Emotion Detection through Static Images using Convolutional Neural Networks

Jihirshu Narayan[1]

[1] The Australian National University, Canberra ACT 0200, Australia
jihirshu.narayan@anu.edu.au

Abstract. Image classification is perhaps one of the most widely implemented application of Neural Networks in industry today and Convolutional Neural Network is perhaps the most effective tool for the same. However, the most basic requirement of a such a deep neural network is a massive dataset. We aimed to demonstrate that CNNs can produce reasonable multi label image classification results even when the model is trained on a dataset of modest size. Additionally, we aimed to demonstrate the effects of neuron pruning on a deep neural network. It was observed that neuron pruning based on output vector in the pattern space is an effective method of building lighter neural networks without compromising on the model performance.

Keywords: Image Classification, Pruning, Emotion Detection, Deep Neural Network, Convolutional Neural Network

1 Introduction

Image processing is rapidly developing and being put to use on an unprecedented scale. Its applications are diverse in all possible aspects, ranging from trivial tasks like recognizing individuals on social media to highly critical tasks such as object recognition in autonomous driving. Deep Neural Network is one of the tools that has been immensely successful in image classification. By principle, a neural network thrives on large databases and the rise of digital media has provided the same.

Emotion detection is a very specific application of image processing. Due to the dynamic nature of facial expression, its features are best extracted from videos. However, under certain circumstances, it becomes imperative that these features are extracted from static images. This classification problem is quite advanced and complicated due to the fact that it has multiple target classes and it's difficult to generalize emotional expressions across individuals. It is therefore extremely important that the training data is massive and equally diverse. For the purpose of this report we will be using the dataset presented by Dhall et al. [3]. The database is a static facial expression database that comprises of unconstrained expressions captured from various angles and at varying resolution. The images are diverse in terms of pose, spatial orientation, subject age, etc. It is named as Static Facial Expressions in the Wild (SFEW) database [3].

Effective image classification requires an accurate and swift predictive model as well. The designed neural networks must be large and robust enough to learn from massive datasets. As such, the networks themselves have multiple layers with large number of neurons. There is no exact method to determine these hyper-parameters and network designers approach the problem with their experience and instinct, sometimes leading to networks that are larger than required and therefore perform sluggishly. Pruning is one of the most effective methods through which a neural network can be reduced in size while maintaining its efficiency, resulting in a lighter and faster model [4]. In recent years, convolutional neural networks have been more popular than any other learning model for image classification, especially in single label image classification task [10]. If the provided database is large and diverse enough, CNNs can handle multiple labels efficiently as well [1].

The classification model used for SFEW dataset by Dhall et al. [3] is based on a non-linear support vector machine which achieved an averaged accuracy of 19% on the Strictly Person Independent dataset. We believe that similar results, if not better, can be achieved with a neural network model. The classification problem was handled over two iterations, each of which produced independent results. In the first iteration, we build a classification model based on a vanilla neural network. In the second iteration, we approach the same problem with a deep convolutional neural network. We also studied the effect of pruning, based on the concept of "Distinctiveness" [4], on both the models.

[1] Student of Master of Computing at Australian National University

2 Dataset

The SFEW comprises of frames selected from AFEW (Acted Facial Expressions in the Wild). It contains a total of 700 images of multiple subjects that vary on various aspects such as resolution, angles and illumination [3]. The 700 samples are labeled with 7 expressions; angry, disgust, fear, happy, sad, surprise and neutral. However, for the purpose of our report 25 images with the 'disgust' label has been left out, leaving us with 675 data instances. This does not mean that there is no sample with the 'disgust' label.

For our preliminary model, we did not use the images itself or even the images' raw data but rather the first 5 principal components each of Local Phase Quantization (LPQ) [8] and Pyramid of Histogram Oriented Gradients (PHOG) features [2]. They are localized feature descriptor vectors that provide information regarding textures and shapes present in the image. For the second model, based on CNN, we used the image data itself with a 720x576 pixel resolution. Both these versions of the dataset were strictly person independent with a total of 95 subjects.

3 Methodology

The steps taken for building both the models were similar, broadly consisting of pre-processing, designing network model, hyperparameter tuning and testing. However, the nature of dataset and task at hand determined the specifics in each of those steps, which are discussed at length in the following sub-sections.

3.1 Dataset Pre-processing

Kotsiantis et al. [6] state that data pre-processing affects the generalizing performance of any supervised learning model notably. It is therefore extremely important that we clean and process the data in such a way that it facilitates pattern learning by the model. In terms of pre-processing, the kind of treatment that a dataset should receive varies from case to case. A well-formed dataset captured with the intention of being used in a learning model may require minimal pre-processing. But for most instances, datasets need to go through steps such as interpolation for missing values, dimensionality reduction, feature encoding, scaling, etc. Since the dataset for our first model is already built upon principal components, there was no need for dimensionality reduction. However, we did encounter missing values and irregular scale across features. We filled the missing values with the mean of the corresponding feature and normalized the feature values using z score evaluation.

The steps for data pre-processing for the CNN based model were more elaborate. Considering that the task focuses on emotion detection, the first step was to detect faces in each image using a combination of multi-task cascaded convolutional neural network and haar cascade classifier. Once the face was detected it was cropped out and resized to 48x48 pixels. The training dataset was also augmented for each epoch of training by random rotation, addition of noise to brightness, contrast and saturation values, and randomly flipping the image across the horizontal axis. Such data augmentation methods are necessary in order to expose the learning model to more variety, thus resulting in a more robust classifier. It is especially relevant in our case because in terms of deep neural networks, a dataset of 675 samples is considered quite small. For instance, a similar 7 label emotion detection classifier based on CNN, achieved an average accuracy of 61% while working on a dataset comprising of 37,000 images provided by Kaggle website [1].

3.2 Neural Network

For our first model, we used a basic feedforward neural network with a single hidden layer of neurons. The output layer had 7 neurons, one each for the target labels, representing the probability that it belongs to a particular label. After testing the performance of various hidden layer activation functions such as Relu, Tanh and Sigmoid, it was observed that the Sigmoid function provided the most robust performance across parameters. Additional hidden layers were tested as well but they did not enhance network performance at all while significantly increasing training time. Similarly, various loss functions and optimizers were implemented and tested. The cross-entropy loss function along with Adam optimizer performed the best across a wide range of parameters. We employed the mini-batch gradient descent for the purpose of weight optimization.

While dealing with the complete dataset, we decided to base our model on a convolutional neural network. Local descriptor-based feature sets, such as PHOG, SIFT, LPQ, etc. have been traditionally employed for the purpose of image recognition. However, in recent years, the popularity of these models has been overshadowed by the emergence of convolutional neural networks (CNN). CNNs are hierarchical structures that extract features through layers and thus produce better generalization [11]. This aspect of learning through CNN is extremely relevant for image recognition because there is high variance in features even in the same target class, thus making the nature of image recognition much more abstract when compared to other forms of data-based learning.

Our CNN model consists of 6 convolution module and each one of them further consists of a convolution layer, batch normalization layer and Relu activation. The first five layers are summarized through max pool method while the last layer uses an adaptive average pooling method. The output size of the first 5 layers gradually increases from 16 to 256 and the last layer condenses the information to 7 output neurons, one for each target label.

3.3 Testing

For our preliminary model, instead of opting for the test set method where we split the initial dataset into training and test set, we went with k-fold cross validation. We also kept 10% of the initial dataset for the purpose of validation. This allowed us to select the most robust and stable model parameters. Since the dataset has limited number of samples, a higher fold cross-validation resulted in very small training folds and therefore 3-fold cross-validation was selected. The use of k fold cross validation added an extra layer of testing which led to the selection of a more robust model.

For our final CNN based model, we split the original dataset into 3 sets. 80% of the original data was kept for training while the remaining 20% was split into test and validation sets. The test set was used to keep track of the accuracy through each epoch of training and the validation set was used as a final hurdle to evaluate the model. In order to get consistent testing values for model evaluation the random seed was kept as ‘0’ through all the iterations.

3.4 Pruning

When constructing a neural network, the designer has no empirical method to determine the correct size of the network. It completely depends on the dataset and the pattern that resides within it. Data visualization and exploration may provide some sense regarding it, but ultimately it comes down to logical guess work, instinct and experience. As such, it is not uncommon for designers to create networks that are larger than required. Network pruning is a useful tool in such cases as it helps to trim down the network to its lightest version without compromising on accuracy.

Network pruning methods have been developed on various levels such as pruning based on weights [5], filter pruning [7] and neuron pruning [4]. We have used the neuron pruning method based on the concept of distinctiveness [4] for both of our models. The degree of distinctiveness is measured by the angle between vectors formed by the neuron’s output activation values formed in the pattern space [4]. In simpler words, it means that each neuron output value obtained by an instance of sample data serves as a co-ordinate for the n-dimensional vector, where n is the total number of samples in the data. Thus, we have a vector associated with each neuron and we determine the angle between all possible pairs of neurons. In order to get values ranging from 0° to 180°, the vectors are normalized in the range [-0.5, 0.5]. A close to 0° angle represents neurons providing similar functionality and an angle nearing 180° represents complimentary neurons. If similar neurons are found, one of the neurons is neutralized and its weight is added to the other. If complementary neurons are found, both are neutralized because they were cancelling out each other’s functionality anyway. The threshold for pruning should be tuned like a hyperparameter and we will see the effects of a large threshold in the next section.

Once the angles for each pair of neuron vectors are calculated, we can start the process of neutralizing the redundant neurons. It is important to keep track of the neurons that have been removed so that over the course of the entire process, we do not end up removing both the neurons of a similar pair. It is also advisable to keep a back-up of the original weights so that it can be referred when we need to add the weight of a neuron that has been already removed. After updating the weights, the validation data is processed through the network and the model accuracy is calculated again. We use the newly calculated model accuracy to compare how the pruning process has affected the model performance.

3.5 Hyperparameter Tuning

Hyperparameter Tuning is the most important step while optimizing any learning model. Pinto et al. [9] argue that image classification performance can be improved better by finding the optimum hyperparameters rather than inventing new learning techniques. In our case, the only parameters that needed to be optimized were learning rate, number of neurons in the hidden layer, number of epochs and the batch size for mini-batch gradient descent. The tuning process was manual, depending mostly on trial and error, aiming to increase the accuracy on the validation set and decreasing over-fitting. For our preliminary model, out of all these parameters, learning rate was perhaps the most crucial one. The correct learning was established after experimenting with a large number of values within the range of 0.00001 to 0.1. Since the pruning technique being implemented for the first model was based on the angle between neuron output vectors in the pattern space [4], the threshold angle for neuron removal was also treated as a hyperparameter. Contradictory to the 15_o threshold suggested [4], it was found that a threshold angle of 10_o removed the optimal number of neurons so that the model performance was not affected.

For our CNN based model, the number of epochs of training had the biggest impact on model accuracy. The reason for this behavior lies in our method of data augmentation, which was discussed earlier. We do not augment the dataset to a fixed size. Rather, we modify the dataset based on random parameters for each epoch of training. Therefore, from the model's perspective, a higher number of epochs of training means that it has more variety of data available at its disposal.

4 Results and Discussion

Dhall et al. [3] state that the baseline averaged accuracy for the Strictly Person Independent category was 19%. The results were achieved with using one of the principal component feature set at a time classified by a non-linear support vector machine. The low accuracy is attributed to the complex nature of problem and large number of target classes [3].

Our first model, based on a basic feed forward neural network, was able to enhance the accuracy of the classifier to 28.21%. With certain parameters, the accuracy remained the same even after pruning neurons in the hidden layer. A complete summary of results with varying parameters is listed in Table 1. It can be observed that as the threshold angle for pruning is increased, the accuracy of the model drops. This is because the increased threshold angle removes neurons which were adding value to the model. Overall, the threshold angle of 10° and 170° removed the highest number of neurons without compromising the accuracy of the model.

Table 1. Vanilla Neural Network Performance Analysis

Neurons in Hidden Layer	Epochs	Learning Rate	Threshold Angle for Pruning	K Fold Average Test Accuracy	Validation Accuracy	Number of Neurons Pruned	Validation Accuracy after Pruning
50	500	0.01	10°, 170°	22.61%	16.67%	0	16.67%
500	500	0.001	10°, 170°	28.3%	28.21%	5	28.21%
500	500	0.01	10°, 170°	26.29%	23.08%	23	23.08%
500	200	0.01	10°, 170°	26.29%	25.64%	45	25.64%
500	500	0.001	15°, 165°	28.3%	28.21%	32	26.92%
500	500	0.01	15°, 165°	26.29%	23.08%	69	21.79%
500	200	0.01	15°, 165°	26.29%	25.64%	88	24.36%

Our second model, based on Convolutional Neural Networks, performed significantly better than our first model, thus confirming our hypothesis that CNNs are better suited for image classification tasks. The best performing model achieved an accuracy of 49.09%. The pruning process was only applied to the last global averaging layer. It was observed that our CNN based models were much more sensitive to pruning than our previous model. Even a threshold of 8_o had observable impact on accuracy. Unlike our vanilla model, we noticed that a lot of iterations showed a slight positive growth in accuracy in spite of significant neuron pruning. This increase in accuracy might be an indication of overfitting in the original model. Overall, we can safely claim that neuron pruning based on the concept of "Distinctiveness" [4] can be effectively applied to even deep CNN based neural networks. A complete summary of results for this model is listed in Table 2. The precision and recall score for each class is captured in Table 3. We can observe that score for the "Disgust" class is significantly lower than the others. Upon close examination of the confusion matrix, we concluded that the model is wrongly classifying images under the "Disgust" labels as "Angry" and "Sad". This lower performance could also be attributed to the fact that the dataset is missing 25 images from the "Disgust" class.

Table 2. Vanilla Neural Network Performance Analysis

Neurons in Final Global Average Pooling Layer	Epochs	Learning Rate	Threshold Angle for Pruning	Average Test Accuracy	Validation Accuracy	Number of Neurons Pruned	Validation Accuracy after Pruning
256	10	0.001	8°, 172°	21.83%	21.82%	111	25.45%
256	10	0.0001	8°, 172°	26.05%	32.73%	7	34.55%
256	20	0.001	10°, 170°	26.83%	45.45%	148	43.64%
256	20	0.0001	10°, 170°	27.81%	41.82%	36	43.64%
256	40	0.0001	8°, 172°	30.11%	49.09%	16	43.64%
256	50	0.001	10°, 170°	33.74%	32.73%	191	36.36%
256	50	0.0001	10°, 170°	31.15%	40.00%	43	40.00%

Table 3. Precision Recall score for 40 epochs and 0.0001 learning rate

Label/Metric	Angry	Disgust	Fear	Happy	Neutral	Sad	Surprise
Recall	0.625	0	0.5	0.77	0.5	0.71	0.2
Precision	0.33	0	0.57	0.875	0.4	0.83	0.33

5 Conclusion and Future Work

The demand for efficient image classification models is rising and it seems unlikely that it is going to decline in the near future. Since there is a lot of scope for improvements as well, we can expect significant development in the coming years. For instance, the concepts we have discussed in this paper need to be built upon for dynamic application such as image classification in videos. Pruning techniques such as the one implemented in this paper will aid the development of models with a shorter response time.

The model accuracy obtained in this report is significantly better than the one reported by Dhall et al. [3], especially considering that the dataset is classified into 7 target labels. Despite the improved model accuracy, we believe that there is a considerable scope of improvement in the prediction accuracy. The use of pre-trained models has proven immensely successful in recent years [1]. These models are trained on a larger dataset and therefore they are able to extract and generalize features better. The output of these models can then be connected to our own custom designed fully connected layers which can be further trained on our own dataset, keeping the weights of the pre-trained model fixed. However, it must be noted that such models would require more resources such as a GPU (Graphics Processing Unit) or Cloud TPU (Tensor Processing Unit). It would also be interesting to explore neuron pruning in convolution layers based on the concept of "Distinctiveness" [4] in a more comprehensive manner, unlike how we have limited our pruning to the last layer. Moreover, there is scope for pruning entire convolution layers, which would result in a much lighter model. We can expect that a filter level pruning technique would be highly successful in such a model [7]. Overall, we believe that this paper would serve as a solid foundation for future work related to image classification on small datasets.

References

1. Alizadeh, S. and Fazel, A., 2017. Convolutional Neural Networks for Facial Expression Recognition, 1704.06756.
2. Dalal, N. and Triggs, B., 2005, June. Histograms of oriented gradients for human detection. In 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05) (Vol. 1, pp. 886-893). IEEE.
3. Dhall, A., Goecke, R., Lucey, S. and Gedeon, T., 2011, November. Static facial expression analysis in tough conditions: Data, evaluation protocol and benchmark. In 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops) (pp. 2106-2112). IEEE.
4. Gedeon, T.D. and Harris, D., 1991. Network reduction techniques. In Proceedings International Conference on Neural Networks Methodologies and Applications (Vol. 1, pp. 119-126).
5. Han, S., Pool, J., Tran, J. and Dally, W., 2015. Learning both weights and connections for efficient neural network. In Advances in neural information processing systems (pp. 1135-1143).
6. Kotsiantis, S.B., Kanellopoulos, D. and Pintelas, P.E., 2006. Data preprocessing for supervised leaning. International Journal of Computer Science, 1(2), pp.111-117.
7. Luo, J.H., Wu, J. and Lin, W., 2017. Thinet: A filter level pruning method for deep neural network compression. In Proceedings of the IEEE international conference on computer vision (pp. 5058-5066).
8. Ojansivu, V. and Heikkilä, J., 2008, July. Blur insensitive texture classification using local phase quantization. In International conference on image and signal processing (pp. 236-243). Springer, Berlin, Heidelberg.
9. Pinto, N., Doukhan, D., DiCarlo, J.J. and Cox, D.D., 2009. A high-throughput screening approach to discovering good forms of biologically inspired visual representation. PLoS computational biology, 5(11).
10. Wei, Y., Xia, W., Lin, M., Huang, J., Ni, B., Dong, J., Zhao, Y. and Yan, S., 2015. HCP: A flexible CNN framework for multi-label image classification. IEEE transactions on pattern analysis and machine intelligence, 38(9), pp.1901-1907.
11. Zheng, L., Yang, Y. and Tian, Q., 2017. SIFT meets CNN: A decade survey of instance retrieval. IEEE transactions on pattern analysis and machine intelligence, 40(5), pp.1224-1244.