Batch Processing Facial Emotion Recognition Predictions

Christopher P. Monyok

Engineering & Computer Science, Syracuse University
Syracuse, New York U.S.A
cpmonyok@syr.edu
2021/03/13

Abstract

The intent of this research project is to demonstrate the use of facial expression recognition and explain how the technology for emotion recognition can derive perceived emotions from facial expressions. Additionally, industry best practices for emotion detection and machine algorithms processing of images will be discussed as well as the standard methodology to approach this use case.

Intro

Humans can subconsciously pick up on subtle clues to the emotions of people they are in contact with. These emotions are displayed through voice tones, physical posture, word choices, and of course facial expressions. This instinctual ability to process these clues to deduce an emotion likely stems from the fact that humans from all cultures share the same basic core emotions and mechanisms for displaying them. [1]

The premise of emotion detection is to take these facial expressions and determine the emotional state of the subject based on focal points such as the position of the mouth, brow, and even subtle postures in the tip of the nose.

However, next generation emotion recognition systems are looking to refine the methodology by incorporating context awareness into the algorithm. [6] For instance, in a facial emotion recognition system, if an image of a person's head displays the mouth wide open and the brow neutral there is a high probability this image would be classified as surprised. However, if the entire image in which this head shot was cropped from showed the person blowing out birthday candles a human would associate the mouth being open with the action and know, without certainty, the subject was not surprised. [7] This same scenario would be repeated for other actions. Another example being an image of somebody lifting weights. The cropped head image would be labeled as angry however the context would point to the action of strain.

This adding context not only improves accuracy by eliminating false classifications due to facial expression because of actions but also helps in classification in scenario's where accuracy is impeded as a result of partial occlusions,

non-frontal image captures, and other performance degrading issues.

For computer vision learning one of the more popular open-source library models available is the TensorFlow Keras Sequential Model. This model serves as the foundation for building several types of supervised deep learning algorithms such as Artificial Neural Networks (ANN), Feed Forward Networks (FNN), Restricted Boltzmann Machines (RBMs), Recurrent Neural Networks (RNN), and Convolutional Neural Networks (CNN). [2]

For this project, a CNN was chosen as this type of deep learning supervised algorithm is the most used for facial emotion recognition. This is due to its efficient processing of images, accuracy, tunability, and ease of use.

Before exploring Facial Emotion Recognition technical implementation, it is important to understand the machine learning algorithm that processes the images. Therefore, a brief introduction to Artificial Neural Networks and, more specifically, CNN's is needed.

"Artificial Neural Networks (ANNs) are computational processing systems of which are heavily inspired by way biological nervous systems (such as the human brain) operate. ANNs are mainly comprised of a high number of interconnected computational nodes (referred to as neurons), of which work intertwined in a distributed fashion to collectively learn from the input to optimize its final output."[2]

As with the architecture of any Neural Network, a CNN is made up of multiple layers that consist of connected nodes. Each layer feeds the next layer in the architecture. The general baseline architecture for a CNN is an input layer, output layer, and hidden layers. For a CNN the hidden layers are made up of three main layers. These layers are the Convolutional layer, Pooling layer, and Fully Connected layer.

The Convolutional layer's purpose is to apply filters to the image. These filters take a small subset of a pattern from a target images and apply that to the matrix mapped pixels of the current image being processed. This is repeated for each section of the image in blocks until the entire image is scanned. When placing the filter over each block a score is given to the pixel section indicating whether it is a match or not. This process is repeated for each pattern identified in the target image.

For a verification system where a face is being matched against a template this could result in several dozen pattern filters being processed over the image presented. However, for expression recognition the number of patterns significantly increases the search space directly proportionate to the target expressions used. In the case of this research project, I am using seven target emotions which results in a total number of patterns equal to filters per target expression times 7.

Once the filters are processed over the image with a Boolean value indicating a match or not per corresponding section a normalization processes are run over the values to ensure the numerical values are manageable and maintain the integrity of the model. For deep neural networks, the sigmoid and tanh activation functions give poor results due to the vanishing gradient effect. So instead, normalization is primarily

accomplished by setting negative values to zero using a process known as ReLU. ReLU stands for rectified linear unit. While this activation function has the desired linear function properties it behaves as a hybrid. This activation function is standard within the TensorFlow Keras Sequence Model used to build deep neural networks.

The pooling makes the image smaller. It accomplishes the reduction by looking at the image in portions called slides. At this point the image is in a matrix representation and slides represent a block of that matrix. For example, a slide could be a block of four cells. Each cell has a number that corresponds to the value given after the convolutional layer ran filters over the image giving each cell a value and the activation function has normalized the values. This layer will take each block that is covered by the slide and determine the max value of the cells. The max value is then copied to a new matrix where one cell represents the cells that were within the slide of the first matrix. So, if the slide covered four cells those four cells would be reduced to one cell. This effectively reduced the image by 75%. This method helps performance of the algorithm in several critical ways. Firstly, it reduces the images representation size significantly which allows better performance of the next layer. Secondly, it makes the algorithm less sensitive to noise or anomalies in the image that could cause false non matches due to poor image quality, positioning, or other factors.

The fully connected layer is the final layer that uses the input from all the previous layers and votes as to the classification. This vote is in the form of a confidence score. The higher the score the more probability the algorithm is correct. This layer, as with other layers in the architecture, can be stacked. However, it is important that the final layer has the same number of nodes as possible classifications. In this case it will have seven nodes representing the seven possible output classifications.

In its simplest form facial emotion recognition algorithms detect the edges of the eyes and mouth and match those edges in relationship to the entire feature. This provides details as to whether the person is smiling, frowning, or displaying other common easily detectable clues to emotion. One method for developing algorithms used for edge detection is the Harris corner detection method. This looks for the edge by running a filter over an image and looking for a scenario where there is a large gradient change in all directions. [1]

Before running this algorithm, color images need converted to grey scale. Then a gaussian filter, or similar functionality filter, needs run over the image. This filter removes the graininess of the photo so there is not an overabundance of edge detection, but instead just primary edges are detected.

Next for this approach, the image is sectioned off in horizontal bands. Each band is searched for corners. The band with the most corners within this horizontal band is extracted as the eye feature and analyzed. This works because for two eyes you will have 8 edges on average. This is the edge of the eye, transition from white of eye to pupil, the other side of the pupil, and then the next corner of the eye. These same points are detected for the second eye. Nowhere else on a face without accessories

will an algorithm find that many edges across a horizontal band.

After this section is detected, indicating eyes or mouth for a secondary score, these portions of the image are cropped and turned into a grid. This grid is analyzed by evaluating the pixels x and y coordinates if they meet a predetermined threshold that indicates intensity. Essentially, a measure that allows traversal of the pixels along the edge.

The next part of the Harris Corner Detection algorithm is to detect the corners of the edge. The is computed using the HOG feature vector calculation. This can determine the direction of the edge.

Alternatively, a Sobel Operator algorithm can be used to detect the edges for the preliminary portion of the algorithm. This algorithm detects edges by running a matrix computation against window sections of the image to detect not only edges but also angular direction of the edge. The computation primarily indicates whether there is a color change by providing deltas between sections of the matrix.

Prior Work

There are dozens of Facial Expression Recognition (FER) datasets available today to train algorithms to recognize emotions through facial feature extraction. The datasets classify the images into emotions originally based on the six Ekman basic classifications. [3] An additional emotion, making the total seven, is now considered the standard set. These seven core emotions are anger, contempt, disgust, fear, happiness, sadness, and surprise.[4]

However, understanding the way in which these datasets were originally created and classified is crucial to the understanding of the algorithms and more importantly how to fine tune the algorithm or improve upon existing algorithms.

The feature extraction points used for Face Emotion Detection are a result of numerous scientific studies over the years that combine traditional metrics from psychology with facial feature extraction. The collection of this data makes up what is known as multimodal databases. One experiment that followed this methodology (similar to how early researchers would have categorized these images into emotions before extracting facial features to train a FER) is outlined in the experiment discussed.

The experiment was called, “A Multimodal Database for

Affect Recognition and Implicit Tagging” and was published in IEEE. This experiment conducted by Mohammad Soleymani was separated into two phases.

For the first part of the experiment subjects watched videos and biometric measures were captured which are known to correspond to certain emotional responses. [5]

Biometric Captured	Description
GSR	Average skin resistance
ECG	A heart rate measure
Respiration Pattern	A calculated measure that sums rate, rhythm, and depth of breath
Skin temperature	Temperature reading
EEG	Measurement spectral power of 14 electrodes placed on the skull

Figure 1

These measurements were compared against knowledge from previous studies calculating emotional responses as well as receiving a categorization from the subject. These video clips were then labeled appropriately. After this the researcher decided which feature capture points would differentiate between the categories the most. This feature extraction process was manually conducted on all of the images. These Extraction points and the dichotomy between the measurements would serve as the methodology for categorizing unseen images. [5]

For the second part of the project videos from the Hollywood Human Actions Database were used

to validate the feature extraction

Category	Number of Images
Angry	3995
Disgust	436
Fear	4097
Happy	7215
Neutral	4965
Sad	4830
Surprise	3171
7 Categories	**28,709**

Figure 2

methodology that was manually done in phase 1. These videos ranged between 12 and 22 seconds in duration and were labeled by the algorithm using the technique conceived in phase one. Subjects were asked to label the videos based on their emotions in viewing the clip so that the researcher could compare this to how the algorithm to the unseen data. [5]

While this is a simplistic summary of the process used to originally create a categorized perceived emotion dataset it clearly illustrates the lengthy process needed and the science behind developing datasets that could be used for this work. A similar methodology would be needed if more advanced ranges of emotion expression were to be used in the future.

Experiment Design

The experiment was setup using the FER-2013 database provided by Kaggle and is licensed under the Open Database license. This database consists of the following 7 image categories.

These images are 48x48 pixel grayscale images of faces which have been centered as part of the database offering.

A Jupyter Notebook using python 3.7 was used as the Integrated Development Environment (IDE). Using a Jupyter Notebook allows for execution of individual blocks of code as well as utilizing Markdown language to document the work for presentation and sharing purposes. The individual execution allows for experimentation in development without the need for running all the code which serves to significantly reduce run time in some circumstances. An additional benefit is before each code block mark down language can be used to document the steps and what the block of code is doing. Then, after execution, the results can be displayed immediately after the description and block code. This provides a nice format for presentation of the work allowing a much more descriptive and intuitive artifact that could be produced in a tool such as PowerPoint. The Mark Down language has a rich set of features for presenting, very similar to HTML, though not as extensive.

Several libraries were used for different capabilities within the project. TensorFlow Keras was used for the Convolutional Neural Network architecture as well as several data processing activities needed before feeding the data to the network. SciKit-Learn was used for several metrics to evaluate the performance of the CNN and produce data to be used for graphical representation of the metrics. Seaborn and Matplot lib were used for the data Visualizations. Numpy and Pandas were used for data manipulation throughout the project. Lastly, several other standard commonly used libraries were counted on for common programming task such as file manipulation.

To get started the training directories were iterated through and a count of images in each directory counted. This ensured that the images could be read and gave perspective into the proportions of images in each category. The proportions would be important to use latter on in the analysis phase of the accuracy results. If one category were underperforming consistently it could be a result of the low image count for that category.

Next, training and validation datasets were created. The was accomplished by taking the training data and splitting it 80/20. 80% was dedicated to training activities and 20% was used for validation purposes. Creating this split allows the validation to occur with unseen data which is an important practice to follow to avoid overfitting of the model. These datasets were created to run in batches that randomly selected images. The accomplished two task. Firstly, it prevented the model from seeing all of one class then all of the next which would cause it to train for a specific type of image then retrain for a different emotion. Using the random feature allowed the model to optimize for all seven classes, Secondly, using the batches would allow a manageable size of data memory constraints could handle to be processed at a time. Loading all the data into memory at once would cause inefficiency slowing down the training if not preventing it from completing entirely. These smaller sizes of data also allow for targeting processing on Graphics Processing Units (GPU). To utilize that functionality, the data being processed must fit into the

GPU memory at once. Batching allows this to happen in the smaller memory of a GPU.

After the batches were created a small cell of code was written that created a nine-by-nine grid of image previews. The was done to ensure that the random feature of the batches was functioning as expected and to randomly sample the quality of the images that were to be used for training.

Figure 3

The next step was to actually create the Convolutional Neural Network (CNN) architecture model. For the model, a TensorFlow Keras Sequential model was used. Several architectures were experimented with before settling on two very different architectures to leave in the project. One being the highest performing architecture achieved in this research project and the other being the lowest scoring.

When creating these models there are a few key concepts that need adhered to for successful model training.

- Scaling - this is to normalize the values between 0 and 1. 255 is used because the dataset is grayscale.
- Flatten- this takes a 3d image and converts to 1d.
- The last layer must have the same number of nodes as the categories being predicted. In this case it is 7.

The most successful model consisted of 6 dense layers prior to the fully connected layer. A fully connected layer is the layer starting with the flattening layer through the output layer which has several nodes equal to the categories being predicted. In this case, with the model being a multi-class problem, the output nodes were 7. For binary problems, the model has the choice to either have a single node which gives a prediction of True or False or 2 output nodes representing the probability of True and False both.

For the activation function on the models using the Relu activation function proved to be the most successful. The more commonly used for CNN models Softmax function did not perform on this dataset.

```
model = Sequential()
model.add(Rescaling(1./255)),
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(48, 48, 1)))
model.add(MaxPool2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPool2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(7))
```

Figure 4

Once the model architecture was complete the model was compiled. This is a straightforward process but it's important to use parameters that dictate the metric category and optimization method which should be used for training. These are the parameters the model uses to make

improvements between epochs so the model can improve over each subsequent run.

The higher performing model ran for 50 epochs which essentially runs data through the model 50 times with each epoch building on the prior one's weights so that continuous improvement of accuracy is achieved. There will be a tradeoff between the time it takes to run a certain number of epochs and the performance gained. There will also be a point where improvement is not made.

To determine if gains are being made the scores prefixed with "Val" should be used. This is how well the model is predicting against unseen data residing in the validation set. If val_loss is increasing or Val_accuracy is not increasing, then the model is not learning despite any performance gains that might be gained in the training accuracy metric. Based on this the epoch runs will need modified. Or, if performance is not desirable, the architecture will of the model will need modified. It is also possible the data will need additional prepping.

In the case of these two models the first one did run for 50 but the second model performed so poorly it was only allowed to run for 10. This was done for academic purposes just to demonstrate the lack of improvement being made and the dichotomy which could be had between two different architectures using the same data.

For comparing epoch results using a variable to hold the history dictionary that is returned by the fit function epochs is useful for plotting metrics, comparing different models, and evaluating performance over epochs. This was the method used in this research to create data visualizations in the results section.

Once the model is trained it is ready to be used to make predictions. With this being a research project, the desire was to run a large dataset against the model see if it could handle production loads. To do so, a testing dataset was created using the same batch method performed with the data used for model training. This data was then sent to the model via the "model.predict" function. The "model.predict" function does as its name implies, runs predictions using the previously compiled and trained model. The output of these predictions is a numpy array which has seven predictions in each row of the multi-dimensional array. These seven predictions represent the probability of the image being fed to the model being one of the seven emotion classes.

To narrow these predictions likelihoods down to one definitive prediction the numpy array was iterated through and the max value index of each row was returned. This index number correlated to an emotion with the index being that of the highest value in the row. Thus, the emotion the image most likely would represent. The emotions were mapped as follows.

Emotion Index Number	Emotion Name
0	Angry
1	Disgust
2	Fear
3	Happy
4	Neutral
5	Sad
6	Surprise

Figure 5

These predictions were saved to a csv file along side with the corresponding labels. The labels were created by iterating through all of the test data directories and creating an index entry based on what directory the algorithm was in at the time. While this worked a technical issue did occur

	Model 1	Model 2
Epochs	50	10
Average Epoch Run Time	64.3 Seconds	93.6 Seconds
Accuracy @ start	.9739	.2528
Accuracy @ End	.9791	.2528
Loss @ Start	0.0	1
Loss @ End	0.0	1
Val_Accuracy @ Start	.4862	.2453
Val_Accuracy @ End	.4835	.2453
Val_Loss @ Start	7.027	1.8
Val_Loss @ End	8.45	1.81

Figure 6

limiting the ability to test anything but load capacity. The issue was that the batch process used to feed the prediction function was randomly selecting images despite the seed parameter being removed from the batch creation code. This will need addressed in the future. Despite this, when feeding images manually the model performed consistently with the validation accuracy metrics collected during the model training phase.

The last activity on this project was the creation of metrics which documented the performance of the training over each epoch iteration. For this a history variable was used to get the log information returned from the model during training. This provides data in dictionary key value pairs. These keys for this particular Keras Sequence model were Loss, Accuracy, Val_loss, and Val_accuracy.

Results

As can be seen by the comparison table, Figure 6, the two models yielded very different results proving model architecture can have a significant impact on the performance. On validation scores, model 1 was nearly twice as accurate and with it's training score being 97 percent there is opportunity to analyze what caused the difference between training accuracy and validation accuracy. Understanding the source of that Gap would enable the two scores to have less of a spread making this a high-performance model.

The adjustment needed are evident by analyzing the graphs, figure 6 and 8. The most significant metric needing evaluation is the loss for model 1. The loss increases over the epochs at a steady rate. This will be the first metric to gauge model remediation efforts against in an attempt to improve performance. With the Training epochs being fairly smooth it indicates these were not overfit. A very jagged line would indicate overfitting as the model would try to weight the nodes heavily in favor of each type of image being processed rather than gradual adjustments towards improvement.

For Model 2 the performance started off poor and learning did not occur. However, the training and validation did yield the same results. Vetting into the reason for this performance was not conducted since Model 1 would be more beneficial to concentrate efforts on. Model 2 was left into the results for academic purposes to show other architectures were trialed. It was also left in to show the difference a Neural Network architecture can make on performance.

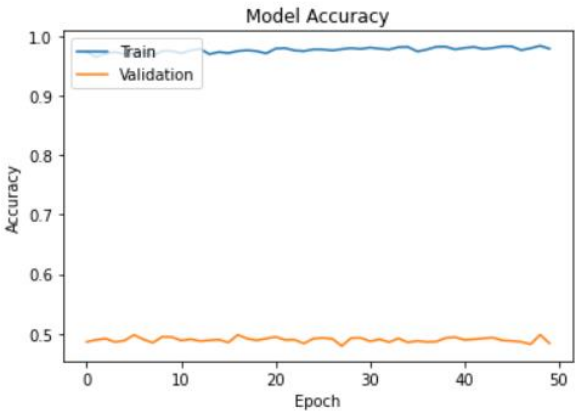

Figure 7

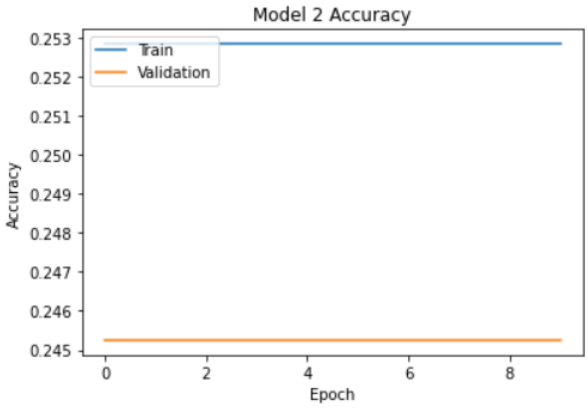

Figure 8

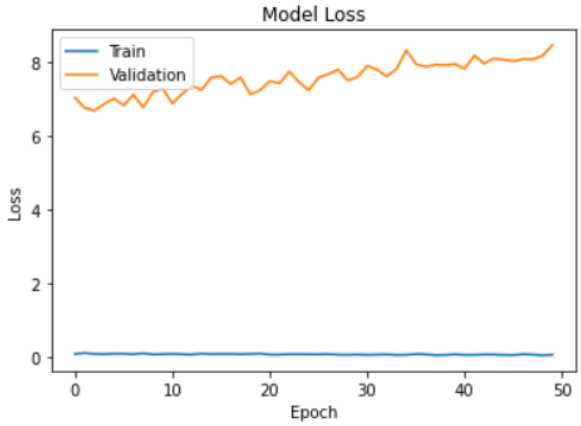

Figure 9

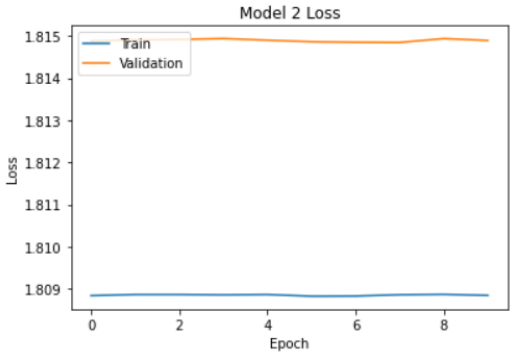

Figure 10

The code and output metrics for each epoch run can be seen by following the Link to the

Kaggle public notebook created for this project.
https://www.kaggle.com/cpmsyr/notebook54355cfd18

Conclusion

The project explored the fundamentals behind Convolutional Neural Networks, Facial Emotion Recognition (FER), Image processing, and Tensor Flow Libraries. It also enabled the first steps needed for any project involving FER, a working model. With this model fully functional and able to make predictions on large volumes of images in batches the next steps towards creating a unique use case can be pursued.

But first, the model 1would need improved to achieve validation and testing accuracy near the 97 percent training accuracy demonstrated. With more time allotted to fine tuning the model metrics indicate this is possible. Only then, the unique use case I would like to pursue could be developed. This unique use case is a system capable of capturing still shot images of video conference participants and running emotion detection on each individual with that output contributing to a group score consensus.

For this development, the next steps would be to add the functionality to perform facial image capturing from a video conferencing participant list, batch predict those images, and log the results. These images would be captured either at a configurable interval or at key moments in presentations so the overall emotion of the participants can be evaluated. These predictions, in coordination with the time stamps correlating presentation key points with the group predictions would allow presenters to understand the overall impact their content was having on the audience. This would allow for insights into adjustments needed to capture the emotions of the audience in the desired way for future iterations of the presentation. This could be applied to presentations, teaching, product pitches, political speeches, and much more.

References

[1]James Pao, Emotion Detection Through Facial Feature Recognition, Stanford University, https://web.stanford.edu/class/ee368/Project_Autumn_1617/Reports/report_pao.pdf

[2]Keiron O'Shea and Ryan Nash, An Introduction to Convolutional Neural Networks, Department of Computer Science, Aberystwyth University, Cerdigion, SY23DB
School of Computing and Communications, Lancaster University, Lancashire, LA14YW

[3] M. Soleymani, G. Chanel, J.J.M. Kierkels, and T. Pun, "Affective Characterization of Movie Scenes Based on Content Analysis and Physiological Changes," Int'l J. Semantic Computing, vol. 3, no. 2, pp. 235-254, June 2009.

[4] Ekman, P. & Keltner, D. (1997). Universal facial expressions of emotion: An old controversy and new findings. In Segerstråle, U. C. & Molnár, P. (Eds.), Nonverbal communication: Where nature meets culture (pp. 27-46). Mahwah, NJ: Lawrence Erlbaum Associates.

[5] IEEE TRANSACTIONS ON AFFECTIVE COMPUTING, VOL. 3, NO. 1, JANUARY-MARCH 2012
DEAP: A Database for Emotion Analysis Using Physiological Signals Sander Koelstra, Student Member, IEEE, Christian Mu¨hl, Mohammad Soleymani, Student Member, IEEE, Jong-Seok Lee, Member, IEEE, Ashkan Yazdani, Touradj Ebrahimi, Member, IEEE, Thierry Pun, Member, IEEE, Anton Nijholt, Member, IEEE, and Ioannis (Yiannis) Patras, Senior Member, IEEE

[6] R. Kosti, J.M. Álvarez, A. Recasens and A. Lapedriza, "Emotion Recognition in Context", Computer Vision and Pattern Recognition (CVPR), 2017

[7] R. Kosti, J.M. Álvarez, A. Recasens and A. Lapedriza, "Context based emotion recognition using emotic dataset", IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2019