Mini Project 4 - COMP 551 W2022

Anonymous Author(s)
Affiliation
Address
email

1 Abstract

In this project, we attempt to reproduce the results of a scientific paper Deep Learning using Linear Support Vector Machines that uses a mix of Convolutional Neural Networks and Support Vector Machines to classify facial expressions of people[1]. The model is trained on 28,709 48x48 grayscale pixel images where there are 7 categories of emotion. Since the research paper Deep Learning using Linear Support Vector Machines didn't have its own implementation, we used community code. We experiment by using a CNN with softmax cross-entropy loss and compare that to the ensemble model. Our main goal is to observe the difficulties of reproducing results from a paper and to examine the ways that we can improve on their findings. Our ensemble model does not perform as well as the one described in Deep Learning using Linear Support Vector Machines but the CNN with cross-entropy loss outperforms their model by about 11% accuracy.

2 Introduction

A major application of Machine Learning is computer vision. For the research paper Deep Learning using Linear Support Vector Machines that we are reproducing, the paper attempts to classify images of people based on their expression. The model Deep Learning using Linear Support Vector Machines uses is a combination of a CNN and SVM where the output layer of the CNN uses a Linear SVM activation function instead of using a softmax function. Deep Learning using Linear Support Vector Machines claims that this alteration of the CNN architecture improves the accuracy compared to a regular CNN[1]. In this project we use community code to recreate the methods mentioned in Deep Learning using Linear Support Vector Machines as well as experiment to see if we can improve on their results using a deep CNN. Each perceptron in a Neural Network is only optimized when it outputs an incorrect value, so the line that separates two classes may not be optimal. Whereas for an SVM, the line that separates classes is one which maximizes the distance between the line and the two classes, this tends to give more accurate results on unseen data[3]. Deep Learning using Linear Support Vector Machines shows that combining CNN and SVM gives a better result than either model alone. The model is trained on 28, 709 48x48 grayscale images where each image corresponds to an expression category. To prepare the data, we simply split the input CSV file into an array of images and an array of their corresponding emotion. The data is then passed to a CNN which has an SVM function as the output layer's activation function. Deep Learning using Linear Support Vector Machines states that the human accuracy on these images is estimated to be between 65% and 68% due to noise and other factors[1]. The model in Deep Learning using Linear Support Vector Machines had an accuracy of 69.4% on their test set. Our SVM and CNN model gave us an accuracy of 25.58% which underperformed the model that we tried to recreate. Although a CNN using softmax in the output layer and cross-entropy loss gets an accuracy of 81.85% which outperforms the model described in Deep Learning using Linear Support Vector Machines.

3 Datasets

The dataset we use is from a Kaggle competition hosted in May 2013. The dataset was prepared by Pierre-Luc Carrier and Aaron Courville, as part of an ongoing research project[2]. The dataset is partitioned into 7 different categories, labeled 0 to 6 correspond to Angry, Disgust, Fear, Happy, Sad, Surprise and Neutral, respectively. There are 28,709 48x48 grayscale images which are used to train the model, the number of instances in each of the different categories

Submitted to ML Reproducibility Challenge 2020. Do not distribute.

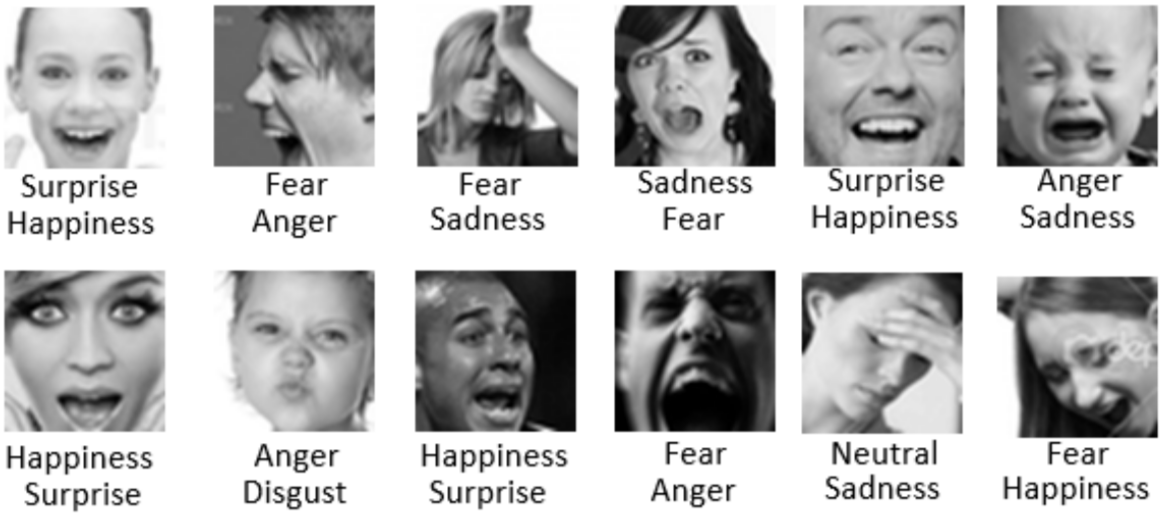

Figure 1: Example images from the dataset with corresponding labels

are Angry:4953 , Disgust:547, Fear:5121, Happy:8989, Sad:6077, Surprise:4002 and Neutral:6198. We can see that the expressions happy, neutral and sad are the most common labels whereas disgust, surprise and anger are the least common.

The dataset is uploaded to our drive and then we load it into our notebook. The CSV has two columns, the first one being the labels and the second column is the images. Once we separate the labels from the images, we are ready to begin training. No preprocessing is done on the images.

4 Results

For testing, we conducted three experiments by modifying the baseline code. We tested a CNN model with Categorical Cross Entropy and Softmax. We test another CNN model with Categorical Hinge and a SVM. And we also tested with just a SVM model using Sklearn's LinearSVC model. The CNN-Softmax model ran for 100 epochs and the CNN-SVM model ran for 40 epochs, based on the Adam optimizer with learning rate 0.0001. For training and testing, we used a 90:10 split.

As seen in Table 1, the results across our three experiments were quite contrasting. The CNN + Softmax model performed very well. In fact, it outperformed the Softmax model mentioned in the original paper. The training accuracy hit 96.7% and the testing accuracy hit 81.9%, whereas the paper's model achieved a max accuracy of 70.1%. As seen in Table 2, this model achieved the highest precision, recall, and F1 scores for the "Disgust" category. It also had the lowest precision for "Fear" and lowest recall and F1 score for "Sad". The model appears to have learned the more drastic emotions slightly better such as "Disgust", "Surprise", and "Angry", which may be due to the fact that faces portraying these emotions contain more easily recognizable characteristics.

For the CNN + SVM model, the accuracies were lower than expected, with a training accuracy of 18.5% and a testing accuracy of 25.6%. These numbers are much lower than the ones mentioned in the paper, which had a max accuracy of 71.2%. Another interesting result is that the testing accuracy is higher than the training accuracy with the CNN+SVM model, suggesting that the model is underfitting. The underfitting is further seen by the results of Table 2 for this model. The highest precision occurred for "Neutral". The highest recall occurred for "Angry". And the highest F1 score occurred for "Surprise". The lowest scores were hard to interpret due to the numerous amounts of 0 entries. Again, this is a sign that the model was not fitted properly.

Even though we tested numerous times on multiple machines, the SVM model was unable to converge. Training on the entire dataset took a very long time, and resulted in convergence errors every time. This is most likely due to the dataset size as SVMs are often a poor choice for datasets which are too large. Additionally, SVMs perform best when the data

	Train Accuracy	Test Accuracy
CNN + Categorical Cross Entropy	0.9853	**0.8355**
CNN + Categorical Hinge	0.1850	**0.2558**
SVM	Failed to Converge	**Failed to Converge**

Table 1: Test Results of Three Different Models

	CNN + Categorical Cross Entropy			CNN + Categorical Hinge		
Class	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Angry (0)	0.79	0.83	0.81	0.17	**0.98**	0.28
Disgust (1)	**0.99**	**1.00**	**0.99**	0.00	0.00	0.00
Fear (2)	*0.67*	0.85	0.75	0.12	0.00	0.00
Happiness (3)	0.82	0.77	0.79	0.00	0.00	0.00
Sad (4)	0.78	*0.66*	*0.72*	0.00	0.00	0.00
Surprise (5)	0.86	0.95	0.90	0.84	0.79	**0.81**
Neutral (6)	0.85	0.68	0.75	**1.00**	0.00	0.00

Table 2: Classification Report of the Two CNN-based Models

is linearly separable. However, with image data, data is often not linearly separable. Thus training just a SVM model is not a good fit for this type of experiment.

To observe and understand the underfitting, we plotted the accuracy vs the epochs that can be seen in figure 2.

As can be seen in Figure 2, the CNN+Softmax model is fitted very well. In fact the accuracies observed here not only replicate the original paper's results but also go above and beyond the original paper's accuracies. Moreover, it can be seen that the majority of the training is completed after around 20 epochs. Beyond 20 epochs, the improvements to the training accuracy does not yield significant improvements to the testing accuracy. After epoch 30, we can see that the accuracies plateau. Additionally, we should note that the training accuracy starts lower than the testing accuracy but eventually crosses the testing accuracy line around epoch 6-7.

This is not the case for the CNN+SVM model. We can see that the training accuracy after 40 epochs is still under the testing accuracy line. However, we see that around epoch 30, the slope of the training accuracy line begins to grow faster than the testing line. We also see that the accuracies for both models are still steadily improving at epoch 40. Thus, we believe that if we train the CNN+SVM for a much greater number of epochs, we will achieve similar results to the CNN+Softmax model.

Although we tried various numbers for training epochs, we were unable to optimize the number of epochs for the CNN+SVM model. In the original paper, the hyperparameter for the number of epochs was not given. We were unable to try very large values (1000+) for the number of epochs because the training time was simply infeasible. Whereas the paper used C++ and CUDA, our experiments were conducted on Google Colab using Python. With these limited resources, we ran into difficulties when training for a high number of epochs.

Figure 2: Training and Testing Accuracies of CNN-based Models Over 40 Epochs

5 Hyperparameter Tuning and Ablation Studies

For this section, we spent most of our efforts on ablation studies and how the model choice and architecture impacts the model's accuracies for our dataset. The main hyperparameter we experimented with was the number of epochs for the CNN models. For the number of epochs, we tested everything within the range 1 to 100 for the CNN+Softmax and 1 to 40 for the CNN+SVM model.

We were able to test 100 epochs for the CNN+Softmax model because training was more efficient for this model, whereas each epoch for the CNN+SVM took longer. However, for the CNN+Softmax model, we did not see much improvement after epoch 40. From epoch 40 to 100, there was only about a 2

For the CNN+SVM model, we observed consistent improvements in accuracies after each epoch of training. If we had access to much more powerful computers, we would like to increase the number of epochs for this model to 1000+ to observe the training accuracy. Based on the results of the original paper, we hypothesize that this model can perform just as well if not outperform our CNN+Softmax model if trained for a long time.

For learning rate and activation functions for each layer, we drew inspiration from the original paper as well as the code implementation and chose a learning rate of 0.0001 and activation functions of ReLU for each layer. We found that these hyperparameters worked best in our experiments as well.

The initial thing we tested for ablation studies were the three models. We wondered the exact effects of varying the model. Because the model mentioned in the paper is an ensemble model composed of a CNN and SVM, we wanted to examine the accuracies, of just the CNN with Softmax, CNN+SVM, and just the SVM. Here we realized the computational overhead of using a SVM. Whereas we were able to train a CNN with ease and achieve results even better than the ones mentioned in the paper, we quickly realized that the CNN+SVM model would have to be

run for quite a long time in order to match the accuracies of the CNN model. In a professional research setting with access to much more powerful computing power, the CNN+SVM model could provide a slight edge over the standard SVM model as seen in the paper, but we realized that the standard CNN model would serve as a great middle-ground between performance and cost. This effect was only exaggerated when we attempted to train a purely SVM model. As mentioned previously, even though a SVM is a less complex model than a Neural Network, we were unable to train the SVM within a reasonable amount of time. Our experiments with this model led to the model not converging properly, which could also be an issue inherent to the fact that we are using image data which may not be easily linearly separable.

6 Discussion & Conclusion

From our experiments and the results achieved on the paper, the main takeaway is that computer vision and image classification can be done rather accurately if the correct ML models and techniques are used. As seen above we have been able to accurately replicate the results from the paper from the CNN model, however, there is some discrepancy for the SVM model.

First, As seen in the results, our SVM model out preforms the model on the Deep Learning using Linear Support Vector Machines. Given the fact that the paper was written about 8 years ago and the dataset has not changed, the diffrance in the in the accuracies achived might be assosiated to increase in the conputation power since. Moreover, in our expirements we reach a very low accuracy of 18.5% on the CNN + SVM model which is not remotly comparible to the results achived on the paper. It is not clear what the reason behind this diffrance is, however as mentioned in the number of epochs used.

7 Statement of Contributions

Sung Jun Lee - Wrote Results, and Hyperparameter Testing and Ablation Studies, Contributed to CNN code. Joseph Boehm - Wrote Abstract, Introduction, Datasets and References Namdar Kabolinejad - Setup code baseline - Wrote Discussion Conclusion - Converted to LaTeX

References

[1]Tang, Y. (2015). Deep Learning using Linear Support Vector Machines. doi:10.48550/ARXIV.1306.0239

[2] Challenges in representation learning: Facial expression recognition challenge. Kaggle. (n.d.). Retrieved April 26, 2022, from https://www.kaggle.com/competitions/challenges-in-representation-learning-facial-expression-recognition-challenge/data

[3] Rabbany, R. (2022). Support Vector Machines. http://www.reirab.com.

[4] Khaled, O. (2022, April 14). Facial expressions. Kaggle. Retrieved April 26, 2022, from https://www.kaggle.com/code/oknashar/facial-expressions