cannot exceed 50 MB');
```

As of now, we don't check whether the frames taken from the video include clear images of the persons facial features since it wasn't needed to test the prototype's functionality. However, we plan on performing data cleaning on the frames by using facial detection to minimize the image to only the face. If no facial features are detected in any of the frames, the data will not be applied to the model and a message will be sent to the front end that a clear video of the person is needed to identify the social cues.

Percentage Breakdown:

Research: 20%

Interface: 40%

- Includes the Flutter + Android download and install on all computer devices.
- Familiarizing ourselves with Dart syntax.
- Research functions writing functionality with Dart.
- Integrating firebase to send video to it from the front-end.

Systems: 40%

- Includes back-end Machine Learning model.
- Sending images to firebase.

Steps to build and run project:

- Front-end:
    1. Install + Setup Flutter and Android studio: https://docs.flutter.dev/get-started/install
    2. Open project in Android studio. (File > Open Folder)
    3. Set-up Flutter plug-in in settings. (File > Settings > Plugins > Install Flutter)
    4. If errors are shown in files, run 'pub get' in the terminal.
    5. Run project after connecting the tablet (Enable developer and USB debugging mode) https://docs.flutter.dev/get-started/test-drive?tab=androidstudio
    6. Home Screen/ Main Menu: Upload video or use video camera options
        i. Click on upload video
        ii. Select a 50 MB or less video from gallery/local device
        iii. Click play button to playback the video
    7. Home Screen/ Main Menu: Upload video or use video camera options
        i. Click on use video camera
        ii. Record a video using the camera
        iii. Click play button to playback the video
- Back-end:

1. Download and setup python on computer.
    a. https://www.python.org/downloads/
2. Use terminal to:
    a. pip install numpy tensorflow firebase-admin opencv-python flask
3. Read README in ModelControl/training to download pkl file.
4. Go to cue-cetera/assets.
5. Add pkl file to this folder.
6. Run python predictLabels.py

Project Documentation

https://github.com/AmaniN16/Cue-Cetera

References

[1] E. J. MARCO, L. B. N. HINKLEY, S. S. HILL, and S. S. NAGARAJAN, "Sensory Processing in Autism: A Review of Neurophysiologic Findings," *Pediatric Research*, vol. 69, no. 5 Part 2, pp. 48R54R, May 2011, doi: https://doi.org/10.1203/pdr.0b013e3182130c54.

[2] N. J. Sasson, D. J. Faso, J. Nugent, S. Lovell, D. P. Kennedy, and R. B. Grossman, "Neurotypical Peers are Less Willing to Interact with Those with Autism based on Thin Slice Judgments," *Scientific Reports*, vol. 7, no. 1, Feb. 2017, doi: https://doi.org/10.1038/srep40700.

[3] "Better Visual Support for Neuro-diverse Children," *IBCCES*, Jun. 12, 2017. https://ibcces.org/blog/2017/06/12/visual-support/ (accessed Apr. 19, 2023).

[4] "Project Overview ‹ Realtime Detection of Social Cues – MIT Media Lab," *MIT Media Lab*, 2023. https://www.media.mit.edu/projects/realtime-detection-of-social-cues/overview/ (accessed Feb. 16, 2023).

[5] M. Qodseya, "Visual Non-verbal Social Cues Data Modeling," 2018. Accessed: Feb. 16, 2023. [Online]. Available: https://oatao.univ-toulouse.fr/22588/1/qodseya_22588.pdf

[6] "Serving models: TFX: tensorflow," *TensorFlow*. Accessed: Feb 21, 2023 [Online]. Available: https://www.tensorflow.org/tfx/guide/serving.

[7] "Methods of integrating AI on Andriod and iOS," *Packt*. Accessed: Feb 21, 2023 [Online]. Available: https://subscription.packtpub.com/book/data/9781789611212/1/ch01lvl1sec06/methods-of-integrating-ai-on-android-and-ios.

[8] B. Ebrahim, "How to integrate Machine Learning Models into your app?," *Medium*, 2019. [Online]. Available: https://medium.com/@BdourEbrahim/how-to-integrate-machine-learning-models-into-your-app-b77717e2702. (Accessed: Feb 21, 2023).

[9] M. McKenzie, "Tensorflow serving practical introduction.," *Medium*, 2016. [Online]. Available: https://medium.com/osldev-blog/tensorflow-serving-practical-introduction-9ce29ccd63f. (Accessed: Feb 21, 2023).

[10] "Neural networks API: Android NDK: Android developers," *Android Developers*. [Online]. Available: https://developer.android.com/ndk/guides/neuralnetworks (Accessed: Feb 21, 2023)

[11] A. Anwar, "Types of regularization in machine learning," *Medium*, 07-Apr-2021. [Online]. Available: https://towardsdatascience.com/types-of-regularization-in-machine-learning-eb5ce5f9bf50 .

[12] Hu. Linda, Ge.Qi, ""Automatic facial expression recognition based on MobileNetV2 in Real-Time." Journal of Physics: Conference Series. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/1549/2/022136

[13] He K., Zhang X., Ren S. and Sun J. 2016 "Deep residual learning for image recognition," *Proceedings of the IEEE conference on computer vision and pattern recognition 770-778*
Google Scholar

[14] Chauhan Singh, Nagesh, " Optimization Algorithms in Neural Networks," *KD nuggets* [Online]. Available: https://www.kdnuggets.com/2020/12/optimization-algorithms-neural-networks.html

[15] Imam, Aadhil, “Build a Convolutional Neural Networks using tf.keras functional API,” *Medium,* 2021. [Online]. Available https://aadhil-imam.medium.com/build-a-convolutional-neural-networks-using-tf-keras-functional-api-cb528647f065

[16] Song, Zhenjie, “Facial Expression Emotion Recognition Model Integrating Philosophy and Machine Learning Theory,” *Schools of Humanities and Social Sciences.* [Online]. Available: https://www.frontiersin.org/articles/10.3389/fpsyg.2021.759485/full