Lilyanne Kurth-Yontz (A13325385)
Jingyi tay (A99014740)
Huong Vu (A12869038)
Syed Zain Ali Baquar (A12732391)

Investigating Models for Emotion Classification

Abstract

In this paper we investigate how four pretrained models make use of machine learning techniques to classify emotions. The models we work with employ both supervised and unsupervised including convolutional neural networks (CNN/AlexNet), recurrent neural network (RNN), support vector machine (SVM), and AlexNet-SVM. We first discuss how each model preprocesses the data and analyze how accurately the model in question can predict what emotion is being displayed in an image. Investigating these models and evaluating how certain parameters affect the accuracy allows us to understand how effective they are in emotion classification. After we analyze the four models, we evaluate the tradeoffs and limitations of each to determine which is the most accurate in classifying emotions from an image dataset.

Introduction

Facial expression detection is a technology that has the potential to revolutionize the way we interface with digital media and devices. While one avenue of this concept involved detecting and recognising people, an alternative motive for this technology is to be able to understand people better. One way in which this has been done is through training algorithms to be able to recognize emotions by observed visual expressions.

Dataset

The JAFFE (Japanese Female Facial Expression) dataset is one that consists of two hundred and thirteen 256x256 grayscale TIFF format images of 6 facial expressions by 10 japanese women. These images were rated by 60 other japanese subjects and were given mean scores for each of emotions: Happy, Sad, Surprise, Anger, Disgust and Fear.

We chose this dataset for three main reasons. The first reason we chose this dataset is because it is relatively small and we wanted to work with a dataset that did not take too much time to train. Since we will be training it on 4 different models it is beneficial to choose a dataset that is smaller. We also chose JAFFE because it contains labels(scores) so that we can streamline the classification of emotions in the training and preprocessing phases. JAFFE is a popular dataset for emotion classification and we wanted to choose a dataset with a lot of documentation and literature.

Methods

KNN

The first model we tried out was the simplest, and since the dataset is relatively small, a K-Nearest Neighbors classifier was an affordable option.

Preprocess

After loading in the images, the labels were stored in a table in a text document accompanying the dataset. This table was extracted, matched with their respective images (according to a provided base ID) and put into a dataframe for analysis. For this model, the outputs required had to be a single class and so the emotion with the highest mean score was selected as the ground truth for each image. This data frame was then shuffled split into training and test sets to be used in the classifier.

Analysis

The classifier was trained with as many K values as there are observations in the training set. Thus the values of K ranged from 1 to the size of the training set. After the model was fit for its respective value of K, predictions were generated and the accuracy score for that K value was recorded.

SVM

Since there are many possible parameter combinations when used a Support Vector Machine (SVM), we attempted to perform these predictions using one model simplistic model with a linear kernel, and the second with a polynomial kernel. Other kernels were used as well, but were not fine-tuned to the same degree as linear and polynomial.

Preprocess

The preprocess for the SVM models was done in a similar fashion as done for the KNN method. The images were loaded in TIFF format and the labels were extracted such that the emotion used for analysis was the emotion with the highest mean score. This data was then stored in a data frame.

Analysis

a. Linear Kernel

The first SVM kernel to be attempted was linear. While this was not expected to perform the best, it serves as a baseline from which to compare our other models.

b. Polynomial Kernel

The second SVM model used a polynomial kernel. This kernel uses a unique parameter labeled as ‘degree’ in scikit-learn, which we set to 2 (default is 3). We used grid search and

3-fold validation to find the optimal 'C' (regularization parameter), which we set to be within a range of 1 and 3. Also, we set the 'gamma' parameter to 'scale' instead of the default 'auto'.

AlexNet-SVM

The fourth model we tried out employs two of our previous models: AlexNet and SVM. We hypothesize that combining models will improve on accuracy than if we were to just use AlexNet or SVM alone because there are more features to work with. The model we are examining is coded in MATLAB and is linked below [1].

Preprocess

The JAFFE dataset was downloaded and put it into a folder called 'Data.' We then used the DataPreprocessing.m script to convert the database to 3 channel, 227x227 images. The script reads the entire dataset and extracted the processed images to a separate folder. The reason the images need to be resized in this manner is due to the input that AlexNet takes (227x227x3 images). So before we can run the algorithm we need to ensure that the JAFFE images are compatible with the model.

Analysis

This pretrained model made use of two methods: transfer learning and feature extraction using AlexNet.

Transfer Learning:

For the first method the dataset was split 30-70 into a training and validation set. From there AlexNet was loaded from the matlab deep learning toolbox all but 3 of the layers (last 3) were extracted. During the training process certain parameters were kept track of so that we could see what progress the training was making. At first we were concerned because the training was taking longer than expected but when we revisited the code we realized if it was running properly then it should update in this table.

Epoch	Iteration	Time Elapsed (hh:mm:ss)	Mini-batch Accuracy	Mini-batch Loss	Base Learning Rate

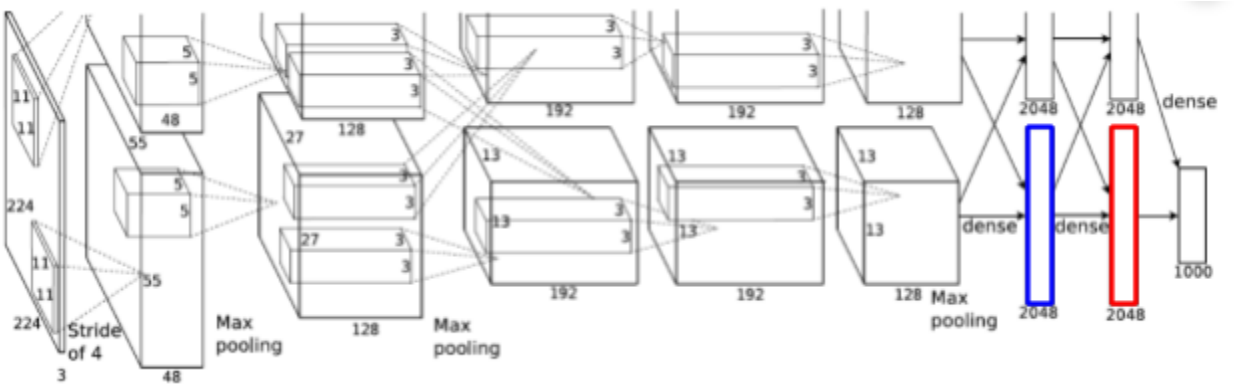
This helped us to see if the model was training the data properly because we see the Epochs updating in real time. In this model, they fine tuned the classes to classify on the 5 emotions and retrained certain layers of AlexNet to be consistent with the number of classes. The accuracy achieved using this model was .90. We wanted to see how the classification would change if at all using 6 classes (including neutral). When I modified the parameters to include 6 classes as opposed to 5 I noticed that the accuracy decreased slightly ~ 1 - 2%. This decrease

may be due to the fact that if you have more classes to classify on it will introduce more error and diminish some accuracy.

Feature Extraction:

For the second method AlexNet was used but they extracted layers from 'Fc7' and passed in the training labels obtained from these layers into an SVM. The SVM was also obtained from the matlab learning toolbox. The accuracy achieved using this model was slightly higher at .93. We hypothesize that a combination of SVM and AlexNet is more accurate because it adds an extra layer that makes classification more accurate than using AlexNet alone. To test this theory we modified some parameters in the SVM-AlexNet model to see how it would behave if a different layer was used. We replaced 'Fc7' (red) with 'Fc6' (blue) to see how the results would vary. The reason we decided to use 'Fc6' was because it has the same number of neurons as 'Fc7' (4096) and they are both fully connected layers in AlexNet.

AlexNet Schematic:



We noticed that the accuracy slightly decreased to 0.92. We think this is because of the layer that the features are extracted from. Extracting features from 'Fc7' yields a higher accuracy because it is a later layer in AlexNet.

Modified CNN in the model of AlexNet using Keras

Rationale

Given that we already know that the pre-trained AlexNet with additional fully connected layers works very well at classify JAFFE images with a 93% accuracy, we want to find a convolutional network that could have a better and more general performance than AlexNet (8 layers) but not as many layers as ResNet (150 layers).

According to Prudhvi Raj Dachapally (2017) of Indiana University, He created a 8-layered CNN with three convolutional layers, three pooling layers, and two fully connected

layers from scratch, with 10 filters at each convolutional layer; his CNN which is similar to AlexNet in terms of architecture was trained similarly to AlexNet but only on JAFFE, which has 215 images of 10 female models posing 7 emotions. Just like AlexNet, he augments his training data by extracting all 48*48 patches of images from his original set of images so that each image was condensed from 64*64 into 16-48*48 sized images. This was done to prevent overfitting when training. Using Keras, he trains the model 20 epochs at a time, with best validation accuracy after 360 iterations and with that model he tested its robustness on colored and varied poses of faces from the Labeled Faces in the Wild (LFW) dataset, which results in an accuracy of 67.62%.

Also, Scene classification with improved AlexNet model (2017), proved that a slightly lengthened and modified AlexNet would improve the accuracy of the original AlexNet by a few percentage points. The increased computational costs due to the depth of the network is compensated by the asymmetric convolutional kernels. The added convolutional kernels at the beginning with size 3 by 3 also helps to reduce the computational cost due to a larger spatial filter while the fully connected layers remain the same. It would be interesting to know how such an improved model would compare when training images of small datasets under a circumstance similar to what Prudhvi Raj Dachapally (2017) does.

Implementation

In order to create the layers of the improved AlexNet model in an attempt to do a comparison using JAFFE, we used Keras library to do so and followed several examples online and documentation to make sure the parameters are adjusted properly. However, after our completion of the model, we were only able to attain an accuracy of about 20%. This might be due to several factors. First, it might be due to human error . Also the lack of images might serve as a detriment to the accuracy as many models either were trained on duplicated images or have more fully connected layers. Lastly, if the problem is indeed the size of the dataset, then the model we created was not trained on enough images for the convolutional layers to provide appropriate weights for the feature vectors to be processed by the fully connected layers.

Results

KNN

The best accuracy of 0.3968 was achieved by the classifier that used a K value of 15. It is worth noting that this accuracy was also achieved by the K = 5 model as well, and that the lowest accuracy achieved was below 0.2 and was held by many values of K.

It is evident that this is not an impressive accuracy at all. While KNN was an affordable option since the dataset is small, it is not nearly comprehensive enough to capture the complex features involved in facial expression. This model only classifies them based on how far apart they are in distance, which is not effective if the observations are not vectorized and transformed

based on their similarity to each other. More advanced models are better at accomplishing this, and so we moved forward to explore them.

SVM

The linear kernel actually managed to achieve a prediction accuracy of 0.54, which is already over 14% more than the accuracy achieved by the K Nearest Neighbors model.

The polynomial kernel, at an optimal 'C' of 2, received a prediction accuracy of 0.70. Other kernels were tested while trying to determine the highest classification score for SVM - such as radial basis function (RBF) and sigmoid - but none of them reached the score of the polynomial kernel. The other kernels also took a different optimal 'C' values - RBF took 3, and sigmoid took 1. The accuracy scores of these other kernels are listed in the table below.

Overall, these results are not stellar, but SVM itself is easy to implement and relatively quick to run.

AlexNet-SVM

The best accuracy achieved by this model was 0.93. Of all the models this was the highest because of the robustness and complexity of the models used. However, there are some tradeoffs to this model. While it achieves the highest accuracy it takes the longest of the four to train.

Overall if you want to work with image classification I think it is the best model to use due to its accuracy, however; if the dataset was bigger and had significantly more images I think it would be beneficial to look into another method where the training does not take as long and the accuracy is still good.

Results Table

Model	**Accuracy Score**
KNN	0.3968
SVM-Linear	0.54
SVM-Polynomial	0.70
SVM-RBF	0.55
SVM-Sigmoid	0.28
CNN from scratch	0.20
AlexNet	0.93

Discussion

The scores and performance of these data sets tell us much about the classification process, particularly when it comes to image processing. The combination of AlexNet and SVM scored the highest by far - with an accuracy score of 0.93 - but the algorithm takes a very long time to run. Simpler algorithms such as KNN and SVM received sub-par scores, but what these algorithms lack in complexity, they make up for in speed and ease of use.

The complexity and computing power of machine learning classifiers has accelerated in recent years, which is promising for the coding community. It appears that combinations and modifications of preexisting classifiers perform the best - for example, ensemble learning meta-algorithms, such as bagging, boosting, and stacking [Smolyakov]. This strength that emerges from increasing algorithmic complexity can be seen in our own experiments with AlexNet and SVM. On the other hand, simple classifiers aren't necessarily bad across the board. Take KNN, for example. While it performed poorly in this image classification task, it generally performs well with images because KNN's entire functionality is based on distances in 2D space, and images are 2D objects [Goswami].

If we were to perform another experiment, we would focus on combinations of simple and complex classifiers, perhaps with ensemble methods stacked on. However, this would be a lengthy experiment - there are so many combinations available.

References

[1] **https://github.com/MUzairZahid/Facial-Expression-Recognition**
https://web.stanford.edu/class/cs231a/prev_projects_2016/emotion-ai-real.pdf

Team GitHub
https://github.com/zainbaq/COGS118B_FinalProject

Dataset
The Japanese Female Facial Expression (JAFFE) Database, Facial Expression Database: Japanese Female Facial Expression (JAFFE) Database. [Online]. Available: http://www.kasrl.org/jaffe.html. [Accessed: 16-Jun-2018].

Goswami, Akash. "Intro to Image Classification with KNN." *Medium*, Medium, 9 Aug. 2018, medium.com/@YearsOfNoLight/intro-to-image-classification-with-knn-987bc112f0c2.

Facial Emotion Detection Using Convolutional Neural Networks and Representational Autoencoder Units
Prudhvi Raj - hhttps://arxiv.org/abs/1706.01509

Smolyakov, Vadim. "Ensemble Learning to Improve Machine Learning Results." *Medium*, Stats and Bots, 7 Mar. 2019, blog.statsbot.co/ensemble-learning-d1dcd548e936.

Xiao, L., Yan, Q., & Deng, S. (2017). Scene classification with improved AlexNet model. *2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*. doi: 10.1109/iske.2017.8258820