Final Project: Facial Emotion Recognition with Support Vector Machine and Convolutional Neural Network

William Livengood, Garrett Rider, Marissa Lane

04/23/2024

Abstract:

This report explores emotion recognition from facial expressions using Support Vector Machine (SVM) and Convolutional Neural Network (CNN) models. Using the FER2013 dataset, images are preprocessed and divided into training and testing sets. An SVM classifier is trained on flattened image data to predict emotion labels, while a CNN model, built with TensorFlow's Keras API, extracts features directly from images. The CNN comprises convolutional layers with ReLU activation, followed by max-pooling and dense layers. Both models are evaluated for accuracy, revealing insights into their comparative performance for emotion recognition tasks. Our findings reveal that the CNN model outperforms the SVM, achieving higher accuracy despite challenges such as data imbalance and underfitting. While SVM demonstrates efficiency in small datasets, CNN excels in complex classification tasks with larger datasets, showcasing its capability in facial recognition and emotion detection. However, challenges such as incomplete facial visibility and non-face images within the dataset hindered our analysis, suggesting avenues for future research. Potential applications in the medical and data security fields highlight the importance of CNNs' capabilities in complex tasks, reaffirming their pivotal role in data science.

Introduction:

Emotion recognition from facial expressions is pivotal in various domains, including psychology and human-computer interaction. Our group, consisting of Marissa Lane, Garret Rider, and William Livengood, aims to explore this field using machine learning techniques, specifically Support Vector Machine (SVM) and Convolutional Neural Network (CNN) models. While initially focusing on SVM, we later incorporated CNN analysis to compare traditional and deep learning approaches in facial emotion recognition. Leveraging the FER2013 dataset, our study seeks to evaluate SVM's performance in recognizing emotions like sadness, happiness, and fear, and assess the effectiveness of SVM and CNN models in large-scale image classification tasks. Through this research, we aim to contribute insights into the capabilities of SVM and CNN models in emotional recognition, with implications for advancements in emotion-aware technologies.

Support Vector Machine (SVM) is a powerful algorithm for classification tasks, including facial emotion recognition as demonstrated by our project. An SVM algorithm is composed of the hyperplane and support vectors. A hyperplane is a flat subspace of one lower dimension than the space it exists in, and the support vectors are the data points closest to the hyperplane. At its core, SVM aims to find the optimal hyperplane that separates the classes in the feature space. In the context of our code, this hyperplane corresponds to the decision boundary that distinguishes between different facial expressions. The SVM model is trained using the FER2013 dataset, the images are preprocessed and flattened into feature vectors, and the corresponding emotion labels are used as class labels.

Mathematically, the goal of SVM is to minimize the classification error while maximizing the margin between the hyperplane and the support vectors. This is achieved by solving an optimization problem where the margin (ρ) is proportional to the norm of the weight vector (**w**).

$$\rho=(2/||\mathbf{w}||)$$

To maximize the margin, SVM solves the following optimization problem:

$$\text{minimize } ||w||^2$$

subject to the constraints: $y_i(w \cdot x_i + \text{b}) \geq 1$

where y_i is the class label of the ith data point and x_i is its feature vector. The optimization problem can be solved using techniques such as the method of Lagrange multipliers, resulting in the following dual form:

$$\text{maximize } \sum_{i=1} \alpha_i - (1/2) \sum_{i=1} \sum_{j=1} \alpha_i \alpha_j y_i y_j x_i \cdot x_j$$

subject to the constraints: $\sum_{i=1} \alpha_i y_i = 0$ and $0 \leq \alpha_i \leq \text{C}$

where α_i are the Lagrange multipliers and C is a regularization parameter that controls the trade-off between maximizing the margin and minimizing the classification error. If the data is not linearly separable, SVM can employ kernel functions to map the input data into higher-dimensional feature spaces, allowing for non-linear decision boundaries.

Overall, the SVM model in our code follows the fundamental principles of SVM by finding the optimal hyperplane to classify facial expressions, with adjustments made for the specific dataset and problem domain of facial emotion recognition.

Convolutional Neural Network (CNN) models are particularly well-suited for image recognition tasks due to their ability to automatically learn spatial hierarchies of features. In our case, we're using TensorFlow's Keras API to construct the CNN architecture, which consists of convolutional layers followed by max-pooling and dense layers.

Mathematically, CNNs perform convolutions between input images (X) and learnable filters (W), capturing local patterns and features. The output feature map (Y) of a convolutional layer is calculated as:

$$Y[i, j] = \sum_m \sum_n X[m, n] \times W[i - m, j - n]$$

where $Y[i, j]$ denotes the value at position (i, j) in the output feature map, and m,n iterate over the spatial dimensions of the input image. The output of the convolutional layer is then passed through a non-linear activation function, such as the Rectified Linear Unit (ReLU), which introduces non-linearity to the network:

$$ReLU(x) = max(0, x)$$

Max-pooling layers are often used to downsample the feature maps, reducing computation and introducing spatial invariance. This operation is typically performed by taking the maximum value within a localized region of the feature map.

During training, CNNs optimize their parameters (e.g., filter weights (W)) using techniques like gradient descent and backpropagation. The loss function, such as categorical cross-entropy, is minimized to adjust the filter weights in a way that improves the network's performance. Through this iterative process, CNNs learn to extract relevant features from the input images and make accurate predictions.

The addition of CNN analysis to our code enriches our investigation, allowing us to explore deep learning approaches alongside the traditional machine learning method SVM, ultimately enhancing our understanding of how well these methods are applied to facial emotion recognition.

Data Processing:

Recall that the focus of our report is to explore a machine learning project focused on emotion recognition from facial expressions using the FER2013 dataset. The FER2013 dataset came in seven subdirectories divided by emotion label, and in .jpg file format. We had to do some data processing before to get this data into a usable format, as well as setting up lists to store the data and the labels. Here is a breakdown of the steps we took during processing:

- **Importing Libraries:** Our code imports the necessary libraries: NumPy, Matplotlib, os, and random. We also imported the following modules for data processing:, imread from Matplotlib, and enum from Enum.
- **Copy to combined directory:** For this, we wrote code to randomly select a certain number of images from each class, and copy it to a new directory. To preserve the data labels, we rename the images to include the label.
- **Convert to PNG:** We also converted each .jpg image to .png format so that the Matplotlib function could properly process the data.
- **Defining Emotion Enum:** We chose to define an enumeration for the seven emotion labels in the dataset.The emotion enum keeps track of each image's label with an integer value representing a different emotion.
- **Loading Images and Labels:** Then our code creates lists for the images and the labels. It iterates through the images, reads them using imread, flattens it, and adds them to the images list and their label to the labels list.
- **Visualization:** To confirm the labels were correctly assigned to the images, we randomly select 25 images and plot them along with their emotion labels.

SVM Model:

Here is a breakdown of our code for the SVM model:

- **Importing Libraries:** Our SVM model required importing the following modules from scikit-learn: svm, train_test_split, and accuracy_score.
- **Partitioning Data:** We define a partition function to split the data and labels into training and testing sets using train_test_split from sklearn.model_selection. For our purposes we used a 80-20 training-testing split.
- **Training and Prediction:** We define a prediction function to train a Support Vector Classifier (SVC) on the training data, predict emotion labels for the testing data, and score the model's accuracy. Our SVM model uses the following hyperparameters: C = 100000, kernel = 'rbf', and gamma = 'scale'. We found the 'rbf' kernel performs much better than other common kernels for this task.

In short, the data processing and SVM code loads image data, preprocesses it, splits it into training and testing sets, trains a Support Vector Classifier model on the dataset, evaluates its performance, and provides visualizations of the data and predictions.

CNN Model:

The second aspect of our machine learning project was related to setting up and training a Convolutional Neural Network (CNN) model using TensorFlow's Keras API for facial emotion recognition. Here's a breakdown of the key steps performed in this code:

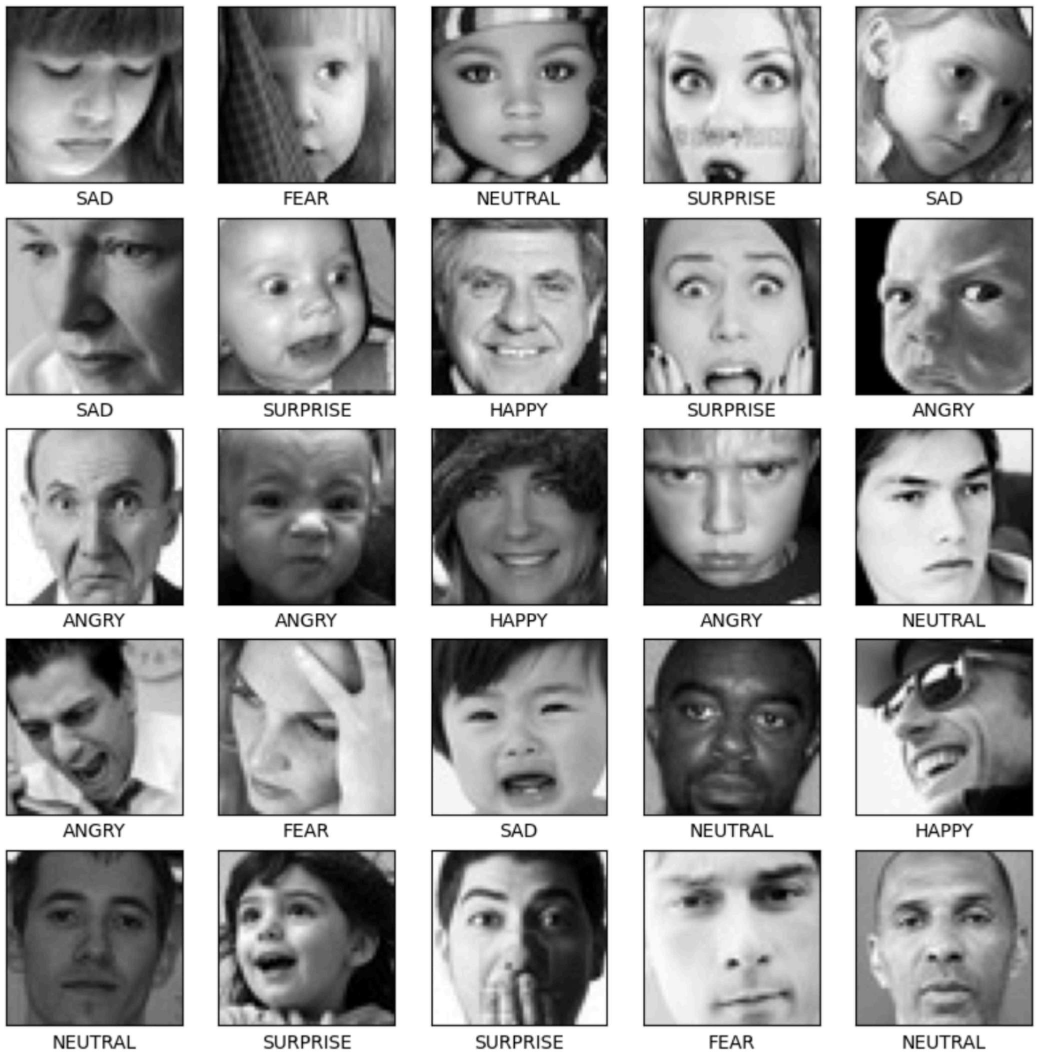
- **Partitioning Data:** In addition to the data processing stage from earlier, some additional processing was needed for the CNN model. The CNN model also used an 80-20 training-testing split. The data returned is converted from a list to a NumPy array and reshaped to match the input shape expected by the CNN model. During this, each image is expanded to include a single channel dimension representing grayscale.

- **Model Definition:** Our CNN model was defined with a convolutional layer defining the input size, followed by a max-pooling layer. These layers are followed by two groups each with one convolutional, max-pooling, batch normalization, and dropout layer in that order. Our convolutional layers have 64 filters of size 3x3, and use the ReLU activation function. The model then has a flattening layer, followed by a dense (fully-connected) layer with ReLU activation, a dropout layer, and an output layer. The final layer is also a dense layer with 7 units (corresponding to the number of emotion classes).
- **Model Compilation:** The model is compiled using the Adam optimizer, a popular optimization algorithm for training neural networks, and Sparse Categorical Crossentropy loss function. The Sparse Categorical Crossentropy loss function is used in multi-class classification problems where the target labels are integers.
- **Model Training:** We call the model.fit() function on the defined and compiled model to train the CNN model on the training data (train_data) and labels (train_target). The model.fit() function also validates the model at the end of each epoch, of which we run for at least 50.
- **Model Evaluation:** To evaluate the performance of the CNN, we look at the last few validation accuracies, we also extract the accuracies of the model by class to see how well it classifies the different facial emotions. Finally, we plot the loss value and accuracy for the training portion of each epoch against the loss for the testing portion. These plots show us how the model progressed, and if it was suffering from overfitting or underfitting.

Our code for this part partitions the data set for testing and training, restructures the data for a convolutional neural network, and defines the CNN model for facial emotion recognition. Then the model is trained using the training data and evaluated using the testing data to assess its performance in classifying facial expressions. Finally, the results of the training and validation are plotted to visualize the performance of the model.

Results:

Example Observations: Below are 25 randomly selected observations from the FER2013 dataset with their corresponding labels.



There are 28,583 images in the subset of FER2013 we used, giving us a very large dataset to train our SVM and CNN models on.

Performance of Final CNN epochs:

50 Epochs				
Epoch	*Train Accuracy*	*Loss*	*Test Accuracy*	*Test Loss*
46	0.540584	1.214495	0.564632	1.159488
47	0.538354	1.21647	0.549414	1.205700
48	0.537129	1.209591	0.540843	1.207993
49	0.534637	1.210482	0.538569	1.215248
50	0.541986	1.207186	0.542893	1.186332
200 Epochs				
Epoch	*Train Accuracy*	*Loss*	*Test Accuracy*	*Test Loss*
196	0.582437	1.094356	0.591044	1.101316
197	0.583574	1.094309	0.592269	1.111496
198	0.580644	1.099591	0.588945	1.098545
199	0.582174	1.095435	0.577751	1.123541
200	0.580250	1.096124	0.582298	1.110850

Above is the final performance of our CNN model, from a 50 epoch run and a 200 epoch run.

SVM Model Accuracy by Emotion:

Emotion	**Accuracy**
Surprise	58%

Sad	36%
Neutral	41%
Happy	61%
Fear	33%
Disgust	45%
Angry	32%

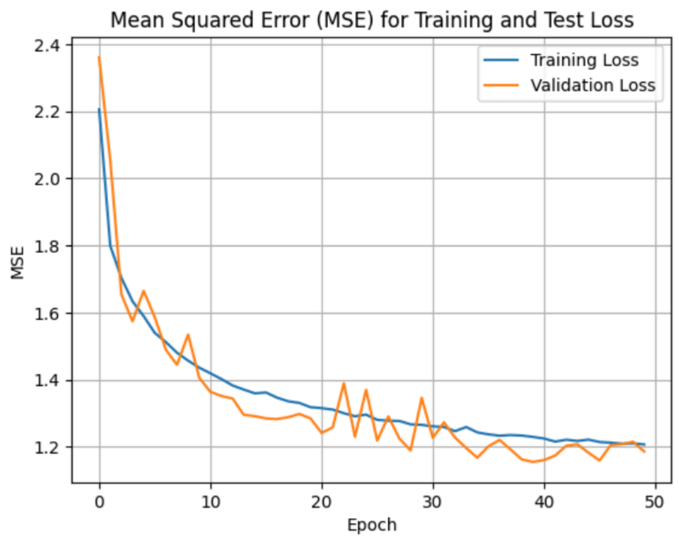
The table above displays the breakdown of the SVM's accuracy when classifying examples of the seven different emotions. The overall accuracy of the SVM model was 44.8%.

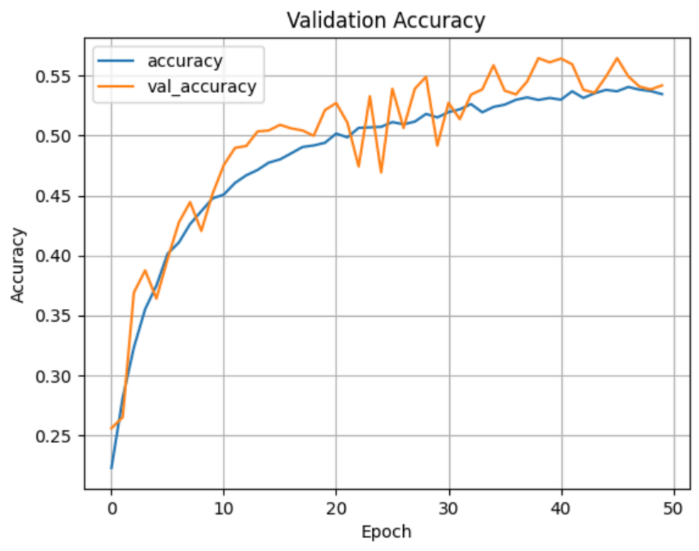
CNN Model Accuracy by Emotion After 50 Epochs:

Emotion	Accuracy
Surprise	76%
Sad	40%
Neutral	60%
Happy	82%
Fear	25%
Disgust	11%
Angry	33%

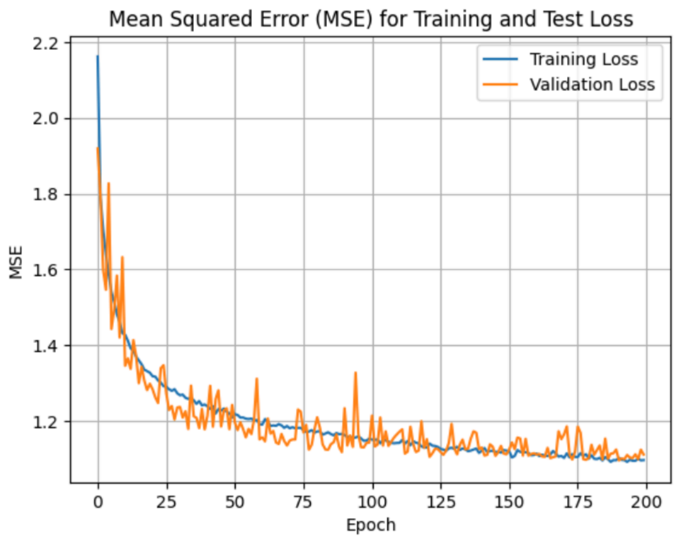
The table above displays the breakdown of the CNN's accuracy when classifying examples of the seven different emotions. The overall accuracy of the CNN model is shown in the first table.

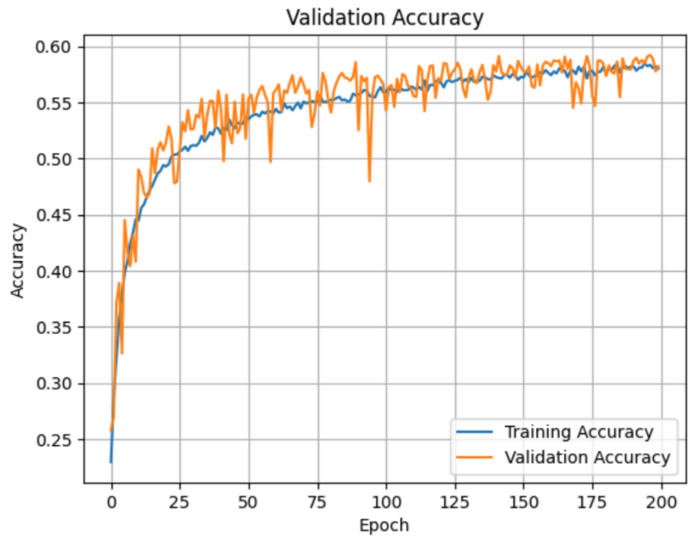
50 Epochs:





200 Epochs:





Discussion:

The results of our study demonstrate that the CNN model significantly outperformed the SVM for the purpose of classifying faces by emotion. The SVM model got an overall accuracy of 44.8% when it was given an 80% training, 20% testing split on 28,583 images from the FER2013 dataset. When trained on the same dataset with the same training-testing split, the CNN model was able to get better results. We ran the model for 50 epochs with a batch size of 100, and our last epoch had a validation accuracy of 55.9%. The CNN model also trained in under half the time of the SVM, indicating some greater efficiency.

We also found there were significant differences in the accuracy of both models on the different emotions. This is consistent with the data imbalance in the FER2013 dataset between the seven emotions; certain emotions, such as happiness and neutral are overrepresented, and others, like disgust and surprise are significantly underrepresented in comparison. The CNN model especially suffered from lack of data on the disgust class, as it provided about 12% as many disgust observations as happiness. Despite this, the emotions which were more easily classified by the SVM model were generally also more easily classified by the CNN model. This implies that some emotions (angry, fear, sadness, and disgust) are generally harder to learn. These negative emotions are more complex and nuanced, as well as being easier to confuse with each other.

The SVM model faces limitations when classifying the FER2013 dataset as these models work best when the data is linearly separable, and the FER2013 dataset is a multiclass problem. When provided with a small sample of the data set, the SVM was able to achieve about 33% accuracy, though it works better when given more data for this dataset (closer to 45%), it also performs much slower when given the entire dataset.

Our initial CNN models suffered from underfitting, since we were providing a much smaller sample of the dataset. Only once it was provided much more of FER2013 was the CNN able to compete with the SVM model. Our next problem became overfitting, the validation accuracy could not keep up with the training accuracy, which led to it peaking at about 55% accuracy, well below the training accuracy. However, when we added regularization and dropout layers to prevent overfitting, the models did not perform better, still peaking at 55% accuracy after 50 epochs, and only getting up to 59% after 200 epochs. As is shown in the Validation Accuracy and Mean Squared Error plots, the model does not appear to be overfitted or under fitted, but is performing about the same. The accuracy could probably be improved by constructing a model with deeper layers, but we were limited by our computational resources and time.

Our findings suggest that CNN models are better fitted for complex classification tasks, especially with large data sets, though in small datasets the SVM model does outperform the CNN in speed and accuracy. Also, we found a deeper architecture which would require much greater computational resources is required to increase the accuracy of the CNN model above 60%.

Conclusion:

We conclude that for the purposes of facial recognition and emotion detection, a Convolutional Neural Network is the recommended method of analysis, since it is capable of predicting emotion with higher accuracy than a simpler classification technique such as SVM, which relies on hyperplanes. The relative success this network had in “learning” facial recognition is significant because this task is not simple, faces have many small details and distinguishing between emotions can sometimes be hard for humans, let alone machines. Some limitations that we had with this dataset that may have held back our analysis are that there were images in the dataset where the face was not fully visible; the subject would either be in side profile or have their face not visible at all. This is not so easy to classify, since the algorithm will struggle to determine what it is supposed to be detecting. Additionally, there were images that did not even contain faces, adding random noise to the data that would bring accuracy down. Future research could attempt to complicate the model further in hopes of getting a prediction accuracy closer to ninety percent. Another study could focus on trying the same model on real-world data or a different dataset, to determine if the inefficiencies in our model were due to the data we used or the Model itself. We could see future implementation of these methods in the medical field as a diagnostic tool, or in fields such as data security, providing facial recognition software for privacy and account security. Neural networks provide pivotal roles in data science, and this study provides proof that it is capable of outperforming more discrete linear classification modeling for a complex task.

References

Bishop, Christopher M (2006). "Pattern Recognition and Machine Learning." [ISBN: 978-0387310732]. Springer.

Goodfellow, Ian, et al.(2015). "Challenges in Representation Learning: A report on three machine learning contests." Neural Networks, vol. 64, pp. 59-63. [DOI: 10.1016/j.neunet.2014.09.005]

Goodfellow, I., Bengio, Y., & Courville, A. (2016)." Deep Learning" (Vol. 1). MIT press.

Nair, V., & Hinton, G. E. (2010). "Rectified linear units improve restricted boltzmann machines." In Proceedings of the 27th International Conference on International Conference on Machine Learning (ICML'10) (pp. 807-814). Omnipress.

(2019, July 26). "Converting all files (.jpg) to (.png) from a directory in Python": Stack Overflow. Retrieved from: **https://stackoverflow.com/questions/57217941/converting-all-files-jpg-to-png-from-a-directory-in-python**

(2018, March 14). "Pulling random files out of a folder for sampling": Stack Overflow. Retrieved from: **https://stackoverflow.com/questions/49280966/pulling-random-files-out-of-a-folder-for-sampling**

(2020, May 9). "Sklearn plotting classification report gives a different output than basic avg": Stack Overflow. Retrieved from: **https://stackoverflow.com/questions/61705257/sklearn-plotting-classification-report-gives-a-different-output-than-basic-avg**