Semiconductor Optoelectronics
ISSN:
1001-5868

COMPARATIVE STUDY FOR THE OPTIMIZATION QOS GROWTH RESULTS OF DEEP LEARNING MODELS USING TENSORFLOW

T. Tritva J Kiran[1], Dr. Pramod Pandurang Jadhav[2]
[1]PhD scholar, CSE Dept, Dr. A.P.J. Abdul Kalam university, Indore, MP, INDIA
[2]Associate Professor, CSE Dept, Dr. A.P.J. Abdul Kalam university, Indore, MP, INDIA
{[1]tritvajkiran@gmail.com, [2]ppjadhav21@gmail.com}

Abstract-- As Deep Learning models construction growing rapidly with complex modeling and vast amount of complex data like low resolutions etc. handling, which impacting the optimization throughout the deep learning models and datasets, this work is mainly focused on the comparisons of step by step wise growth results comparisons of how the optimization achieved by which deep learning model and how much the loss function for the model as comparing with previous models and in terms of parameters count improvement etc. all these comparative summarization results study I mentioned in this paper. Obviously deep learning models achieved high accuracy in vision on TPU and GPU as compared with CPU optimization..

Keywords-- Deep Learning, Non-linearity, Classification, Prediction, Representation learning, Adaptive learning, CNN(convolutional Neural Networks), GPU, TPU, TensorFlow.

I. INTRODUCTION

[1] This session introducing the concept of Machine Learning follows with classes of ML and importance of Optimization in ML especially in Deep Learning. Machine learning has become one of the most popular research directions and plays a significant role in many fields, such as machine translation, speech recognition, image recognition, recommendation system, etc. The fundamental nature of the majority of the machine learning algorithms is to construct an optimization model and learn the parameters in the objective function through the given data. According to the modelling purpose and the problem to be solved, machine learning algorithms can be divided into supervised learning, semi-supervised learning, unsupervised learning, and reinforcement learning. Particularly, supervised learning is further divided into the classification problem (e.g., sentence classification, image classification etc.) and regression problem; unsupervised learning is divided into clustering and dimension reduction among others.

[1] - [8] from the perspective of the gradient information in optimization, popular optimization methods can be divided into three categories: first-order optimization methods, which are represented by the widely used stochastic gradient methods. [1] - [8] High-order optimization methods, Deep neural networks (DNNs) have shown great success in pattern recognition and machine learning. There are two very popular NNs, i.e., convolutional neural networks (CNNs)

and recurrent neural networks (RNNs), which play important roles in various fields of machine learning, combining the stochastic gradient descent and the characteristics of its variants is a possible direction to improve the optimization, switching an adaptive algorithm to the stochastic gradient descent method can improve the accuracy and convergence speed of the algorithm [1] - [8].

In the next session I have presented and explained the literature survey as previous research work done on Deep Learning Optimization and methods with results.

II. LITERATURE SURVEY

In this session, we discussing about open problems and challenges for optimization methods in deep machine learning. Also discussing the previous researchers work on deep learning models to achieve optimization. [1]-[8] As for the summarization adding L2 regularization to the objective is a natural method to reduce the model complexity and also for insufficient Data in Training Deep Neural Networks. In broad-spectrum, deep learning is based on big data sets and complex models. It requires a large number of training samples to achieve good training effects but it may lead to high variance and overfitting. There are some techniques in neural networks that can be used to reduce the variance that is the M subnets and those can be sampled like bagging by multiple puts and returns. Each expected result at the output layer is calculated as with High-Order Methods for Stochastic Variational Inference for Optimization Problems in Unsupervised Learning. Here Clustering algorithms divide a group of samples into multiple clusters ensuring that the differences between the optimization problem for the k-means clustering algorithm is formulated as minimizing the loss function. Finally, In the framework of Bayesian methods, some prior distributions are often assumed on parameter θ, which also has the effect of alleviating overfitting. Particularly, in RNNs, the problem of gradient vanishing and gradient explosion is also prone to occur. So far, it is generally solved by specific interaction modes of LSTM and GRU or gradient clipping. Better appropriate solutions for dealing with problems in RNNs are still worth investigating. Finally, summarizing the stochastic methods exhibit powerful capabilities when dealing with large-scale data, especially for first-order optimization.

III. COMPARATIVE STUDY OF ALL OPTIMIZED DEEP LEARNING MODELS OUTPUT RESULTS

Based on the study of previous work results that are mentioned in reference section references [2][3][4][5][6][7][8], the optimization in deep learning models achieved in 6-steps followed as below Step-1 to Step-6 respectively to get more Optimization within Vision Accuracy of Deep Learning Models using various types of Datasets

STEP- 1: Non-Linearity Accuracy for a Deep Learning model [2]

STEP- 2: Computer Vision Classification Accuracy with Deep Learning model [3]

[3] STEP- 3: Deep Transform Learning Vision Accuracy [4]

[4] STEP- 4: Deep Representation Learning QoS – Deep Auto Encoders [5]

STEP- 5: Deep Inceptionism learning performance – Deep Dream Algorithm [6]

STEP- 6: Text Predictions of LSTM RNN Performance– Deep Adaptive Learning [7]

In this section I am presenting a comparative study of above six step-wise growths resulted for QoS in vision accuracy optimization of deep learning models results comparing on TensorFlow with Python. Comparative study work is simulated and tested on Intel® Core™ i3-7100U CPU and GPU with TensorFlow and Python Programming languages platform.

[2] optimization experiment for non-linearity resulted within the initial Epoch, the model showed 0.64 loss but surprisingly at the last epoch the model showed only 0.25 loss rate, this is really good experimental result of accuracy with very low loss rate of 0.1. The result of performance analysis is clearly depicted and can be observed in the figure (1).

In the similar way, though the accuracy is only 60% at the starting epoch but surprisingly at the last epoch the model given greater improvement on accuracy as 88%, which is depicted in the figure(2).

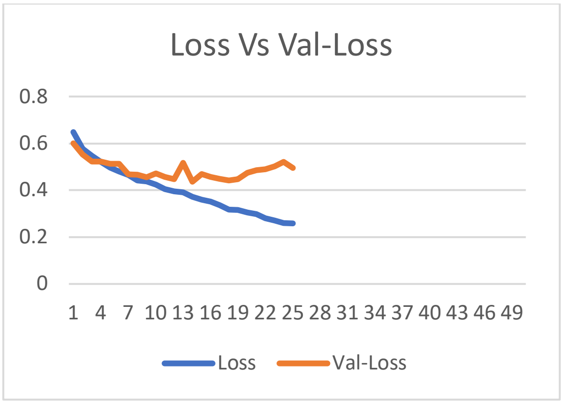


Figure (1) – Loss Vs Val-Loss

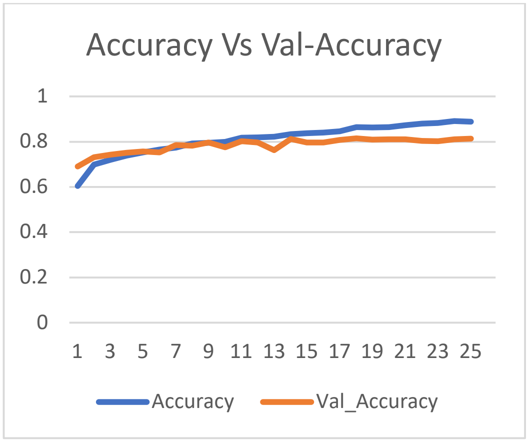


Figure (2) – Accuracy vs Val_Accuracy

The above results are the output analysis for non-linearity in deep learning models optimization, now we need to compare with classification accuracy with STEP 2 [3]. [3] We can observe clearly with the very low-resolution images, the model achieved maximum accuracy with very low error rate with the model having different large datasets with various classes of very low resolutions and also model with a lot of parameters, convolutions and drops. By many Epochs run in the model we can observe clearly the accuracy and loss results in the Figure (3) [3].

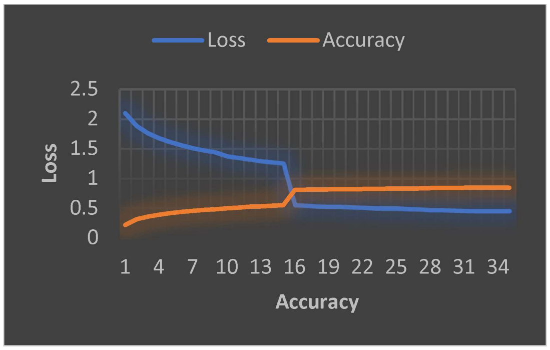


Figure (3) - Accuracy vs Loss Result.

Here, loss rate decreased as number of epochs increased simultaneously the accuracy reached high and we can observe the below output resulted 77% accuracy for the previous work shown below output results, my model on TensorFlow with GPU achieved 85% accuracy compared to previous work [3].

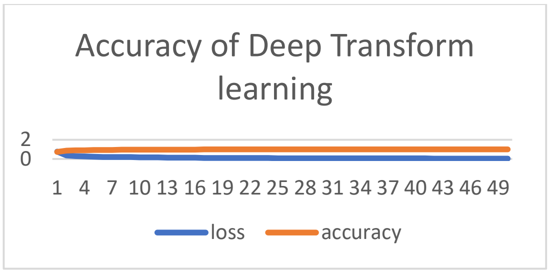


Figure (4)- Accuracy vs Loss

Now we will comparing the optimization results of STEP-1 AND 2 with STEP-3 of Non-linearity, Classification and accuracy by the model Transform learning concept [2][3][4].

Output:

```
313/313 [==============================] - 2s 6ms/step - loss: 0.6811 - accuracy: 0.7737
Test Accuracy: 0.7736999988555908
-------------------------------------------------
```

In this model, I used the dataset ImageNet Flowers dataset and applied over 6 million parameters using sequential method on them using MobileNet model. In the result I got output which showing the number of classes with number of dense layers including the categorical parameters which is shown in below Output Figure (5).

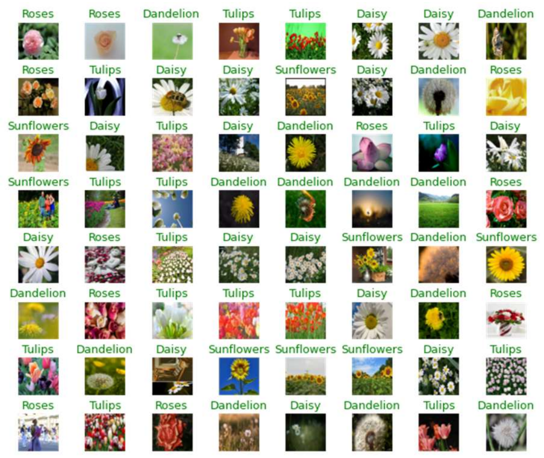
OUTPUT

```
---------------------------------------------------
"sequential_1"
___________________________________________________
Layer (type)      Output Shape          Param #
```

```
================================
keras_layer_1 (KerasLayer)  (None, 1280)          2257984
__________________________________
dense (Dense)           (None, 5)             6405
================================
Total params: 2,264,389
Trainable params: 6,405
Non-trainable params: 2,257,984
```

Figure (5): Dense layers & Parameters Count

In the simulation starting epoch the model shown 71% accuracy but on go, in the end of last epoch model showed great accuracy 99.89% which is almost 100% accuracy the model got on ImageNet data set, which we can clearly see in the Figure (4) given above indicates the Loss and Accuracy comparison result. And all the classes are classified and Predicted by the model correctly without error, which we can see in the above Figure (6), result of the model indicated correct predictions of classes in green label name and if any wrong prediction for class classification happened then that will indicate in red label name. But this Deep Transform learning model predicted and classified with 99.89% accuracy, which is nearly 100% accuracy without errors and with thousands of dense layers and with millions of parameters with almost above 50 epochs the result of accuracy for the predictions and classification for transform learning is very pretty good result. This is indicating the computer vision accuracy is pretty good than human vision accuracy over a low configured network.


Roses Roses Dandelion Tulips Tulips Daisy Daisy Dandelion
Roses Tulips Daisy Daisy Sunflowers Daisy Dandelion Roses
Sunflowers Daisy Tulips Daisy Dandelion Roses Tulips Daisy
Sunflowers Tulips Tulips Dandelion Dandelion Dandelion Dandelion Roses
Daisy Roses Tulips Daisy Daisy Sunflowers Dandelion Sunflowers
Dandelion Roses Tulips Tulips Tulips Daisy Dandelion Roses
Tulips Dandelion Daisy Sunflowers Sunflowers Sunflowers Daisy Tulips
Roses Tulips Roses Dandelion Daisy Dandelion Tulips Dandelion
Figure (6): Classification & Prediction Output

[2][3][4][5] Now we comparing STEP-1 to STEP – 4 results of achieved optimization in deep learning models. All the parameters applied on each layer are clearly shown below during the training of my model and also given all layers parameter values for 4 sequential models. By applying above noise, the result of dataset is resulted with noise is depicted in figure (6) above line numbers and individual noisy numbered I extracted to depict as shown in Figure (6). All

the parameters applied on each layer are clearly shown below during the training of my model and also given all layers parameter values for 4 sequential models.

```
----------------------------------------------------
Layer (type)              Output Shape            Param #
===============================
conv2d (Conv2D)             (None, 28, 28, 16)       160
_________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 16)       0
_________________________________
conv2d_1 (Conv2D)           (None, 14, 14, 8)        1160
_________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 8)          0
_________________________________
conv2d_2 (Conv2D)           (None, 7, 7, 8)          584
_________________________________
up_sampling2d (UpSampling2D) (None, 14, 14, 8)        0
_________________________________
conv2d_transpose (Conv2DTran (None, 14, 14, 8)        584
_________________________________
up_sampling2d_1 (UpSampling2 (None, 28, 28, 8)        0
_________________________________
conv2d_transpose_1 (Conv2DTr (None, 28, 28, 1)        73
=================================
Total params: 2,561
Trainable params: 2,561
Non-trainable params: 0
----------------------------------------------------
```

Model: sequential_1

```
_________________________________
Layer (type)              Output Shape            Param #
================================
conv2d_3 (Conv2D)           (None, 28, 28, 16)       160
================================
Total params: 160
Trainable params: 160
Non-trainable params: 0
```

Model: sequential_2

```
_________________________________
Layer (type)             Output Shape            Param #
================================
conv2d_4 (Conv2D)           (None, 14, 14, 16)       160
```

```
================================
Total params: 160
Trainable params: 160
Non-trainable params: 0
------------------------------------------------
```

Model: sequential_3

```
________________________________
Layer (type)          Output Shape            Param #
================================
conv2d_5 (Conv2D)   (None, 26, 26, 16)        160
================================
Total params: 160
Trainable params: 160
Non-trainable params: 0
----------------------------------------------------
```

Model: sequential_4

```
________________________________
Layer (type)            Output Shape            Param #
================================
conv2d_6 (Conv2D)           (None, 13, 13, 16)        160
================================
Total params: 160
Trainable params: 160
Non-trainable params: 0
--------------------------------------------------
```

The final output is depicted in figure (7).

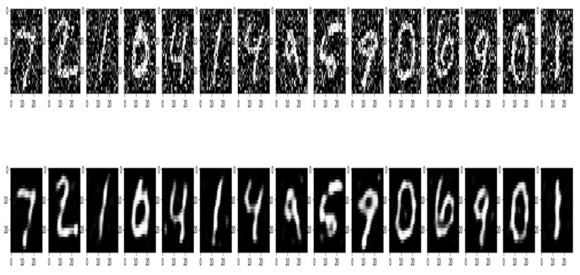


Figure (7): final output

[2]–[6] Here we comparing STEP- 1 to STEP-5 optimized results, by running my model and you can see it all becomes a lot smoother a lot more. Below you can observe the model used how many parameters totally after simulation:

```
-------------------------------------------------
Total params: 21,802,784
Trainable params: 21,768,352
Non-trainable params: 34,432
```

With the above parameters the image is smoothen, the result of output you can see in below figure (7). And for mix values of (225, 375, 3) the result of output image smoothen is shown below figure (8).

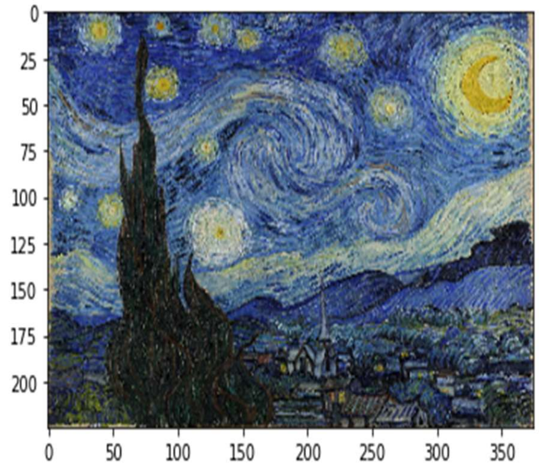


Figure (7): with mix values (225, 375, 3)

And Below we can observe the output of losses and Activation's of Deep Inceptionism learning model.

```
-------------------------------------------------
LOSS (FROM MULTIPLE ACTIVATION LAYERS) = [<tf. Tensor: id=34133, shape= (), dtype=float32, numpy=0.2634555>, <tf. Tensor: id=34135, shape= (), dtype=float32, numpy=0.17727219>]

SHAPE of LOSSES (by MULTIPLE ACTIVATION LAYERS) = (2,)
SUMMATION OF ALL LOSSES (by ALL SELECTED LAYERS) = tf. Tensor (0.4407277, shape= (), dtype=float32)
-------------------------------------------------
```

the losses we got from the above, I am showing the result of smoothen for the input image with the deep dream algorithm for step 0, step 100, step 200 and step 300 with corresponding loss values in Figure (8), Figure (9), Figure (10) and Figure (11) respectively.

Figure (8): Step 0, loss 0.6168044805526733

Figure (9): Step 100, loss 1.4076865911483765

Figure (10): step 200, loss 1.7209874391555786

Figure (11): Step 300, loss 1.90653395652771

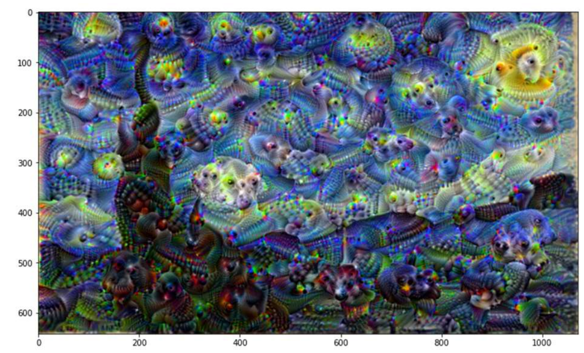


Figure (12): Step 0, loss 0.9181942939758301

And the above figure (12) is showing the final lot more smoothen image output of Deep Inceptionism (Dream) algorithm. Actually, see these kinds of trippy effects because it gives you kind of a sense of how close a eye is to the actual biological neurons that our human level intelligence.

Finally, I want to conclude that, this work achieved a great Optimization in Vision through the testing Performance of Accuracy in both of various integrated large sets of images even with low resolutions and with trippy effects and also for various types of Text classifications and predictions. All the results reached my expectations and given me encouragement for the work and to classifies/Predicts very Accurately and also to get a great Optimization achievement within Computer Vision. In the next session I presented the summary of final conclusion of my work improvement growth up to STEP-5 in regards with using large count of parameters [2]-[8].

Parameters used

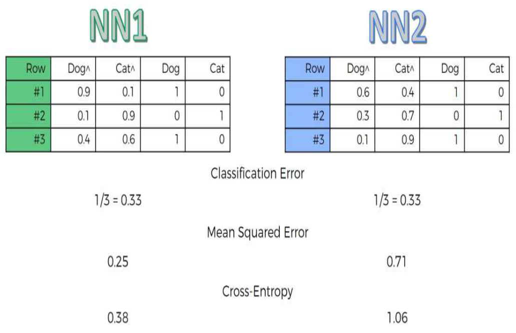
```
Model: "sequential"
_________________________________
Layer (type)            Output Shape          Param #
=================================
embedding (Embedding)       (64, None, 256)          16640
_________________________________
gru (GRU)   (64, None, 1024)         3938304
```

After Loss decreased Parameters OUTPUT

```
./training_checkpoints/ckpt_10
Model: "sequential_1"
_________________________________
Layer (type)            Output Shape          Param #
=================================
embedding_1 (Embedding)      (1, None, 256)           16640
_________________________________
gru_1 (GRU)  (1, None, 1024)  3938304
```

```
________________________________________
dense_1 (Dense)            (1, None, 65)          66625
=========================================
Total params: 4,021,569
Trainable params: 4,021,569
Non-trainable params: 0
```

By all the result and observations, one thing we can conclude that the Cross-Entropy is the preferred method for classification.



NN1

Row	Dog^	Cat^	Dog	Cat
#1	0.9	0.1	1	0
#2	0.1	0.9	0	1
#3	0.4	0.6	1	0

NN2

Row	Dog^	Cat^	Dog	Cat
#1	0.6	0.4	1	0
#2	0.3	0.7	0	1
#3	0.1	0.9	1	0

	NN1	NN2
Classification Error	1/3 = 0.33	1/3 = 0.33
Mean Squared Error	0.25	0.71
Cross-Entropy	0.38	1.06

This work also tested on Python for the comparison with TensorFlow. TensorFlow showed better progress in ETA and accuracy as compared by the Python and is presented the output of Python below.

PYTHON OUTPUT

```
Epoch 1/25
 250/7000 [..............................] - ETA: 1:20:28 - loss: 0.5933 - accuracy: 0.7015
```

IV. CONCLUSION

Disclosure relates to achieve more accurate optimization by measuring the performance analysis of accuracy for the vision on classification and predictions with GPU and TPU using TensorFlow. Though Deep Learning to achieve more vision accuracy QoS performance with the deep convolution neural network (DCNN) configured various neural network components of deep representation learning, deep auto encoder, deep transform learning, deep adaptive learning and deep Inceptionism learning used to achieve more vision accuracy QoS performance & configured for each layer of the DCNN during the classification of an image. it's a great thing the computer vision accuracy performed an excellent vision nearly 97% on TPU in this work.

REFERENCES

[1] Shiliang Sun, Zehui Cao, Han Zhu, and Jing Zhao, A Survey of Optimization Methods from a Machine Learning Perspective, orXiv, 2019.
[2] T. Tritva Jyothi Kiran, Analysis of Non-Linearity Accuracy for a Deep Learning model using GPU on TensorFlow, **Elsevier & Scopus** indexed IJAIEM, VOL 10, Issue 4, April 2021,

ISSN 2319 – 4847

[3] T. Tritva J Kiran, Computer Vision Accuracy Analysis with Deep Learning using TensorFlow, IJIRCST, VOL. 8, Issue 4, July 2020, Page 319-325, ISSN 2347-5552, **DOI:** https://doi.org/10.21276/ijircst.2020.8.4.13

[4] T. Tritva Jyothi Kiran, Deep Transform Learning Vision Accuracy Analysis on GPU using TensorFlow, **Elsevier & Scopus** indexed IJRTE, VOL 9, Issue 3, September 2020, 224-227, ISSN 2277-3878, **DOI:**10.35940/ijrte.C4402.099320

[5] T. Tritva Jyothi Kiran, Deep Inceptionism learning performance analysis using TensorFlow with GPU – Deep Dream Algorithm, JETIR May 2021, Volume 8, Issue 5, ISSN-2349-5162, **DOI**: **http://doi.one/10.1729/Journal.26762**

[6] T. Tritva Jyothi Kiran, Text Predictions of LSTM RNN Performance Analysis using TensorFlow on GPU – Deep Adaptive Learning, JETIR International Journal, Volume 8, Issue 5, 06-05-2021, ISSN: 2349-5162, **DOI:** **http://doi.one/10.1729/Journal.26778.**

[7] T. Tritva Jyothi Kiran, Deep Representation Learning QoS on GPU using TensorFlow – Deep Auto Encoders, **ICCSEA 2021 Conference –** Unique Paper ID: KU/ ICCSEA /August 2021/602.

[8] T. Tritva J Kiran, Computer Science – Deep learning Journal on Optimization in Vision Accuracy of Deep learning models with Tensorflow, IARDO RACE-2022/ June/P714 Scopus indexing conference proceeding.