Real-time Emotion Estimation: Progress Report

Problem statement

Emotions play a crucial role in human interaction. Understanding them is fundamental in establishing meaningful relationships and productive communication. For many individuals, deciphering these emotional cues is challenging, especially for those with social or cognitive disorders. For example, some individuals with autism spectrum disorder (ASD) have varying degrees of difficulty in differentiating between neutral and emotional facial expressions. Such difficulty ultimately leads to the misclassification of emotional expressions (Eack, et al.). Considering that emotional awareness is a significant component in understanding person-to-person communication, providing real-time feedback on emotional states based on facial information would be helpful for improving communication between individuals. With advancements in AI, machine learning models can be leveraged to help bridge this gap. This can provide more context of a speaker's visible emotional state. Such context would be particularly pertinent in a virtual setting, such as a video call.

The objective of this project is to employ image classification techniques to analyze facial expressions in real-time, providing instantaneous emotion estimations. The system will classify facial expressions into specific emotional categories and display the top predictions based on confidence scores. This will be presented in an intuitive and interactive front end, providing immediate emotion estimation based on incoming video data of a real-time interaction.

Data set description

The dataset, titled FER2013, is sourced from Kaggle and contains labeled images representing various emotions.

Data Set: https://www.kaggle.com/datasets/msambare/fer2013

Specifications:

1. Features: The dataset comprises grayscale images of faces, each labeled with an emotion.
    a. Images have been centered and all occupy similar space
2. Number of images: 35,887 images.
3. Labels (Emotions covered): Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral (labeled from 0-6 respectively)
4. Image Format: Grayscale, 48 by 48 pixels
5. Dataset Split: The dataset is categorized into training and testing.
    a. We further split the training dataset into training and validation (80 - 20)

Data preparation

The FER2013 dataset was partitioned into a training and testing set when we received it. We did, however, have to partition the training dataset into our training and validation. We chose to do an 80-20 split, where 80% of the data is for training and 20% of the data is used for validation. We also scaled the data by dividing each pixel value by 255. With this scaling, each pixel value now lies within the range [0,1].

Using keras, we set up ImageDataGenerators that allow for data augmentation. This is a technique that increases the training set size by introducing new training instances that are the result of applying a transformation to the original instance. It counteracts risks of overfitting by requiring the model to become more resilient to various discrepancies in images of the same class (Géron). For instance, using data augmentation could make a model more likely to accurately detect the emotion of a facial expression, whether it is oriented 'normally' or if it is rotated.

We use several features of ImageDataGenerators to apply data augmentation. As mentioned above, ImageDataGenerators can take care of rescaling pixel values. Furthermore, we apply rotation_range in order to get random rotation amounts for each image. We use shear_range to randomly shift or stretch certain aspects of images. The zoom_range accounts for randomly adjusting the size of the faces shown by either zooming in or out by a certain amount (in our case, in or out by 30%). We also randomly flip along the horizontal axis by setting horizontal_flip=True. We also set fill_mode="nearest" to determine how pixel values are filled in when a transformation results in empty regions of space (e.g., zooming out).

We then use the flow_from_directory method from the ImageDataGenerator to specify where the data will be loaded from and apply some parameters to how they should be loaded. This includes specifying the directory to load from and identifying what types of images there are. For instance, specifying that the images are grayscale, the target_size of the images, the desired batch_size, the fact that we are using categorical data, and the fact that we want the data to be shuffled.

Model and/or technique selection

The problem we are trying to solve is fundamentally a computer vision problem. It deals with trying to make sense of digital images, or more specifically, trying to classify emotions observed in images of facial expressions. In the field of computer vision, there are two main techniques; deep learning and convolutional neural networks (*What is comp…)*.

Neural networks are a subset of machine learning concerned with mimicking the human brain's system of neuron signaling. Neural networks in machine learning are made up of layers between the model's input and output. These layers are broken up into the input layer, the hidden layers,

and the output layer. Each layer consists of nodes which receive input data, perform an evaluation on the input data, and deliver an output to other nodes dependent on a threshold. Most neural networks are unidirectional from the input layer to the output layer, referred to as feedforward. However, models can be trained through backpropagation, which involves moving the data from the output layer to the input layer. The purpose of backpropagation is to fit the model's parameters by calculating the attribute error associated with each neuron. Through backpropagation, we are recursively finding loss of layers, allowing for the optimization of weights at each node (*What are neural…)*.

Deep learning is a subfield of neural networks that is associated with a high depth of layers, specifically 3 or more (*What is deep…)*. Deep learning encompasses techniques such as recurrent neural networks, deep neural networks, and convolutional neural networks. Most deep learning techniques tend to be insufficient for image processing tasks. For example, consider a deep neural network; it has fully connected layers, and must flatten data into a 1-dimensional format. The former characteristic means that for large images, a deep neural network will have an excessively high number of connections, which will inevitably make the algorithm inefficient and slow.

Convolutional Neural Networks (CNNs) are the primary model used for processing large images efficiently. They employ a specialized architecture tailored for image data processing that puts them far ahead of other deep learning techniques. CNNs use convolutional layers, where neurons are connected to receptive fields within the previous layer rather than the entire input image. This allows CNNs to focus on low-level features in early layers and combine them into higher-level features in later layers. This hierarchical structure is a common feature of actual images, making the algorithm apt for image processing.

Another benefit of CNNs is that both input data and convolutional layers are represented in 2 dimensions, meaning that the matching of neurons with their inputs is easier. With this, image data in the first layer is already represented in 2 dimensions, and spatial relationships in the image are preserved.

In a CNN, neuron's weights are called filters, or convolutional kernels. These filters are small image patterns that, when applied to an image, can detect patterns that an image has. During the convolutional layer, CNNs learn the best filters for accurate predictions. Neurons sharing the same filter output a feature map. This significantly reduces parameters as the CNN learns the same pattern at all locations in the input space. This identified pattern can then be applied to the subsequent layer as an input.

Along with the convolutional layers in a CNN, we employ pooling layers to transform the data. The purpose of the pooling layer is to decrease the dimension of the feature map and eliminate

redundant details (Ramya, et al.). This is important considering that convolutional layers record the precise position of the features in the input. Thus, alterations to the image, such as shifting or rotation, will result in a different feature map. The pooling layers can mitigate this through downsampling, where the resolution of the input is decreased. This eliminates the fine details while maintaining the important structural elements. Pooling specifically occurs after the ReLU has been applied to the feature map output of the convolutional layer. The actual operation performed in the pooling layer involves applying a pooling filter, usually 2x2 pixels large, and striding it across the image by a factor of 2 with each iteration. This specific implementation would decrease the size of each dimension in the feature map by 2. Pooling is an operation that is specified rather than learned within the model. The two most common pooling operations are average pooling, where the average of each patch of the feature map is calculated, and maximum pooling, where the maximum of each patch of the feature map is calculated (Brownlee).

Our model comprises multiple convolutional layers of increasing sizes for feature extraction. This starts with a small number of filters which slowly increases. This allows the model to learn simple patterns at first, gradually accounting for more complexity as time goes on. There are also max-pooling layers to reduce spatial dimensions and mitigate overfitting. To enhance the network's generalization capabilities, dropout layers have been added. This generalization allows for random neurons to be ignored, helping with avoiding overfitting. A fully connected layer with 512 neurons is used for the final classification, and an output layer with seven neurons, each corresponding to a distinct emotion class, uses the softmax activation function to output a probability distribution. We chose the Adam optimizer for its efficiency, and used categorical cross-entropy as the loss function to train the model for multi-class classification.

Future Work

The first course of action for future work will be tuning hyperparameters and altering the model to improve performance metrics. In our first evaluation, we found that the accuracy of the model was about 67%. This accuracy is far weaker than desired. Thus, we will need to perform a full analysis on the current state of the model to improve its performance. For instance, if the model is overfitting, we may need to add regularizers to generalize estimations. We may also need to apply more preprocessing to the data before training and testing the model. Improving this performance will also include altering hyperparameters within the model. This will involve a process of trial and error to observe what set of hyperparameters maximizes recall, precision, and accuracy. Once the model has been tuned to a desirable degree of accuracy, we will move onto creating an interactive front end. This will also involve establishing real time estimations which will be displayed on the front end. An additional component will be creating a secondary model to discover where the individual’s face is in the real time video, cropping out an image of their face to send to the model. This addition should improve estimations during field testing as the images inputted into the model will be more representative of the training and testing data.

References

Brownlee, J. (2019, July 5). *A gentle introduction to pooling layers for Convolutional Neural Networks*. MachineLearningMastery.com. https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/

Eack, S. M., Mazefsky, C. A., & Minshew, N. J. (2015, April). *Misinterpretation of facial expressions of emotion in verbal adults with autism spectrum disorder*. Autism : the international journal of research and practice. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4135024/

Géron, A. (2020). Chapter 14. Deep Computer Vision Using Convolutional Neural Networks. In Hands-on machine learning with scikit-learn, Keras, and tensorflow: Concepts, tools, and techniques to build Intelligent Systems (pp. 578–647). essay, O'Reilly.

Ramya, R., & Venkatesh, S. (n.d.). *Pooling layer*. Pooling Layer - an overview | ScienceDirect Topics. https://www.sciencedirect.com/topics/mathematics/pooling-layer

What are neural networks?. IBM. (n.d.-a). https://www.ibm.com/topics/neural-networks

What is Computer Vision?. IBM. (n.d.-b). https://www.ibm.com/topics/computer-vision

What is deep learning?. IBM. (n.d.-c). https://www.ibm.com/topics/deep-learning