RESEARCH ARTICLE OPEN ACCESS

Performance Evaluation of Different Machine Learning Frameworks for Facial Expression Recognition

Md Asif Shahjalal*, Rafi Ud Daula Refat**, Rushrukh Rayan***

*(Electrical and Computer Engineering, University of Michigan - Dearborn, Michigan, USA)
**(Electrical and Computer Engineering, University of Michigan - Dearborn, Michigan, USA)
** (Computer Science and Engineering, Chittagong University of Engineering and Technology, Chittagong, Bangladesh)

--************************----------------------------------

Abstract:

Image processing is one of the hot topics nowadays. Machine learning frameworks like Pytorch, Keras, Tensor Flow etc. all provides support for solving image processing related problems. But each framework has its own philosophy and system requirements. It is difficult for the user to choose the framework perfect for his system. The goal of this work is to provide a comparative analysis of different machine learning algorithms in terms of memory and system requirements. The whole experiment was performed on google collaboration platform, used convolutional neural network algorithm and coded in Pytorch, Keras and Tensor Flow. From the experiments, most of the cases we found out that increasing the batch training size, the training time decreases in all frameworks.

Keywords **—Tensor Flow, Pythorch, CNN, Keras, Facial expression recognition.**

--************************----------------------------------

I. INTRODUCTION

A framework is a collection of libraries and utility computer programs. It is actually a utility application that is built on top of a particular technology and usually has a lot of smaller programs integrated into it. The aim of the framework is to make user life easier. Every framework has its own architecture and has different sets of functions to perform specific tasks. According to a user point of view, for a framework it is not important to know what is running underneath, rather important to know how to use the piece of code. Like all the frameworks, there is machine learning frameworks to make life easier for data scientists. These machine learning frameworks are providing utility functions to provide implementation of various complex machine learning algorithms or even models that is compiled previously. Currently people are using many ml frameworks in the industry and in academia. For example Tensor flow, Keras, Caffe, Pytorch[1] etc.

The goal of our project is to evaluate the performance of different ml frameworks. We have chosen the most popular ones by performing our analysis on Keras, Pytorch and Tensor Flow. The use of different machine learning frameworks is increasing day by day. And the debate of the easy usability of frameworks is never old as well. In [2], Robert Bosch LLC compared five deep learning frameworks in terms of extensibility, utilization of resources and speed performance. In [1], a research group from Wayne State University published their work on CNN algorithm and published in a top conference. The work is confined to the pre-trained models only. The novelty of our work will be implementing our own CNN architecture to compare the different frameworks rather using pre-trained models

Pytorch, Keras and TensorFlow are one of the most popular machine learning frameworks nowadays. All of them are open source. And have a lot of machine learning algorithms implemented within them. Pytorch is built on popular Torch library and Keras is a Python library aiming to provide high level API for machine learning algorithms. On the other hand, TensorFlow is a framework build with the aim of developing machine learning models faster and easier. All of them may differ in philosophy and life cycles. Each of the frameworks have their own lifecycle and philosophy. But all of them aim to solve machine learning in conveniently and achieve higher accuracy.

To compare the three frameworks, we have chosen the human facial expression recognition problem. And we know that for image classification Convolutional neural network is the perfect choice. It is a neural network that can extract features from the dataset automatically and build model. Typically, a convolutional neural network consists of convolutional layers, hidden layers and has down sampling and up sampling layers. Researchers are working on different CNN architectures for decades. Usually, they vary the number of layers or the size of the kernels. That makes them achieve significant difference in accuracy. The popular CNN architecture for image classification are LeNet, AlexNet[2], GoogLeNet[5]. Table 1 provides one overview of them.

TABLE I

YEAR	CNN	LAYERS	DEVELOPED BY
1998	LENET	7	YANN LECUN
2012	ALEXNET	8	SUPERVISION GROUP
2014	GOOGLENET	22	GOOGLE

II. BACKGROUND

Authors in [6], used convolutional neural network (CNN) to improve accuracy in facial expression detection model. The whole experiment was done on a specific dataset (FER 2013). The dataset has seven specific types of images. And traditional work has achieved 62.44% accuracy using CNN. The authors indicated that the face recognition systems include three stages. They can be termed as face detection, biometric feature extraction and classification. PCA has also been used to identify facial expression also.

They have used TensorFlow as a framework to perform their experimental work. Their CNN framework has 3 convolutional layers, a max pooling layer. They have used Relu and softmax as activation function. As an optimizer they have used adam and cross entropy as loss function. For better performance for their model, they have augmented their training images. They have used rotations, centre crop and color jitters etc. Finally, they have achieved 91.12% accuracy on the Fer2013 dataset.

In [7] author explored the use of Multistage Hidden Markov Model (HMM) for detecting facial expressions. For feature extraction they have used partition-based technique and used the modified version of HMM as a classifier. It consists of two layers. The bottom layer represents the atomic expression made by eyes, noses and lips. The upper layer is actually the combination of those atomic expressions. They worked with 6 expressions. The model was evaluated using JAFFE dataset and achieved 82% accuracy. Finally, they have concluded the paper with the suggestion that generative nature HMM is a weak classifier compared to other discriminative classifiers.

[8] classified the most common 7 human expressions. They tried to identify if the expression falls under happy, angry, sad, afraid, disgusted, surprised or a neutral impersonation. They used 28,709 samples for training the model and 3,589 test samples for benchmarking. Two sets of features were used. Eigenfaces were fed as input to SVM and a shallow neural network. In the second set the model was improvised with the help of a Gabor filter. They utilized the python sci-kit learn library for training and testing the model. Multiclass

classification problem was tackled with one vs many approaches. A grid search was utilized to obtain the best possible sets of features. The Neural Network was implemented using Theano framework. The fully connected hidden layer of the NN consists of 3723 neurons and 7 neurons in the output, each corresponding to different class labels. Tanh activation functions were used in the hidden layer. To overcome long training time a high powerful NVIDIA Tesla C2070 GPU has been involved. The DNN consists of an input layer, 3 convolution and max pooling layers along with a fully connected layer and a 7-layer output unit. Out of all different algorithms they showed that CNN performed better in both test and train data sets.

[9] compared different ML algorithms performance to recognize a fully automated facial expression recognition. They utilized popular algorithms like Linear Discriminant Analysis, SVM. For feature selection they used AdaBoost to filter the data before classifying. A subset of Gabor filter with AdaBoost followed by SVM showed the best result in their experiment. The system showed real time accuracy of 93% on novel subjects on the Cohn-Kanade dataset for facial expression.

[10]'s work was based on continuous video input. They utilized Bayesian classifier for classifying facial expression from a video. Bayesian network was chosen for its inherent ability to deal with missing data both for training and prediction. They implemented the system both based on supervised and unsupervised data. They showed that using unsupervised technique make it difficult to assume correct modelling heuristics. They also introduced a stochastic search algorithm for learning Bayesian classifier structure. With a reasonable amount of labelled training data and with a huge unsupervised data they made the performance better by utilizing unlabelled data. Finally, they showed that traditional Naïve Bayes and Tree Augmented Naïve algorithms performance reduce drastically when adding unlabelled data.

Carnegie Mellon University [11] had a benchmark research on this back in 2000. They presented the problem domain for facial expression analysis. They mentioned multiple issues like: temporal organization, level of description, eliciting conditions, individual differences reliability of manually coded expression, in subjects, head orientation and scene complexity, image acquisition, and relation to non-facial behaviour. Then they reasoned the database characteristic on the problem domain and analysed first Phase of the CMU-Pittsburgh AU-Coded Facial Expression Database. This database is a benchmark for a comparative analysis on different facial expression recognition strategies.

III. Design

In this section we are going to discuss about our whole design for comparing the three frameworks. The main goal of our design is to compare them in the frameworks in terms of time complexity, CPU usage and memory usage. While designing, the challenge was not only restricted to the evaluation parameters, but also choosing the appropriate convolutional neural network architecture, the loss function, the optimizer and the dataset. This section is categorized into multiple sub sections. We describe the selection and modification of dataset subsection A. subsection B holds the information of our chosen convolutional neural network, the loss function and the optimizer. We present our approaches for performance measurement of three frameworks in subsection C. Finally, in subsection D we conclude the section with our overall design.

A. Dataset

For facial expression detection, we have chosen dataset named Fer2013 []. The dataset was used in many competitions arranged by Kaggle over the past few years. It has 48*48 grayscale images with 7 types of human facial expression. Each expression is labelled as either 0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral. The training set consists of 28,709 sample images.

It has two types of test samples. One is public test and the other one is private test samples. Each having 3,589 samples. Making the size of the total dataset to 35,887. The paper [] suggests the dataset has highly imbalanced and has close difference among the image categories. So, in order to reduce the influence of the dataset in our framework performance evaluation work, we converted the 7 categories into 3 categories. They are, (i) Positive expression, (ii) Negative expression and (iii) Neutral expression. Positive expression holds the expression of Happy and Surprise, on the other hand we converted the Angry, Fear, Disgust and Sad. Neutral expression remains as neutral here. And finally make our dataset having 21,591 sample. Among them 19228 samples are used as training and 2363 samples are used as validation.

B. Convolutional Neural Network

As the main goal of our project is to compare different frameworks, we have chosen a pretty simple convolutional neural network. The has two convolutional layers, two linear layers and one max pooling layers. Besides we have used rectified linear unit (Relu) and SoftMax pool as our activation functions. We have used batch normalization between each convolutional layer and Relu activation function for speeding up our training. As a loss function we have chosen cross entropy. This is by default given in the three frameworks. Finally, we needed one optimizer. We have chosen the widely used optimizer Adam [12]. It has default implementation in those three frameworks as well. The following figure is the overview of our convolutional neural network.

C. Performance Evaluation Measurements

To make the whole comparison meaningful and reliable, we have chosen a couple of sectors to achieve our goal. We were interested to know about the time required to train in terms of epoch. As the memory and time is generally inversely proportional to each other. We decided to measure the physical memory required in the training phase. For the CPU usage comparison, we considered the system level interrupt calls happened during the training phase. Finally, we wanted to know about the accuracy achieved with the tree frameworks. So, the performance difference in three machine learning frameworks was done in terms of (i) training time, (ii) physical memory usages, (iii) interrupt calls and (iv) test accuracy.

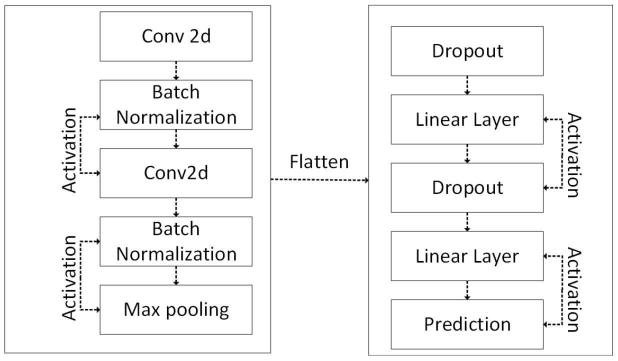


Figure 1: Convolutional neural network

D. OverallDesign

The whole task was done with following traditional machine learning life cycle. We have the data pre-processing phase, training phase and evaluation phase. In the data pre-processing phase, we have only done normalization which means converted

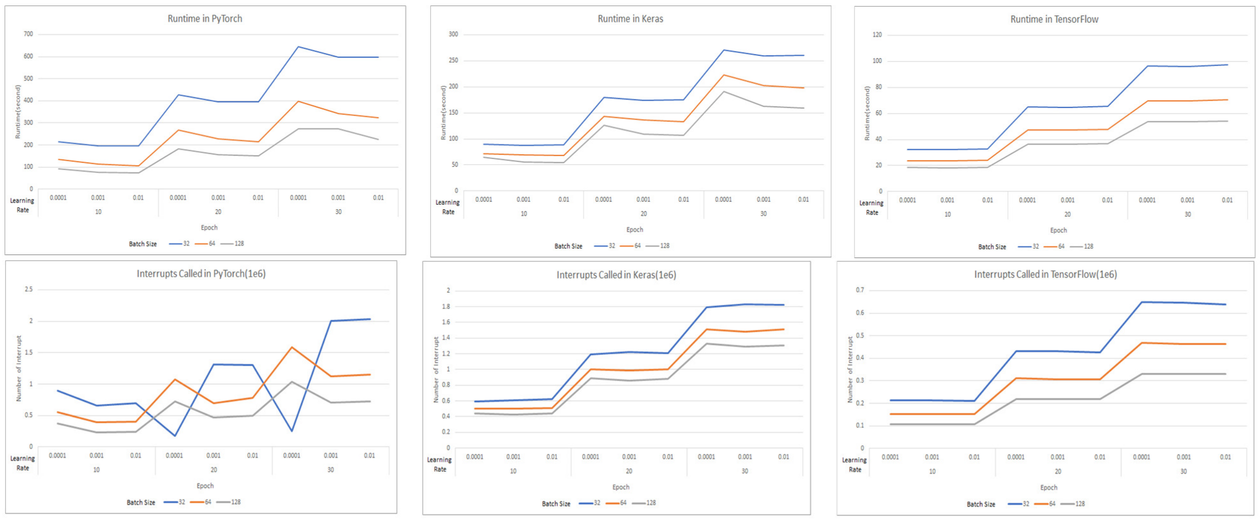


Figure 3: Runtime and Interrupts in different framework for different batch sizes and epochs and learning rate

the grey scale pixel data (0-255) scaled down with in zero and one. After pre-processing we have trained our chosen convolutional neural network with our processed dataset. Finally, we evaluated our model using the evaluation dataset and record the data in terms of training time, physical memory usage, number of system interrupt call and test accuracy. The following figure summarizes our overall design.

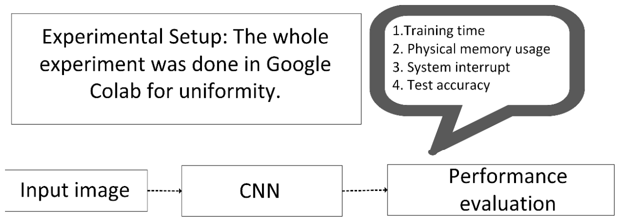


Figure 2: Overview of the system

IV. RESULTS

To keep the runtime environment similar, we used Google Colab in 3 different computers. In terms of runtime, increasing the batch size increased runtime in all of the frameworks. As evident from Figure 3 all 3 frameworks showed a very similar trend here.

In terms of system interrupt call TensorFlow and Keras showed a very similar pattern, which is not surprising considering that Keras is built on top of TensorFlow. Among the frameworks TensorFlow was the fastest to train the model but in trade-off it used the most amount of system memory as well.

We observed almost similar accuracy in all of the frameworks. Increasing the epochs after 10 increased the training accuracy up to 92% but the test accuracy was still not fluctuating much, which was kind of a sign of overfitting. So, epoch 10 would be an ideal parameter for this model.

Training the model in GPU was a lot faster than the CPU. In Figure 5, it shows the memory usage between TensorFlow and PyTorch. As it can be seen that the faster training time in GPU is largely because of the higher memory usage in GPUs. However, comparison between the frameworks in

GPUs won't be an apple-to-apple comparison as Colab uses local system GPUs, which is not the same for the 3 computers used in the experiment.

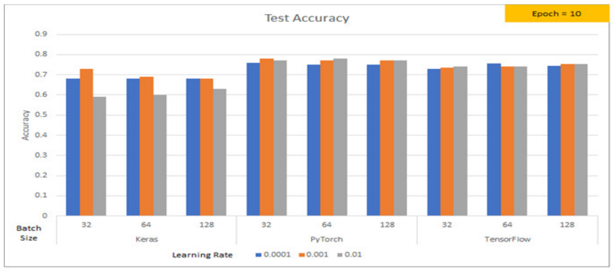
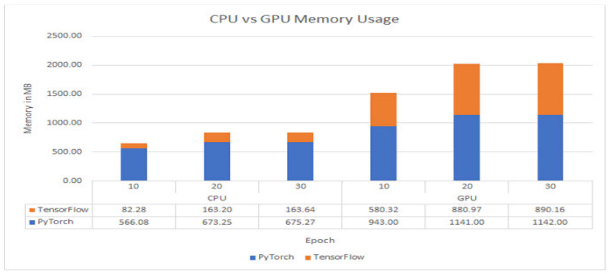


Figure 4: Test Accuracy in different frameworks different batch sizes and epochs and learning rate



	10	20	30	10	20	30
		CPU			GPU	
🟧 TensorFlow	82.28	163.20	163.64	580.32	880.97	890.16
🟦 PyTorch	566.08	673.25	675.27	943.00	1141.00	1142.00

Figure 5: CPU vs GPU memory usages in different frameworks

V. CONCLUSIONS

The results dependent on the framework internal behaviour and data structures. Thus, the comparison between frameworks cannot be one-to-one performed. Nevertheless, the results shown in this project are significant since they were obtained using the same CNN Architecture. In future we want to run native pertained model for those frameworks and compare them.

REFERENCES

[1] Zhang, X., Wang, Y. and Shi, W., 2018. pcamp: Performance comparison of machine learning packages on the edges. In {USENIX} Workshop on Hot Topics in Edge Computing (HotEdge 18).

[2] Bahrampour, S., Ramakrishnan, N., Schott, L. and Shah, M., 2015. Comparative study of deep learning software frameworks. *arXiv preprint arXiv:1511.06435*.

[3] Keras, Keras Documentation. [Online]. Available: http://keras.io.

[4] Sayanthini, Keras vs TensorFlow vs PyTorch : Comparison of the Deep Learning Frameworks. [Online]. Available: https://www.edureka.co/blog/keras-vs-tensorflow-vs-pytorch/

[5] Ballester, P. and Araujo, R.M., 2016, February. On the performance of GoogLeNet and AlexNet applied to sketches. In Thirtieth AAAI Conference on Artificial Intelligence.

[6] Christou N., Kanojiya N. (2019) Human Facial Expression Recognition with Convolution Neural Networks. In: Yang XS., Sherratt S., Dey N., Joshi A. (eds) Third International Congress on Information and Communication Technology. Advances in Intelligent Systems and Computing, vol 797. Springer, Singapore

[7] Rahul M., Mamoria P., Kohli N., Agrawal R. (2019) An Efficient Technique for Facial Expression Recognition Using Multistage Hidden Markov Model. In: Ray K., Sharma T., Rawat S., Saini R., Bandyopadhyay A. (eds) Soft Computing: Theories and Applications. Advances in Intelligent Systems and Computing, vol 742. Springer, Singapore

[8] Chudasama, Bhrugurajsinh, Chinmay Duvedi, and JithinParayil Thomas. "Learning facial expressions from an image."

[9] Bartlett, Marian Stewart, et al. "Recognizing facial expression: machine learning and application to spontaneous behavior." Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 2. IEEE, 2005.

[10] I. Cohen, N. Sebe, F. Cozman, M. Cirelo, and T. Huang. Learning baysian network classifiers for facial expression recognitionusingbothlabeledandunlabeleddata. Computer Vision and Pattern Recognition., 200

[11] T. Kanade, J.F. Cohn, and Y. Tian. Comprehensive database for facial expression analysis. In Proceedings of the fourth IEEE International conference on automatic face and gesture recognition (FG'00), pages 46–53, Grenoble, France, 2000.

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014