A Deep Learning-Based Approach for Adaptive Virtual Learning with Human Facial Emotion Detection

Ishani Das[1] and Kent Paris[#]

[1]Cupertino High School, Cupertino, CA, USA
[#]Advisor

ABSTRACT

Classroom learning has become difficult since COVID-19 began. Students and educators have had to adapt to virtual learning by using the available tools and technologies. However, a virtual classroom does not simulate the same experience as a real, in-person classroom.

In this setting, teachers can immediately receive feedback on the students' understanding of content by analyzing their facial expressions. By doing so, they can take immediate action to create a more effective learning experience. For example, teachers can individually help students that express an emotion of confusion by reiterating the concept privately and in greater detail. This style of teaching allows educators to ensure every student is receiving the appropriate amount of support and guidance. With online learning, this method of adaptive teaching is compromised. In a virtual class with video conferencing software such as Zoom, it is not practical for a teacher to be able to constantly check each students' webcam while also teaching and managing technical difficulties.

Utilizing classification models in deep learning, an advanced subfield of machine learning based on neural networks, offers a novel approach to potentially working towards solving this issue. These models are trained with large datasets to mimic human behavior and achieve Artificial Intelligence (AI). The software simulates an effective virtual learning environment by using these methods to detect students' emotions from facial expressions and providing educators this real time feedback. In this study, it was discovered that a convolutional neural network classification model produced results with the highest accuracy of 55.0%.

Introduction

Objective

Teaching is an interactive process based on the dynamic stream of communication between a teacher and their students. This continuous, closed feedback loop allows teachers to adjust their instruction techniques or content to improve the overall learning outcome for all students in a class. Because verbal and student expression analysis are the most common methods of determining a student's emotion, a virtual learning environment often breaks this loop due to the teacher not having the ability to sell all the students' video stream or students are muted.

The objective of this project is to use Artificial Intelligence to re-establish this feedback loop by capturing the classroom video feed from of a communication system (e.g Zoom, WebEx, Google Meet, Amazon Chime, etc) and doing a series of real time processing steps involving deep learning model on that video to detect the students' emotions based on their facial expressions (e.g. happy, sad, netural, surprised, angry, etc).

This information is sent back to the teacher as a notification of feedback, therefore reinstalling the loop. To create a model with the highest prediction accuracy, a detailed study was done on various machine learning classification and deep learning models to understand the accuracy of each, as well as create an informed decision about what techniques can be used for the final engineered product.

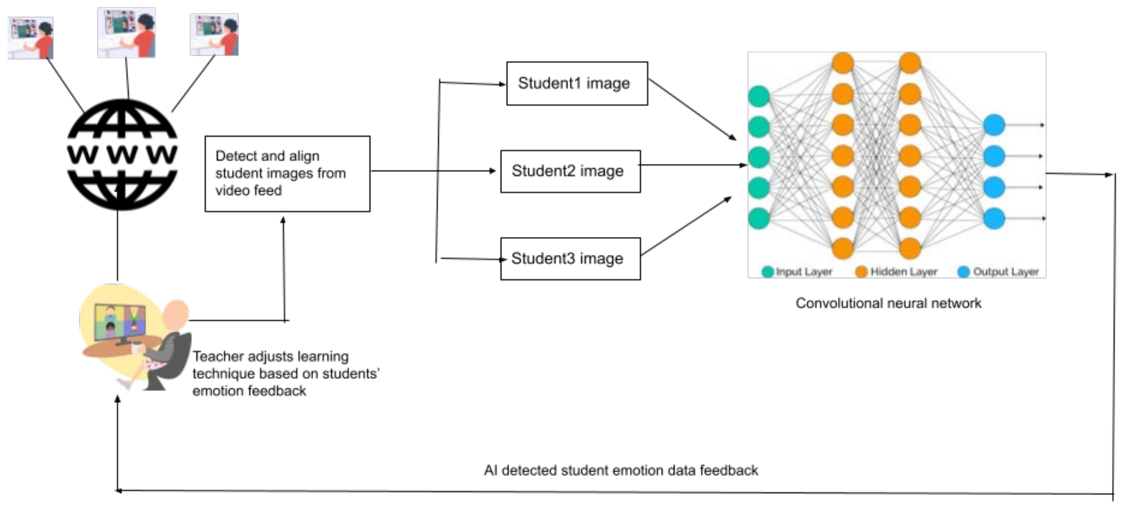

Detect and align student images from video feed
Student1 image
Student2 image
Student3 image
Input Layer
Hidden Layer
Output Layer
Convolutional neural network
Teacher adjusts learning technique based on students' emotion feedback
AI detected student emotion data feedback

Figure 1. Adaptive Virtual Learning Engineered System Software Workflow

Foundational Concepts

In machine learning, models are able to make a prediction or decision by applying knowledge of what it "learned" through an algorithm based on training data, rather than being directly told what to do.

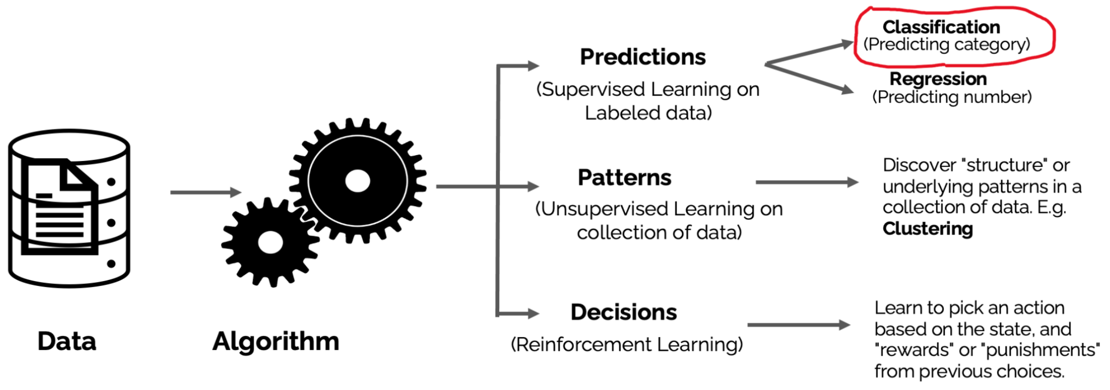

Data
Algorithm
Predictions
(Supervised Learning on Labeled data)
Classification
(Predicting category)
Regression
(Predicting number)
Patterns
(Unsupervised Learning on collection of data)
Discover "structure" or underlying patterns in a collection of data. E.g. Clustering
Decisions
(Reinforcement Learning)
Learn to pick an action based on the state, and "rewards" or "punishments" from previous choices.

Figure 2. Machine learning techniques and outcomes. Partial image diagram courtesy of Inspirit AI. [9]

With Artificial Intelligence (AI), machines mimic human behavior and machine Learning (ML) helps achieve AI through algorithms trained with data. The learning process follows the following steps:

a) Feeding the data into an algorithm, including prerequisite steps like feature extraction
b) Using this data to train a model.
c) Testing and deploying the model.
d) Using the deployed model to do an automated predictive task in a real use case.

Deep learning, an area of machine learning, is inspired by the structure of the human brain (i.e. a neural network). The learning process is considered "deep" because the structure of artificial neural networks consist of multiple input, output, and hidden layers. Each layer contains units that transform the input data into information that the next layer can use for a certain predictive task. Information is transferred from one layer to another layer connected by a weighted channel. Each neuron has a bias, a unique number associated with. This bias is added with a weighted sum of inputs to reach the neuron which is then applied on activation function. The result of activation function determines if the neuron is activated, which passes information to the next layer. To train the algorithm, weights and bias can be adjusted to achieve higher accuracy.

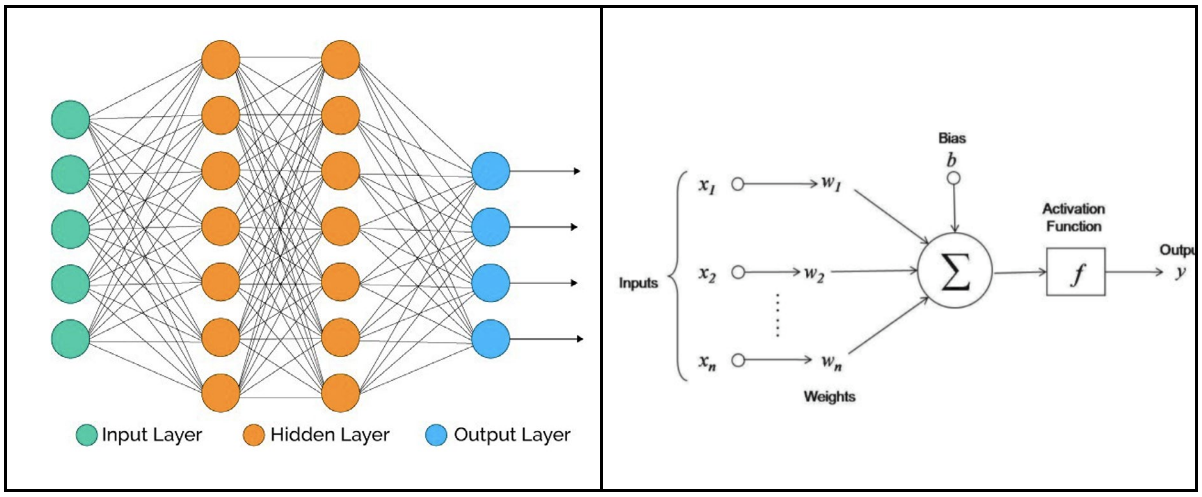


Figure 3. Deep Learning Neural Network [9] [13]

Image recognition with deep learning is a widely used and effective application of artificial intelligence and computer vision. The fundamental goal of image recognition is the classification of detected objects into different categories within an image. In traditional machine learning, classifications generated by machines are based on predefined features or rules. However, with deep learning, the features of large datasets are studied by neural networks without human intervention. Once we successfully detect the face and facial landmarks within an image, our next logical step is to analyze certain features to see how they correspond to particular emotions. To do so, we can find the distance between key facial points using the Facial Action Coding System (FACS), an emotion prediction system developed by Paul Ekman and Wallace Friesen[2]. This is done by analyzing the movements of individual facial muscles encoded by FACS that result in slight changes in facial appearance with "action units". It is a common standard to systematically categorize the physical expression of emotions within a category out of five universal human emotions: happy, sad, surprise, fear, and anger.

Action Unit 6: Cheek raised	Action Unit 12: Lip corner pulled
(4a)	**(4b)**

Figure 4. FACS Emotion Example: Happiness [2] (a) Action Unit 6, (b) Action Unit 12.

Methods

Materials

1. Custom Python software
2. FER-2013 dataset [3]

Methods for emotion detection

Two approaches using different classification models and machine learning techniques were experimented with to tackle the task of determining a human's emotion through analysis of facial expressions. The goal was to experimentally determine which model produced the most accurate results and could be used for the final engineered solution. When creating these models, several open source datasets and libraries were used, including Logistic Regression, K-Nearest Neighbors, Decision Trees, Neural Networks, and a Convolutional Neural Network.

The two approaches included:

1. Facial emotion detection using distance changes between facial landmarks
2. Facial emotion detection using a Convolutional Neural Network (CNN)

Facial emotion detection using distance changes between facial landmarks approach

The first step was acquiring the labeled dataset, FER-2013, containing 20,000 labeled 48*48 grayscale cropped images equally distributed with five emotions.[3] The data was stored in a dataframe with each row representing an image, and two columns representing 1) the pixel intensities of the image, and 2) an integer encoding of the target emotion label. The labels are mapped as follows:
label_map = {0: angry, 1: happy, 2: sad, 3: surprise. 4: neutral}

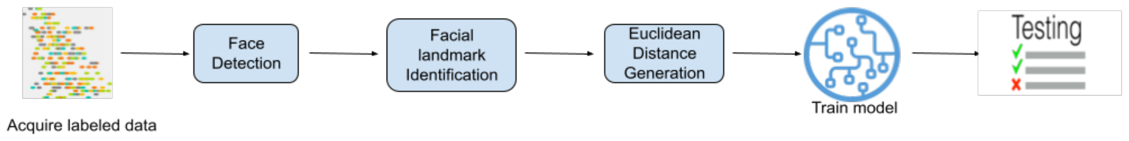


Figure 5. Facial Landmark Approach Workflow

Identifying the location of the face is the first step in the facial landmark-based emotion classification pipeline. Python's dlib[1] library's pre-trained face detection model extracts the bounding box coordinates of a face, helping eliminate parts of the image that have no relevance in detecting emotion. The second step in the pipeline is facial landmark detection, which is vital for feature extraction. Here, given an input image, a shape predictor finds key points of interest along the outline of a face to understand its structure. The number of key points on a face can vary depending on the pre-trained facial landmark model; in the case of dlib[1], it detects 68 2-Dimensional points on the human face. See the diagram below for an example.

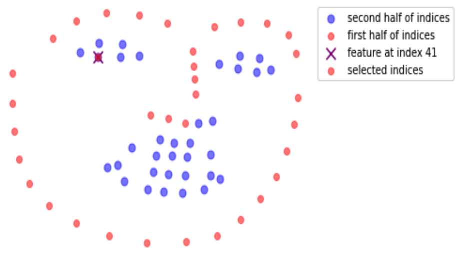
	
Original image	68 Extracted Facial Landmarks
(6a)	**(6b)**

Figure 6. Facial landmark testing with Dlb's pre-trained model. (a) Original Image, (b) 68 Extracted Facial Landmarks.

The next stage in the pipeline was defining the inputs for the model Here, the software essentially calculated the Euclidean distances between every pair of facial landmarks for each image in the dataset. These distances were used as features (x) into the model where the outputs (y) were emotions. This calculation allowed the model to distinguish finer details and distance changes with different emotions (e.g. an open eye vs a closed eye).

Once the Euclidean distances were calculated for the entire dataset, the model training process began. Three different classification models were used in this process to determine which had the highest relative accuracy:

1. KNeighborsClassifier (KNN) [4]: K-Nearest Neighbors is a machine learning algorithm that classifies a "point" in a certain category based on categorization of the "nearest" neighbor.
2. LogisticRegression [5]: This algorithm can be used to train models to understand the relationship between the dependent variable and one or more independent variables by estimating probabilities using a logistic regression equation. All in all, this helps the model predict the likelihood of an event happening or a choice being made.
3. DecisionTreeClassifier [6]: This is a classifier that continuously splits data according to a certain parameter. It consists of nodes (to test for the values of a certain attribute), edges/ branches (which correspond to the outcome of a test and connect to the next node or leaf), and lead nodes (a terminal node that predicts the outcome).

The last stage was testing, where 10% of the input dataset was used to check the accuracy of each model.

The Convolutional Neural Network (CNN) Approach

As described earlier, a neural network is a series of algorithms that aims to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. The previous experiment of using facial landmarks used various ML Classification models and performed at ~50 % accuracy. In this approach experiment was done with a Neural Network as well as a Convolutional Neural Network to compare and get better results.

The pipeline for this approach was:

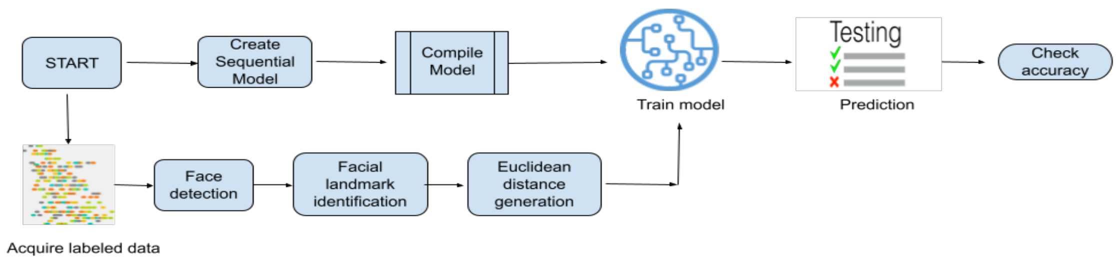
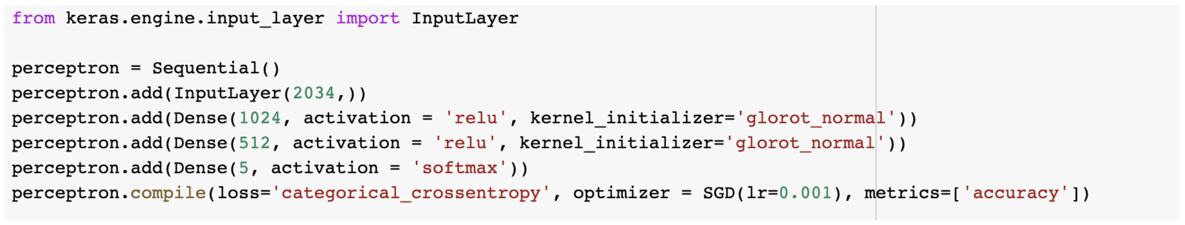


Figure 7. Neural Network Model Approach Workflow

The custom software used tensorflow and keras packages to build the neural networks which involved three parameters. [7]

1. The number of neurons
2. The activation of the neurons
3. The losses and metrics

For the experiment, a simple 3-layer neural network was built with an input of 2034, three hidden layers (1024, 512 neurons) and an output layer with 5 neurons.

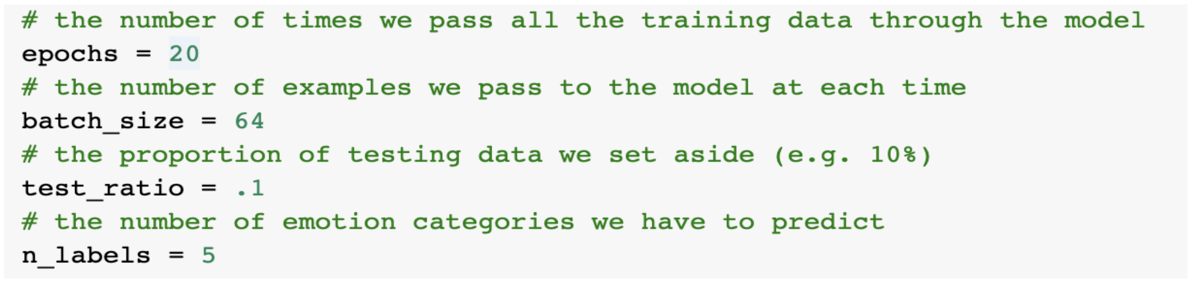


```
from keras.engine.input_layer import InputLayer

perceptron = Sequential()
perceptron.add(InputLayer(2034,))
perceptron.add(Dense(1024, activation = 'relu', kernel_initializer='glorot_normal'))
perceptron.add(Dense(512, activation = 'relu', kernel_initializer='glorot_normal'))
perceptron.add(Dense(5, activation = 'softmax'))
perceptron.compile(loss='categorical_crossentropy', optimizer = SGD(lr=0.001), metrics=['accuracy'])
```

Figure 8. 3-Layer Neural Network Model Creation Code Sample

The performance of a neural network depends on how it is trained. Although training is vital for stronger predictions, a model becoming too familiar with a partilat training dataset can result in overfitting.



```
# the number of times we pass all the training data through the model
epochs = 20
# the number of examples we pass to the model at each time
batch_size = 64
# the proportion of testing data we set aside (e.g. 10%)
test_ratio = .1
# the number of emotion categories we have to predict
n_labels = 5
```

Figure 9. Defining Model Factors Code Sample

For training, the same data was used (from the previous experiment), computed Euclidean distances between facial landmarks of the FER-2013 dataset.

```
# load data
dataX_pixels = np.load('pureX.npy')
dataY_labels = np.load('dataY.npy')
```

Figure 10. Defining x and y Matrices Code Sample

For the experiment, 90% of the data from the dataset was used for training and 10% was used for testing.

```
# split Data into Train, Test (90-10)
X_train, X_test, y_train, y_test = train_test_split(dataX_pixels, y_onehot, test_size=test_ratio, random_state=42)

#### Standardize the data ##########
pixel_scaler = StandardScaler()
pixel_scaler.fit(X_train)
X_train = pixel_scaler.transform(X_train)
X_test = pixel_scaler.transform(X_test)
```

Figure 11. Standardizing Neural Network Model Factors Code Sample

Later, this same experiment was repeated with a Convolutional Neural Networks (CNN) for Emotion Detection. CNN has a lot more layers and is good for image data. The building blocks of CNNs are filters a.k.a. kernels. CNN has primarily 2 stages. Convolutional layer is what lets us find the pattern anywhere in the image at once. Kernels are used to extract the relevant features from the input using the convolution operation. And in the fully connected dense layer the output data is generated using the data from the convolution layer.

```
# initialize model
cnn_model = Sequential()
# this conv layer has 64 filters! the input shape needs to be the same dimensions of the image
cnn_model.add(Conv2D(64, kernel_size=(2, 2), activation='relu', input_shape=(width, height, 1)))
# batch normalization
cnn_model.add(BatchNormalization())
# max pooling
cnn_model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
# dropout
cnn_model.add(Dropout(0.5))

# flatten all the outputs between convolutional and dense layers
cnn_model.add(Flatten())
# add a "dense layer" (i.e. the fully connected layers in MLPs) with dropout
cnn_model.add(Dense(512, activation='relu'))
# output layer
cnn_model.add(Dense(n_labels, activation='softmax'))
```

Figure 12. Standardizing Convolutional Neural Network Model Factors Code Sample

Results

Facial Landmark Approach: Emotion Prediction Analysis		
Logistic Regression Classifier	KNN Classifier	Decision Tree Classifier

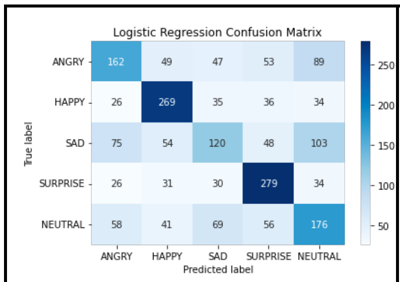
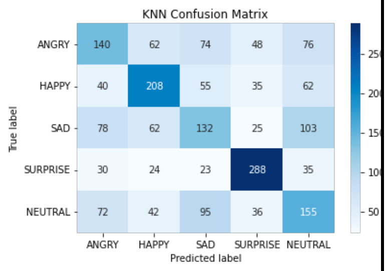
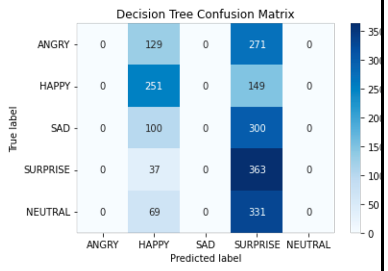
		
50.30% accuracy	*46.15% accuracy*	*30.70% accuracy*
(13a)	**(13b)**	**(13c)**

Figure 13. Facial Landmark Approach Emotion Prediction Analysis. (a) Logistic Regression Classifier, (b) K-Nearest Neighbors Classifier, (c) Decision Tree Classifier.

Using the approach of identifying facial landmarks, the machine learning model's facial emotion detection accuracy scores were ~20% less than a standard human's prediction of 72%. [8] Specifically, the model resulted in accurate emotion predictions for 30.7% , 46.15%, and 50.3% of the testing data using a Decision Tree, K-Nearest Neighbor, and Logistic Regression respectively. Upon analysis of each classifier's performance, it can be determined that the model can most accurately predict the "surprise" emotion, but least accurately predict the emotion "sad".

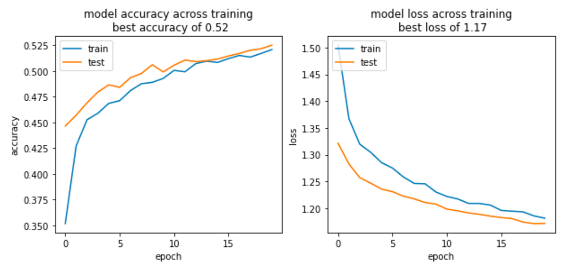
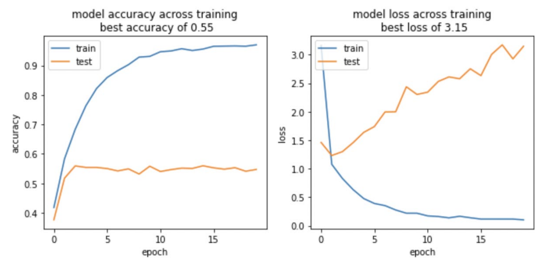
Neural Network Approach: Emotion Prediction Analysis	
Neural Network	Convolution Neural Network
	
52.00% accuracy	*55.00% accuracy*
(14a)	**(14b)**

Figure 14. | Neural Network Approach Emotion Prediction Analysis. (a) Neural Network, (b) Convolutional Neural Network (CNN).

From the analysis of each of the following models: a neural network and a convolutional neural network, it can be observed that these Deep learning models are significantly more accurate compared to the models using the first approach of calculating facial landmarks. Although both of the models had a strong performance, the convolution neural network model shows signs of overfitting, a concept in machine learning that refers to a model that models the training data too well. This may be due to a high number of epochs, the number of times the

training dataset enters the model, which results in the model learning the outliers and minute nuisances of the specific data. Resolving this issue can be approached by reducing the number of hidden layers or using dropout layers, which randomly remove certain features by setting them to 0.
The CNN model resulted in the highest accuracy in the facial emotion detection prediction study.

Discussion

All in all, the results from the facial emotion detection study using various ML and deep learning models were very promising. It was concluded that using a CNN based facial emotion detection model would be the best choice for the target engineered solution of creating an "Adaptive virtual learning system". As a product, this software can aid educators during virtual learning sessions by detecting emotions and sending feedback to teachers in real time. While focusing on teaching, screen sharing, and taking care of technical difficulties, it is hard for teachers to also watch their students' faces to check for general comprehension. With this tool, educators can teach more easily and effectively while also stimulating the same degree of "observation" as in-person classes. The idea of the closed loop system has been described with illustration in the objective section.

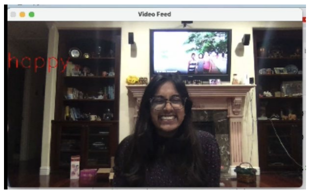
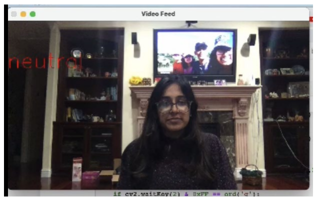
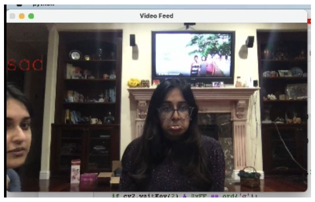
Conclusion

Engineered Solution

The following key steps and softwares were decided to be used to engineer the final "Adaptive virtual learning system".

1. Utilize a specific virtual meeting system for online education (e.g. WebEx, Zoom, GoogleMeet, Amazon Chime etc)
2. Capture individual participants' video feeds for facial emotion detection. This can be done via two methods:
    a. Receiving the video feed from the vendor SDK (Software Development Kit) of the virtual meeting system (e.g. WebEx, Zoom, GoogleMeet, Amazon Chime etc), provided the vendor supports this capability.
    b. Capturing the sample screen images from educators' machines during the meeting session.
3. Identify individual participants' face boundaries from the images, because they may contain areas unnecessary for emotion detection.
    a. Utilize an object detection machine learning algorithm such as OpenCV's Frontal Face Haar Cascade. [11]
4. Utilize a CNN-based deep learning facial recognition system such as DeepFace to predict facial emotion. [9]
    a. After cropping the image to only preserve the area necessary for emotion prediction analysis, preserve the 'dominant_emotion' for each meeting participant in a data structure upon analyzing using the DeepFace CNN-based model.
5. Create custom Python software to integrate steps 1-4 of the workflow and develop the ability to run as an independent tool.
    a. Utilize the 'pyscreenshot' and 'time' Python libraries to take continuous screenshots of the virtual meeting screen.
    b. Display the analysis output/educator feedback on an external window using the following libraries: ipywidgets and tkinter.

A drawback of the approach discussed in step #2b is that the screenshot of the educator's screen will not be able to capture the webcam feeds of all students in a large virtual classroom. For that reason, it is important to consider step #2a as another option. Here, the custom software captures the individual video feeds of each student via the vendor SDK. However, this information may not be publicly available due to privacy reasons. Because of this roadblock, I decided to utilize the approach outline in step #2b when testing the model using my laptop's web camera video feed with my sister. The results were very promising; the system took an image sample of the video feed at a frequency of every 5 seconds and was able to detect the emotions at an accuracy of 90% (9 out of 10 times)! The increase in accuracy in the final engineered solution was possible due to the use of the open source facial recognition system DeepFace [9] , an extremely sophisticated convolutional neural network. The demonstration of my test from the engineered solution is linked here [12], and examples of the algorithm predicting each of the five emotions are below.

		
happy	*neutral*	*sad*
(15a)	**(15b)**	**(15c)**

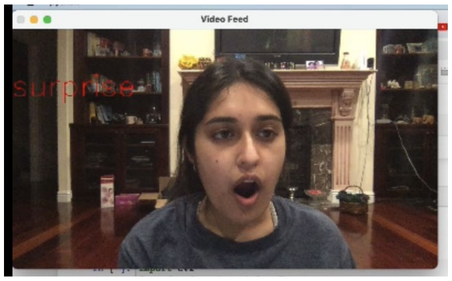
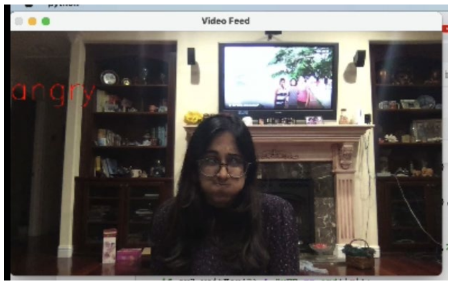
	
surprise	*angry*
(15d)	**(15e)**

Figure 15. Software Emotion Predictions. (a) "Happy" Prediction, (b) "Neutral" Prediction, (c) "Sad" Prediction, (d) "Surprise" Prediction, (e) "Angry" Prediction.

Future Work

1. If granted permission by the meeting participants and the virtual meeting system's security & privacy guidelines, developing an automated system that is able to capture the real time video feed of individual participants, rather than a screen capture, will allow for a more efficient feedback loop that will allow educators to adjust their teaching mechanisms.
2. Current feedback mechanism was just text on the video feed. This can be improvised by creating sound or other notification mechanisms.

3. Also it is necessary to test the system performance and resource utilization (CPU, memory) while using this tool as CNN is very computation heavy.

Limitations

Integrating this software into well-known video conferencing platforms such as Zoom, Skype, Webex, and Google Meet may be difficult based on their open-source software guidelines. Additionally, the process of productization would require permission based on primacy guidelines from each of the companies and meeting participants to have access to the meeting participants' video feed.

Acknowledgments

I learned ML and AI techniques from various online courses, videos, papers, and artifacts. Additionally, I used software and data from multiple open source communities whose names are mentioned in the "References" section. I am very thankful to all of them for providing me with the necessary background information.

I am also very thankful to "Inspirit AI", an online learning community whose development tools, resources and framework I used for doing research, coding and experimentation.

References

[1] dlib: "Dlib." *PyPI*, https://pypi.org/project/dlib/.
[2] Facial Action Coding System: "Facial Action Coding System." *Paul Ekman Group*, 30 Jan. 2020, https://www.paulekman.com/facial-action-coding-system/.
[3] FER-2013 Dataset: Sambare, Manas. "Fer-2013." *Kaggle*, 19 July 2020, https://www.kaggle.com/msambare/fer2013.
[4] K-Nearest Neighbors (KNN): "1.6. Nearest Neighbors." *Scikit*, https://scikit-learn.org/stable/modules/neighbors.html.
[5] Logistic Regression: "Sklearn.linear_model.Logisticregression." *Scikit*, https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.
[6] Decision Tree Classifier: "Sklearn.tree.decisiontreeclassifier." *Scikit*, https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html.
[7] Sequential Model: Team, Keras. "Keras Documentation: The Sequential Model." *Keras*, https://keras.io/guides/sequential_model/.
[8] Human Emotion Prediction Analysis: Ucl. "Artificial Intelligence Still Lags behind Humans at Recognising Emotions." *UCL News*, 28 Apr. 2020, https://www.ucl.ac.uk/news/2020/apr/artificial-intelligence-still-lags-behind-humans-recognising-emotions#:~:text=28%20April%202020.
[9] CNN Diagram: Courtesy of InspiritAI; "Ai Taught by Stanford/MIT Alum for High School." *INSPIRIT AI*, https://www.inspiritai.com/.
[10] Grayscale Test Image: https://www.theclickcommunity.com/blog/wp-content/uploads/2014/10/black-and-white-portrait-of-man-with-his-eyes-closed-by-Brian-Powers.jpg.
[11] OpenCV FrontalFace HaarCasacde: Avelino. "PYTHON-OPENCV-DETECT/HAARCASCADE_FRONTALFACE_ALT.XML at Master · Avelino/Python-Opencv-Detect." *GitHub*, https://github.com/avelino/python-opencv-detect/blob/master/haarcascade_frontalface_alt.xml.
[12] Realtime Facial Emotion Detection Demo: *YouTube*, YouTube, https://www.youtube.com/watch?v=29_gSPR8jkg
[13] CNN Logic Diagram: Imgur. *Imgur*, https://i.stack.imgur.com/YwCCU.png.