Facial recognition with neural network and transfer learning

Sankhya Singh[1]
[1] Research School of Computer science and Engineering,
Australian National University
2601 Canberra, Australia
{u6737668@anu.edu.au}

Abstract. Facial recognition through static images can be done through deep neural networks.However, with deep neural networks the problem of vanishing gradients and high variance is still a major issue. A new proposed Resnet50 quantized transfer learning model has been implemented to the dataset. The effects of pruning and distinctiveness on the trained model weights has been observed. Although, the pretrained Resnet50 model increased the accuracy from the baseline dataset predictions. The combination of quantized transfer learning and Resnet50 resulted in an overall accuracy of 54% on the test set. The described model would be enough to classify static images in raw collection and the application of distinctiveness could help us to realize our pattern weights efficiently.

Keywords: Resnet50, quantized, transfer learning, vanishing gradients, Variance

1 Introduction

Realistic face data plays a vital role in the research of advancement of facial expression analysis systems[1]. The overall research indicates that the lab controlled emotions are not real henceforth they do not depict the raw emotion of humans. Facial recognition has a wide range of applications such as identity authentication,access control and surveillance[2]. The face expression emitted by subjects relies heavily on the background situation or surroundings. The stimulation which occurs on the face should be raw and it shouldn't be forced. Previous study indicates that both JAFFE and CMU Pose Illumination and Expression (PIE) databases are not worthy of doing the recognition as both are lab controlled[1]. Henceforth, our motivation relies on doing facial recognition on SFEW database.

.

(The dataset which was provided to me earlier was thermal stress detection. The expanded version of the dataset was very huge 22 gigabytes of video, furthermore it would be a huge task to form features for every video (for each thermal and RGB) so it forced me to change my dataset to faces emotion)

The SFEW database which is extracted from AFEW (subject reactions are recorded in short movie clips). The database consists of seven emotions: 'anger','sad','surprise','disgust','happy','neutral','Fear'. Each emotion consists of 100 images extracted from movie segments from the past three decades. The images are divided into their expression labels in their respective expression folders[1]. The plain argument which holds the validity of this dataset for facial recognition is that 'even though the actors facial expressions are not hundred percent natural but still the current scenario stimulates them to express the raw expressions which is strong enough to predict their emotion'.

The proposed system in the original dataset considered LBP (local binary pattern) on images and LPQ (local phase quantization 8 bits on the images in AFEW and SFEW) and used SVM for predicting seven class emotion classification problem[1]. The real challenge to classify the dataset (SFEW) was that the images were both high and low resolution which adds further complexity to the problem. The use of Viola-Jones [3] face detector seems to be invariant with this data set because the transformed images with regular standard deviation and mean were able to bring satisfied results.

In order to get a suitable result we shifted our deep cnn network to pretrained Resnet50 and then fused it with quantized transfer learning. The process was done in google colab as the fine tuning of quantized transfer learning requires gpu while the normal Resnet process was done on cpu.

2 Methods

2.1 Resnet50 architecture

The intuition behind implementation of transfer learning and Resnet50 relies on the fact that stacking more layers on deep neural networks causes accuracy to degrade eventually. In order to rectify this problem one can take a shallower model and a deep model that was constructed using the former model and then add identity layers to it. The added identity layers have the same activation function (i.e ReLU) to the previous one. Moreover, the L1 normalization remains the same in the new layer. The benefit of adding identity shortcut mapping is that no additional parameters were placed between those layers.

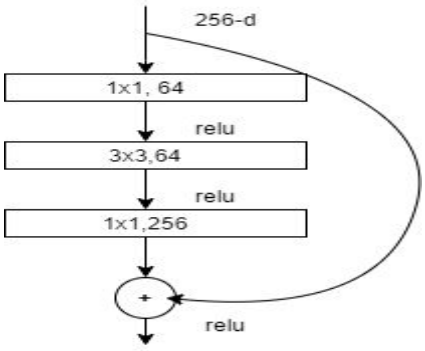

Fig 1 Depicts the three layer stride Resnet50

The kernel is changed after each repetition (1x1,64, 3x3,64, 1x1,256(convolution layers))and for three times there are nine layers, similarly with different kernel size the total amount of the layers is 50 in this architecture. One basic difference between Resnet 18,34 with this Resnet is that it adds an identity layer after three strides as opposed to two strides in the former one.

The model was already pre trained on the image net data and we modified the model by freezing it's final layers to add an extra head layer which is sequential in the nature (i.e suppose the last layer before being frozen is of 4096 dim then adding a layer would be (4096,100) and (100,7) where 7 is the number of classes we want to predict). The overall benefit of using this approach is that total time expenditure on training is reduced and since our model was already trained on a lot of images further classification seems rather easy. Train only on the custom classifier: (head layer) for the task therefore optimizing the model for small dataset.

2.2 Quantized transfer learning

The intuition behind quantization is to speed up the training process. Quantization refers to techniques for performing computations and storing tensor to a lower bitwidths than the floating point precision.Instead of using 32-bit floating point we convert them to 8-bit integers and reduce our training time by four folds. The conversion is simply using the line equation in cartesian geometry *{y = mx + c}*

$$x_{int8} = \frac{x_{float}}{x_{scale}} + x_{offset}$$

The whole conversion of a tensor from floating point to integer is known as Quantization and similarly integer to floating point is dequantization. The equation represented above shows that *xoffset* is the intercept and *xscale* is the slope of the line into the conversion. The dequantization process is similar as one subtracts the offset and multiplies with the negative slope.

The feature extraction by the quantization is done by freezing the Resnet50 classification layer. The choice of number of frozen layers does not require us to set the required_grad to False as it has no trainable parameters.The main idea of quantization is to extract the features from the frozen layer. Henceforth, before defining the quantized model one has to quantize at beginning and dequantize at the last (before our head sequential layer).

2.2.1 Fine tuning quantized model

The next part of the fine tuning model we create the quantized feature extractor after specifically taking consideration in the specific interested dataset.The combined model enjoys the benefits of both quantization and feature selection. The series of steps are taken to ensure the model is quantized:-

1) The (conv(layer) + ReLU) and (conv +bn +ReLU) are fused together using pytorch
2) After dequantization a head layer is added at last which mapps to number of features to classes for classification
3) At last step we add fake quantization step to mimic the quantization in fused model[4]

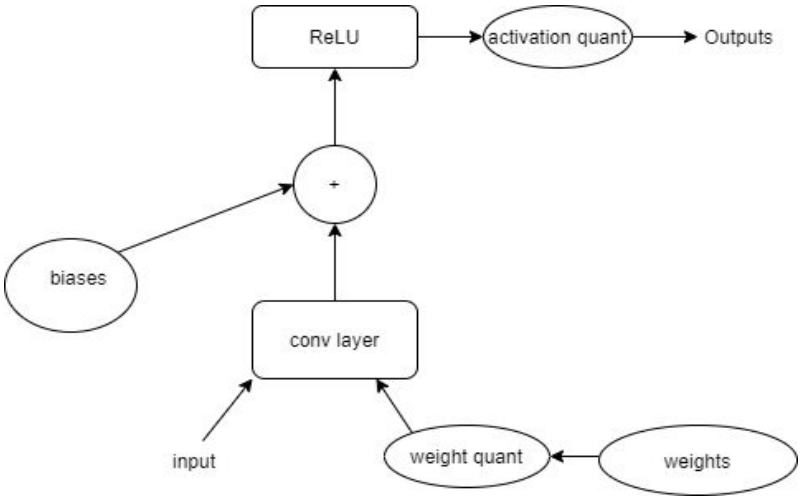

Fig 2 Basic flow chart illustrating fake quantization steps (see source)

Training with simulated quantization of the convolution layer. All variables and computations are carried out using 32-bit floating-point arithmetic. Weight quantization ("wt quant") and activation quantization ("act quant") nodes are injected into the computation graph to simulate the effects of quantization of the variables. The resultant graph approximates the integer-arithmetic-only computation[4].

The fake quantization step is needed to ensure the conversion loss is minimal while converting float precision to the integer bits.

2.3 Network reduction

The measure of distinctiveness on the final head layer can be crucial in understanding the behaviour of the pattern vectors. The distinctiveness of the hidden units is determined from the unit output activation vector over the pattern presentation set[5]. The measure of similarity or complementary neurons comes by calculating the angle between pattern vectors. T.D. Gedeon et al [5] proposed that if the angle between any two pairs of weight vectors is less than 15(degree) then both of these pattern vectors are similar to each other. Henceforth, adding weights to any one of the neurons and deleting the other could result in increased accuracy. Similarly, the pattern vectors which are complimentary have an angle greater than 165(degree). The angle between the vectors is calculated by this formula:-

$$\theta(f_A, f_B) = \cos^{-1} \frac{f_A * f_B}{\|f_A\| * \|f_B\|}$$

The last sequential head layer is obtained from a saved model dictionary. Weights and bias are extracted from that layer and a minimum-maximum scaler is used to convert them within a range of (0,1). In order to obtain the angle between the range of 0 (degree) to 180 (degree) we subtract the whole weights by 0.5. Henceforth, the range of our weights is between (-0.5,0.5).

2.4 Hyperparameters and optimization

Learning rate scheduling was used to decay the learning rate after every epoch. Normally for both the training quantized and fine tune quantized model the number of epochs were 25. Mini batch stochastic gradient descent was used for each iteration.The ultimate goal was to update parameters so that the model churns output closer to the labels. The benefits of scheduling learning rate is that the parameter converges rapidly and we obtain a much higher accuracy for the model.
At every epoch :

$$\eta_t = \eta_{t-1}\gamma$$
$$\gamma = 0.1$$

Where gamma is the decay factor of the learning rate. The updated learning rate is now (1/10)th of the original rate. Similarly we modify the SGD momentum at every epoch:

$$v_t = \gamma v_{t-1} + \eta \cdot \nabla J\left(\theta - \gamma v_{t-1}, x^{i:i+n}, y^{i:i+n}\right)$$
$$\theta = \theta - v_t$$

The theta changes after each iteration with decay in the learning rate.
The original data set contains 100 images from each emotion. The train val test was splitted in (80,10,10) ratio. Pytorch transformers were used to convert them in tensors and apply standard deviation, mean to the passed images.

3 Results and discussion

The results as obtained in the original dataset were calculated by HOG (histogram of oriented gradients) and PHOG(pyramid of histogram of oriented gradients) on seven class classification problems. The accuracy obtained on the SFEW dataset with PHOG and LPQ was 43.71%[1]. While our model achieved 54 % maximum accuracy but only with quantized trained results.

3.1 Pruning results

The below table shows the expanded version of resulted test accuracy after applying pruning (as measure of distinctiveness) the table will broadcast results for two scenarios 1) Quantized resnet50 (last head layer mapping from 2680 to 100) and (100 to 7)
2) Quantized Resnet50 (with head mapping from 2680(num of features) to 7 (classes)

Fig 2 The feature mapping table illustrates the accuracy and similar and deleted neurons

Head Feature Mapping	Similar Neurons in observed weight	Deleted neurons in the observed weight	Testing accuracy before pruning	Testing accuracy after pruning
(100,7)	**4**	**26**	**41.7%**	**38.3%**
(100,7)	**9**	**28**	**45.7%**	**43.7%**
(100,7)	**15**	**21**	**46.9%**	**46.3%**
(2068,7)	**247**	**1388**	**50%**	**51%**
(2068,7)	**329**	**1428**	**52%**	**53.3%**
(2068,7)	**217**	**1370**	**51.9%**	**54.4%**

The above table shows that whenever there is an extra head sequential layer added to the quantized resnet model, overall accuracy slips by 10%. One possible explanation regarding the degradation would be that the absence of ReLU activation function in the final layer. The quantized Resnet model serves as a backbone for this classification process. The model was already trained on Vgg13, Alexnet and deensenet121 but the maximum accuracy obtained from all these were below 45%.

The model however could not achieve accuracy greater than 54% as the images were both high and low resolution. However, if the data was from a lab controlled environment we could have achieved the accuracy above 70%.

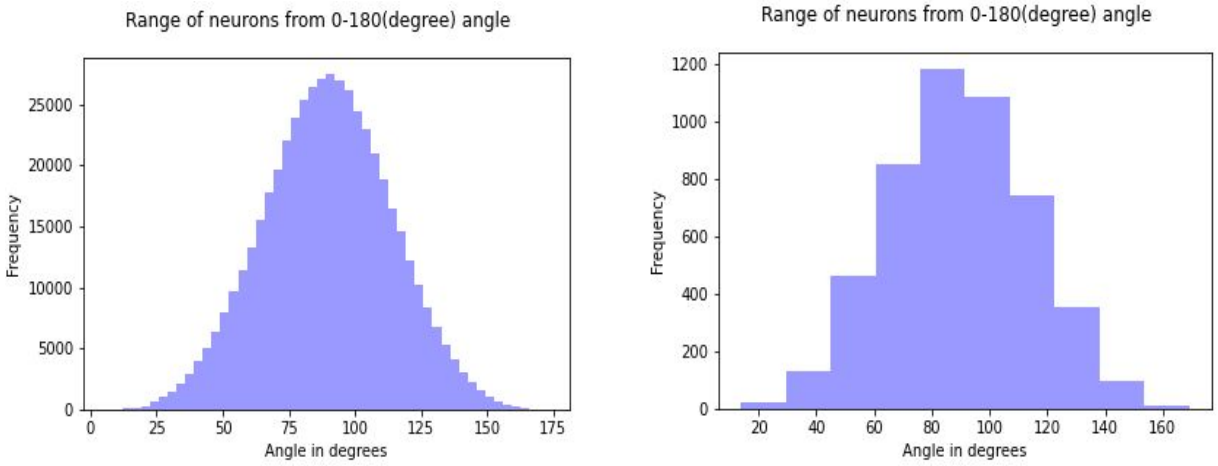

Fig 3 The histogram depicts the variation of angle from (0-180) degree for 1000 features and 100 features.

The above described graphs show the variation of angle with respect to the neurons, as the number of features are increased there is less number of similar neurons as well as complimentary neurons. However, whenever the compression ratio between the pattern vectors is low we obtain high accuracy in our model after pruning. The 100 feature graph also somehow illustrates that even though the number of distinct neurons are higher, the overall accuracy of the model is still depleted (45 to 43)%. The possible reason behind could be that when we are adding an extra head layer in our last sequential layer, the weights which are disposed to zero are being propagated to the next layer.

3.2 Quantized model results

The model was trained using a custom classifier based on Quantized feature extractor.The results were improved but they were not good enough, because the model was not fine tuned and there was very little to no scope for improving the accuracy of the head layers. Moreover, at the starting of training different pretrained models were used. Alexnet being on top of the chart with the least error and less parameters paved a way for realizing the underlying error of the model, and these models had a superb validation accuracy around 44% . However, the model had a problem of high variance. In order to decrease the variance, ensemble (bagging) method was used. Since the quantized model fused mimicked layers of quantization and dequantization, it kind of imitates the ensemble method and this helped to decrease the variance problem.

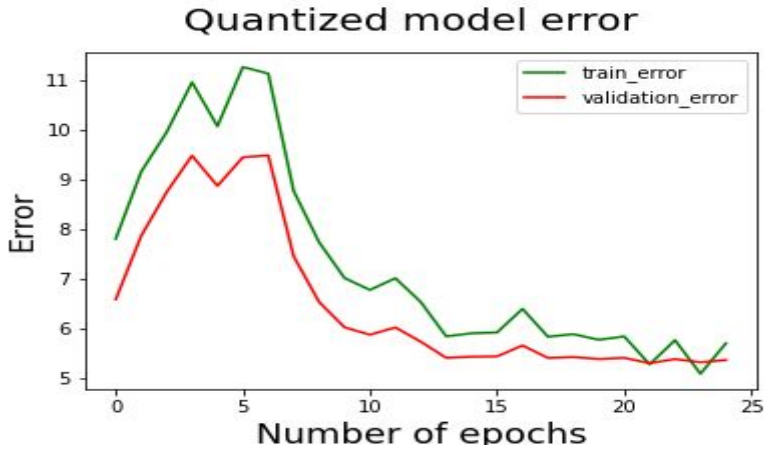

Fig 4 Illustrates the training and validation error of quantized model

The above figure 4 clearly depicts that at the start of the training there is a wider graph between the training and validation loss (problem of high variance) but it starts to converge at last. The best validation accuracy received during the entire lot of training was 38.8%, which is encouraging given the size of the dataset.The dropout of 0.5 was used at last head sequential layer, if the size of training dataset had been increased there could have been better results.

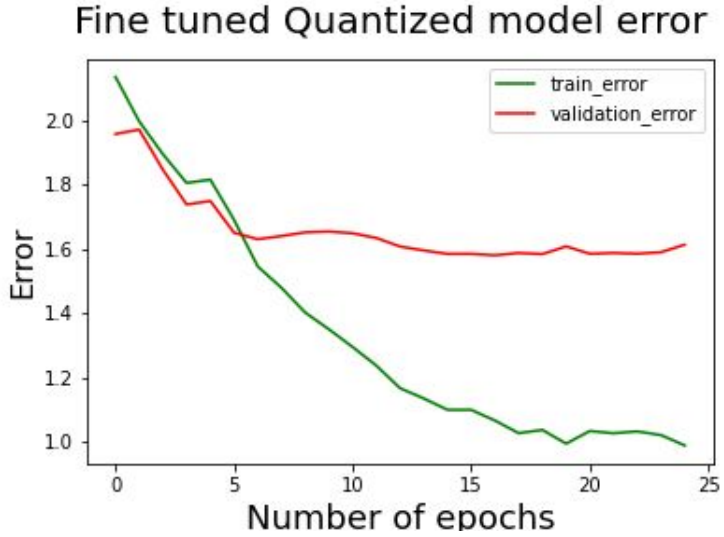

Fig 5 Illustrates the training and validation error of fine tuned Quantized model error

The figure 5 clearly depicts that the model was starting to overfit at the last stages of the training, there is a wider graph between the training and validation error. However, the model achieved best validation accuracy of 50.27 % which is far better than the previous model. The loss function is still decreasing at the end of training, increasing the number of epochs can however increase the validation accuracy. The fine tune model gave a glimpse on how an ensemble learning will look like on quantization layers but still a further improvement can be done on decreasing the variance of the problem.

The model achieved a top most testing accuracy of **54.4%** with the fine tuned quantized model.

Fig 6 Illustrates the testing results from the fine tuned quantized model

The testing results displayed in Fig6 are quite accurate given the accuracy is 54% only. The results could have been further improved if the resolution of the images were high throughout the dataset. Moreover, if the data set could have been vaguely related to any of the pretrained models we could have obtained accuracy above 70% . However, given it is a multi class classification problem the scope of obtaining accuracy above 90% is bleak. Early stopping was also used to prevent overfitting the model but it only increased the bias on the training data and the training was not even done by 10 epochs.

4 Future work and conclusions

In our work we successfully classified the emotion recognition with static images at 54.4% accuracy. The effects of the quantized model significantly improved our test accuracy from 45% to 54%. The model could have been performed better if we had a large number of images in our dataset.
The proposed quantization method is elusive but further work can be done to reduce overfitting. The model can also be experimented with a large number of frozen layers and the head of our sequential layer can also be non-sequential with optimal learning parameters.
The proposed study also illustrated that there can be a lot of work done on quantized transfer learning. The conversion of floating point tensors into integers can not be done on GPU. Henceforth, optimizing the model with cpu can be really hard and tedious.
Network reduction technique[5] also proved vital to recognize why the accuracy had been diminished in some cases. However, this technique works very well for sequential neural networks. In order to achieve the feasible size pruning one could work on a pruning strategy for neural networks[6]. Pruning residual blocks by 50% and the remaining kernel size has been reduced to half[6]. The pruning strategy also works really well for transfer learning as the identity layer normalization weights are not being disturbed by the pruning of kernels.
The overall result by transfer learning and quantization could have improved if we had high resolution images and large quantities of images in the dataset.

References

[1] A. Dhall, R. Goecke, S. Lucey and T. Gedeon, "Static facial expression analysis in tough conditions: Data, evaluation protocol and benchmark," 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), Barcelona, 2011, pp. 2106-2112, doi: 10.1109/ICCVW.2011.6130508.

[2] S. Z. Li and Juwei Lu, "Face recognition using the nearest feature line method," in IEEE Transactions on Neural Networks, vol. 10, no. 2, pp. 439-443, March 1999, doi: 10.1109/72.750575.

[3] P. A. Viola and M. J. Jones. Rapid object detection using a boosted cascade of simple features. In Proceedings of the IEEE International Conference on Computer Vision.

[4]B. Jacob et al., "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, 2018, pp. 2704-2713, doi: 10.1109/CVPR.2018.00286.

[5]Gedeon, T. D., and D. Harris. "Network reduction techniques." Proceedings International Conference on Neural Networks Methodologies and Applications. Vol. 1. 1991.

[6]Y. He, X. Dong, G. Kang, Y. Fu, C. Yan and Y. Yang, "Asymptotic Soft Filter Pruning for Deep Convolutional Neural Networks," in IEEE Transactions on Cybernetics, doi: 10.1109/TCYB.2019.2933477.