International Journal of Innovative Research in Computer Science & Technology (IJIRCST)
ISSN: 2347-5552, Volume- 8, Issue- 4, July- 2020
https://doi.org/10.21276/ijircst.2020.8.4.13
www.ijircst.org

Computer Vision Accuracy Analysis with Deep Learning Model Using TensorFlow

T. Tritva Jyothi Kiran

ABSTRACT- Deep learning has absolutely dominated computer vision with creating a model that most accurately classifies the given image in the dataset and surpassing human performance. In previous research works many deep learning models are created and tested for image Classification on various datasets like MNIST, CIFAR-10, ImageNet using Python. Though they got good results of Accuracy for Classification, in this paper I have extended the work of measuring the performance analysis of Accuracy for Classification and also for the Predictions on CPU and GPU using TensorFlow2.0 and Keras on CIFAR-10 dataset having 50000 images of 10 datasets having a lot of different classes with very low resolutions. TensorFlow is an emerging technology on top of Python libraries developed by Google. This work reached an Accuracy 85% on GPU of Intel® Core™ i3-7100U CPU which is acceptable with datasets used in this work are not easy to deal and all with very low resolutions having a lot of classes. That's why it's impacting the performance of the network. To classify and predict very low-resolution images from more datasets is really challenging one, it's a great thing the computer vision accuracy performed excellent in my work.

KEYWORDS- Accuracy, TensorFlow, GPU, Deep Learning, CNN, Classification, Prediction, Python, RELU, Backpropagation, Pooling, Flattening, Loss, Gradient Descent.

I. INTRODUCTION

The main challenge with such a large-scale image classification [13] task is that the diversity of the objects and classes. Thus, any model/algorithm that we use for this task must be able to handle very fine-grained and specific classes, even though they may look very similar and are hard to distinguish.In more technical terms, we want to maximise the inter-class variability [1]. Image classification is a task that is associated with multi-label assignments [1]. It involves the extraction of information from an image and then associating the extracted information to one or more class labels. Image classification within the machine learning domain can be approached as a supervised learning task.

Manuscript received July 22, 2020

T. Tritva Jyothi Kiran, Assistant Professor, Computer Science Department, AKNU, Rajahmundry, Andhra Pradesh, India. (e-mail: tritvajkiran@gmail.com)

A Perceptron is a fundamental component of an artificial neural network, and it was invented by Frank Rosenblatt in 1958. A perceptron utilizes operations based on the threshold logic unit. Perceptron's can be stacked in single layers format, which is capable of solving linear functions. Multilayer perceptron's are capable of solving even more complex functions and have greater processing power. [2] A Multilayer perceptron (MLP) is several layers of perceptron's stacked consecutively one after the other. The [3][1] MLP is composed of one input layer, and one or more layers of TLUs called hidden layers, and one final layer referred to as the output layer[4]. Convolutions use a kernel matrix to scan a given image and apply a filter to obtain a certain effect[4]. An image Kernel may be a matrix used to apply effects like blurring and sharpening. [4] Kernels are utilized in machine learning for feature extraction to pick most vital pixels of an image. Convolution preserves the spatial relationship between pixels. [1] RELU Layers are used to add non-linearity in the feature map. It also enhances the sparsity or how scattered the feature map is. The gradient of the RELU does not vanish as we increase x compared to the sigmoid function, this we can observe in the below Fig (1). [1][5][6] Pooling or down sampling layers are placed after convolutional layers to reduce feature map dimensionality. This improves the computational efficiency while preserving the features.

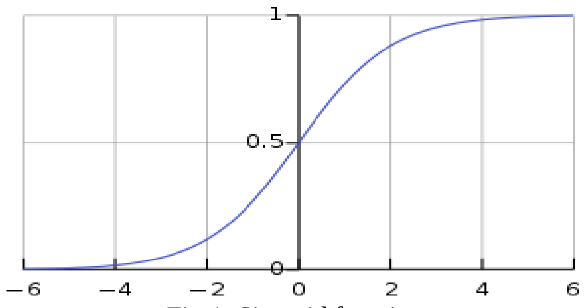

Fig 1: Sigmoid function

Pooling helps the model to generalize by avoiding overfitting. If one of the pixels is shifted, the pooled feature map will still be the same. [14] Max pooling works by recollecting the maximum feature response within a given sample size in a feature map. [12] "Gradient descent is an optimization algorithm used to obtain the optimized network weight and bias values". It works by iteratively trying to minimize the cost function. It works by calculating the gradient of the cost function and moving in the negative direction until the local/global minimum is achieved and if the positive of the gradient is taken, local/global maximum is achieved.[9] [10] Backpropagation is a method used to train ANNs by calculating gradient needed to update network weights. It is commonly used by the gradient descent optimization algorithm to adjust the weight of neurons by calculating the gradient of the loss function, this we can observe in the Fig (2) as shown below.

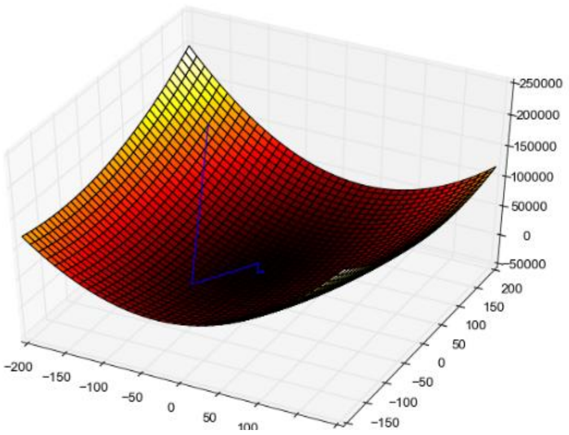

Fig 2: Gradient descent

II. LITERATURE REVIEW

The overall idea of convolution neural networks is first we take the image and then we apply what we call it convolutions. first which is simply we scan the image using what we call it featured detective to extract important features within the image by apply a blue activation function or [19] rectified linear unit activation function. And the objective of Renew activation function is to try to add nonlinear entity to my network. After perform loop what we call it pooling or we call it as well down sampling and the overall idea is, we try to compress the feature maps we want to take this data instead of having let's say 24 pixels by 24 pixels. I want to compress them make them 12 by 12 something like that. So now compressed version of feature maps ready and then take all these pixels and flatten them up and then ready to [18] feed that data into forward. Artificial neural network to perform classification.[15] The overall idea of convolution on neural networks is just a nutshell. The entire training process is trying to find the best or optimal values of these weights that basically connect all the neurons to the next neurons in the next hidden layer. What do you mean by feature detectors? what we call it kernels [4]. The overall idea of convolutions is to use or called the kernel meant matrix and the objective of the kernel matrix is to scan through the entire image and try to find kind of the most important features within the image. [4] Kernels we call them feature extraction. Try to extract important pixels for example in the image some pixels mean nothing, they are basically like vanilla. They don't mean much but some pixels are a lot more important because there are some features, let's say change in intensity between neighbouring pixels. Important pixels to extract some features from the image.

To improve the performance of the network generally there are so many different parameters so many different parameters that can tune to optimize the network and achieve better results. [5] There are mainly three kind of categories or strategies for training. First one is the supervised learning which is simply used if we have input output data. If we have labelled data then apply supervised learning. Supervised learning is used when there is a lot of set of datasets with known labels or not outputs. To label data and the overall objective of the supervised learning into learning algorithm will evaluate the output will start to make predictions and compare the output against label and then the network will try to calculate the error signal and try to adjust the network weights and keep doing that in a supervised fashion.The other strategy is Unsupervised learning. The network will start to create different clusters in the data and try to train in an unsupervised fashion. One of the major drawbacks when it comes to unsupervised learning is that because the learning algorithm works with unlabelled data there is no way to assess the accuracy of the structure suggested by the algorithm because we can't compare it to anything else. So, we can compare the network predictions to our ground truth or the true label and assess the network performance.The last technique or the last way of training is Reinforced learning or Reinforcement learning. And this is simply a learning algorithm that can take action that try to maximize some notion of cumulative reward. The first step, we use the training data to simply calculate the gradient and update the weights and use the validation data set to perform cross validation. The testing dataset is done after the entire network has been trained. Progress accuracy by accumulating more feature detectors/filters or adding a dropout. Dropout states to dropping out parts in a neural network. Neurons develop co-dependency amongst each other during training. Dropout is a regularization technique for dropping over fitting in neural networks. It allows training to ensue on several architectures of the deep neural network. Dataset is generally divided into 50%, 25%, 25% segments for training, validation, and testing, respectively. Training set is used for gradient calculation and weight update. Validation set is used for cross-validation which is performed to assess training quality as training proceeds. Cross-validation is implemented to overcome over-fitting (over-training). Over fitting arises when algorithm emphases on training set details at cost of losing generalization ability. Trained network MSE might be small during training but during testing, the network may exhibit poor generalization performance. Testing set used for testing trained network.

III. LEARNING RATE

Another important parameter when it comes to gradient descent definition is the learning rate. Which network to change or adapt its weights. Learning rate is simply the size of the steps taken. If the learning rate increases that means the network is very hungry and if learning rate increases, the area covered in the search space will increase so we might reach global minimum faster. However, we can overshoot the target for small learning rates, training will take much longer to reach optimized weight values.

- ***Gradient descent works as follows***

[12] Gradient descent is an optimization algorithm that is used to obtain the optimized network weights and by his values. The loss function or minimize error basically which is a difference between network predictions. What is the network is predicting and what's true label is? So, our objective again as I mentioned is to try to optimize the network and find the best values of weights and biases. Gradient descent works by iteratively trying to minimize the cost function. To minimize the error by

calculating the gradient of the cost function. Calculate the derivative (gradient) of the Loss function. Pick random values for parameters m, b and substitute. Calculate the step size (how much are we going to update the parameters?)

$$\boldsymbol{Step\ size\ =\ Slope * learning\ rate}$$

Update the parameters and repeat

$$y = b + m * x$$

Assuming that a straight-line Y equals two X plus B and the objective is to try to find the best values of m and b, y intercept will minimize cost function.

IV. BEST PARAMETERS

$$Loss\ Function\ f(m,b) = \frac{1}{N}\sum_{i=1}^{n}\left(y_i - (b + m * x_i)\right)^2$$

$$gradient\ f'(m,b) = \begin{bmatrix} \frac{df}{dm} \\ \frac{df}{db} \end{bmatrix} = \begin{bmatrix} \frac{1}{N}\sum_{i=1}^{n} -2x_i\left(y_i - (b + m * x_i)\right)^2 \\ \frac{1}{N}\sum_{i=1}^{n} -2\left(y_i - (b + m * x_i)\right)^2 \end{bmatrix}$$

[16]

A. Back Propagation

Back propagation is a method used to train artificial networks by calculating the gradient needed to abate the network weights. Actually, we update our parameters using the back-propagation algorithm.

Back propagation simply works in kind of four steps:

First Step the forward propagation. We take the inputs we feed them through the entire network. Basically, perform all the mathematical operations and multiply all the weights by you know by the neurons then apply the activation function generate the output and propagate the inputs throughout all the layers and generate the output of the end. The second step perform error calculation. Compare what the network is predicting minus error calculation which is simply minus true output or to label. So that's what the network is telling we're going to subtract it minus true target class or ground truth. Basically, it's coming from the training data obviously it is supervised learning. And step three is to perform back propagation is to propagate the error back and try to perform step four which is simply weight update. Take gradients and go back and update weights over and over again throughout series of number of epochs. And that's where basically the simple training which is referred to as back propagation. It is commonly used by the gradient descent optimization algorithm to adjust the weights of the neurons by calculating the gradients of the loss function.

B. Backpropagation Phase 1: Propagation

Propagation forward over the network to produce the output values. Calculation of the cost (error term). Propagation yield activations back over network using training pattern target in order to generate the deltas (difference between targeted and actual output values).

C. Phase 2: Weight update

Calculate weight gradient. A fraction of the weight's gradient is subtracted from the weight. This ratio influences the speed and quality of learning and called learning rate. The greater the ratio, the faster neuron train, but lower ratio, more accurate the training is.

Linear Regression model uses a straight line to fit the training dataset. Linear regression model lacks flexibility so it cannot properly fit the data (as the true perfect model does!). The linear model has a large “bias” which indicates that the model is unable to accurately capture the true relationship between temperature and rental usage. High order polynomial model is able to have a very small bias and can perfectly fit the training dataset. High-order polynomial model is very flexible. Regularization works by reducing the variance at the cost of adding some bias to the model. A trade-off between variance and bias occurs. Variance measures the difference in fits between the training dataset and the testing dataset. If the model generalizes better, the model has small variance which means the model performance is consistent among the training and testing datasets. If the model over fits the training dataset, the model has large variance able to minimize the bias and minimize the variance. We assess the performance simply using confusion matrix. Create matrix in which put all the predictions which is what the model is predicting here on the rows. And we put all the true classes which is our ground truth. We put them on the columns.

A confusion matrix is second-hand to label the performance of a classification model. Below Fig (3) shows sample Confusion matrix. And the prediction values are defined with below terminology.

The case, True positives (TP) predicts when classifier predicted TRUE and correct class was TRUE.

The case, True negatives (TN) raises when model predicted FALSE and correct class was FALSE.

The case, False positives (FP) (Type I error) raises when classifier predicted TRUE, but correct class was FALSE.

The case, False negatives (FN) (Type II error) raises when classifier predicted FALSE.

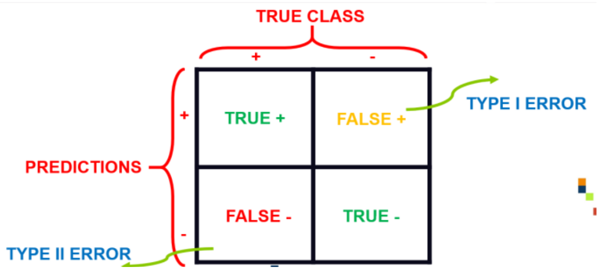

Fig 3: Confusion matrix

V. KEY PERFORMANCE INDICATORS (KPI)

The model used to calculate classification Accuracy is defined as (TP+TN) / (TP + TN + FP + FN).

The Misclassification rate (Error Rate) is calculated with the model using (FP + FN) / (TP + TN + FP + FN) formula.

The precision is defined as TP/Total TRUE Predictions that is defined as TP/ (TP+FP) (When model predicted TRUE class, how often was it right?)
And the Recall is modelled as TP/ Actual TRUE that is defined as TP/ (TP+FN) (when the class was actually TRUE, how often did the classifier get it right?).
In the next section I presented the detailed implementation steps of the model.

VI. IMPLEMENTATION

Tools and Libraries used in this work are:

A. *TensorFlow*

An open-source platform for the implementation, training, and deployment of machine learning models.

B. *Keras*

An open-source library used for the implementation of neural network architectures that run on both CPUs and GPUs.

C. *Pandas*

Data analysis and modification library.

D. *Matplotlib*

Tool utilized to create visualization plots in Python such as charts, graphs and more.

E. *NumPy*

Enables several mathematical computations and operations of array data structures.

First, I mounted the drive and installed TensorFlow 2.0, then loaded the data. I have 50000 images each 32 pixels by 32 pixels by 3. So, it's a coloured image. And the testing dataset consists of 10000 samples each. The next step is going to perform some data visualization. Imported random in the beginning and then we can use Matlock lib to perform Imus show and then can show the actual image along with its original label so it can show extreme along with white train of the index. So, actually they are in order 0 1 2 3 4 5 6 7 and so on. And create grid and within that grid we're going to visualize our image along with the actual label of the image. First built our first layer convolutions fitted two filters each three by three. The kernel three by three and we're going to have a blue activation function and that's what the input shape looks like. The next step added an additional convolution layer 30 to the depth of 32 and each three by three as well. And activation value and then added the max pooling layer of two by two. And then added a drop out here to simply improve the gender network generalization capability and then afterwards added again other convolutions again another convolution. And then again Max pooling layer followed by a drop out. And then flatten the network up and then added a dense network dense network of 1000 almost a thousand neurons then drop it out and then added dense 1000. And that will be the output again. The output having a soft max function. Network again convolutions followed by Max pulling dropout and evolution convolution Max pulling dropout flatten dense dropout dense. And here I have two point seven million parameters. Run it on a GPU. Then afterwards going to compile our model. Specify the optimizer will be Kerry's optimizer and then used the autumn as prop as the optimization strategy. We can also use Adam optimizer. But we found that on Emma's prop actually performs quite well in this case. Specify the loss will be categorical cross entropy. And then the metrics will be accuracy. Create a matrix this matrix consists of seven rows and seven columns and I used subplots. Flatten it up. Then imported confusion matrix. And then imported seaboard as well because to use a heat map out of seaboard and then afterwards call confusion matrix pass along my true classes along with our predictive classes. Now imported libraries and also imported data and here loaded CIFAR-10 datasets and then visualize again couple of data samples and covered it or we discover that the actual resolution of the images is very low. And then created grids or this matrix with all the images along with the labels and then afterwards normalize the data and build the model.So, here I have completed the model and tested by running it on TensorFlow with GPU. In the next section I presented all of my simulation results and also with the previous work comparisons.

VII. RESULTS AND COMPARISIONS

In the first model simulation, the model showing the images in a grid model and successfully the model is assigned the correct labels for different classifications of images. If we observe the result shown in the below Fig (4) the model successfully assigned the labels and classified the images perfectly. And also, we can clearly understand the CIFAR-10 datasets are having very low-resolution images. This is the real challenge to analyse the accuracy for the classification and prediction of the images having very low resolutions. But all the results showed good performance compared to previous work results.

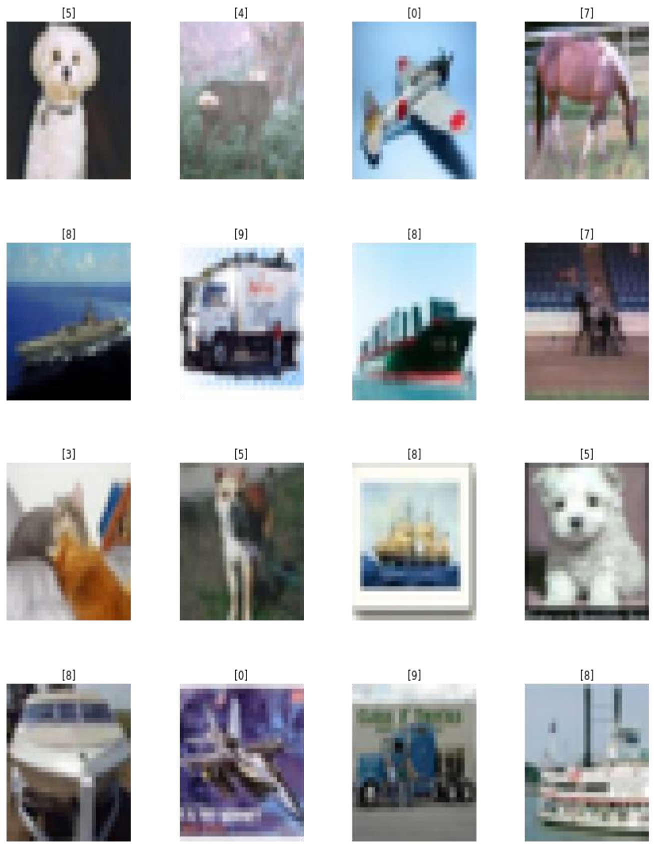

Fig 4: Grid Labelled Classification Results.

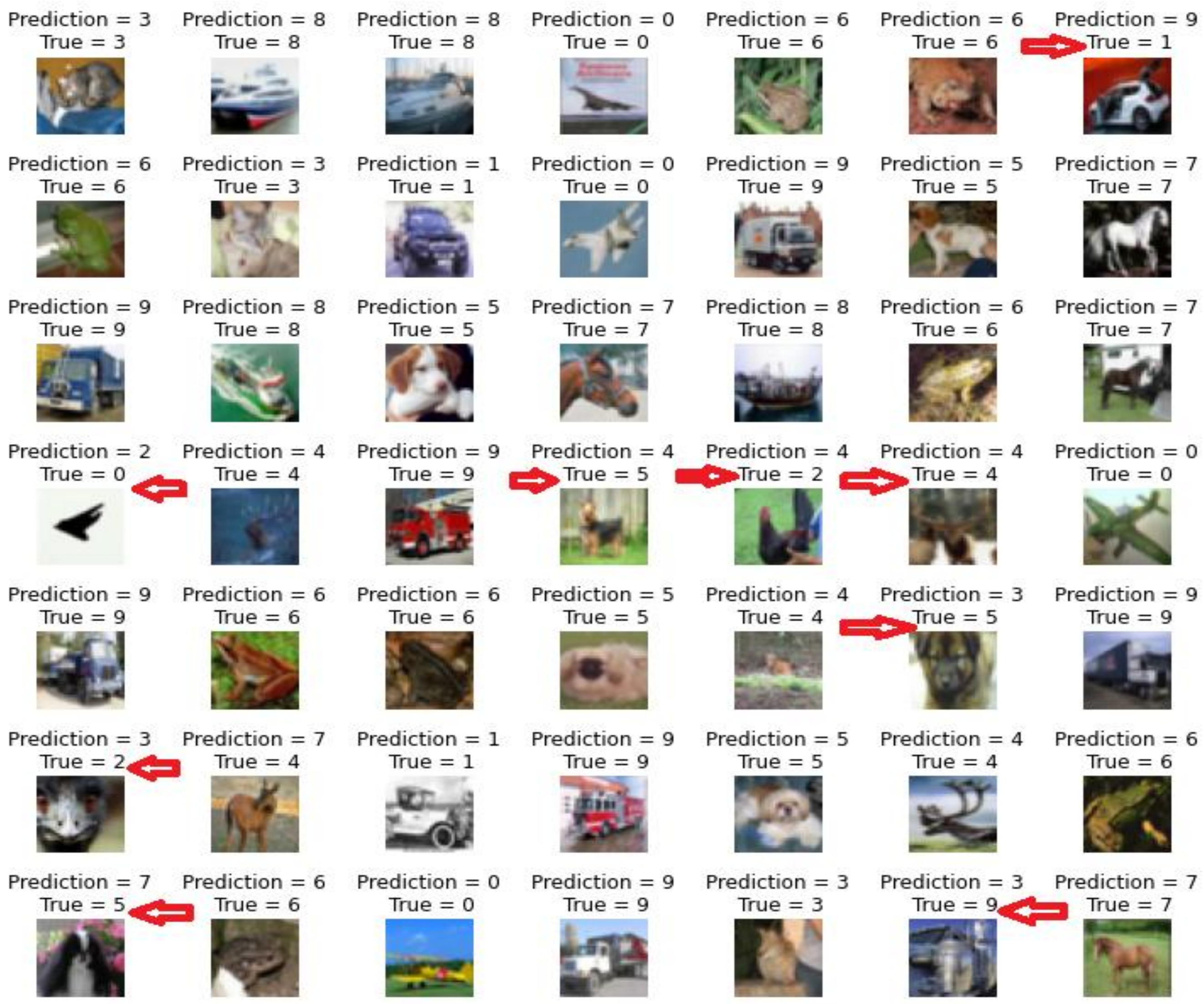

Fig 5: Classification and Prediction Results.

The above Fig (5) showing the results of the model, really the results are encouraging as observed in the above Fig (5), because the dataset with very low resolutions the result of the simulation showed 85% accurate classification and prediction results clearly.
In the Fig (5) the red arrow pointing images are predicted wrong as for example, classification 1 image predicted as 9 in the first row, classification label 0 predicted as 2, but very low percentage of predictions got wrong results as we already know the dataset is with very low resolutions.

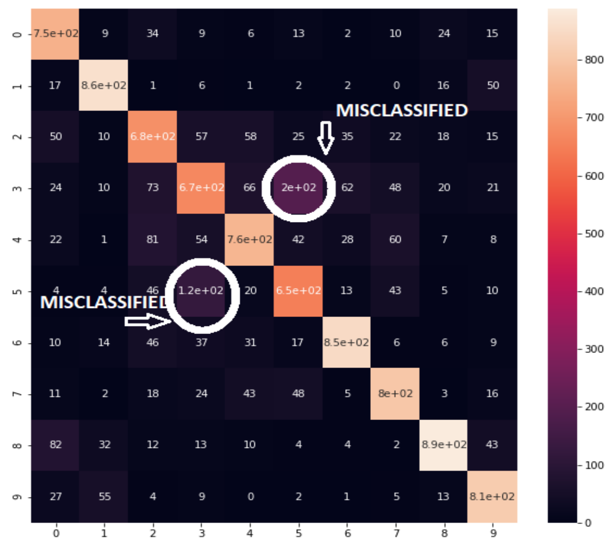

Fig 6: Misclassified Result

The model simulation in TensorFlow Cross matrix grid result for the classifiers and for the mis classifiers results as shown through heat map seaboard in the Fig (6). In the result the white circles showing mis classified error predictions of the model result. 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 Fig (7).

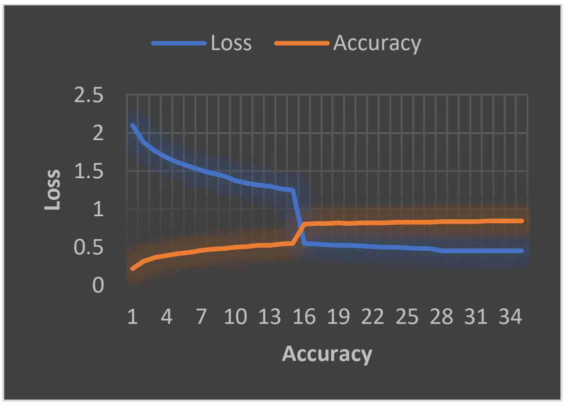

Fig 7: Accuracy vs Loss Result.

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, my model on TensorFlow with GPU achieved 85% accuracy compared to previous work.

Output:

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

VIII. CONCLUSION

To classify and predict very low-resolution images from more datasets with a lot of classes is really challenging one, it's a great thing the computer vision accuracy performed excellent in my work. After a hundred epochs the model reached an accuracy of almost 85%. Which is acceptable with Ten datasets, because again CIFAR-10 datasets actually not easy to deal with first of all the resolution of the actual images is very low and we have a lot of classes as well. That's why it's impacting the performance of the network. And perfect regression model shall have small bias and small variability. A Trade-off between the bias and variance shall be performed for ultimate results. The confusion matrix and the scene that the network is performing is excellent with GPU on my machine Intel® Core™ i3-7100U CPU.

IX. FUTURE SCOPE WORK

This work is tested for computer vision accuracy on Intel® Core™ i3-7100U CPU and GPU and got improvement 85% comparing to previous work. I also have a plan to extend this work to test on TPUs and other different HPC hardware accelerated machines. And also, I have a plan to extend the model simulation by increasing the classes of images with various datasets again with very low resolutions.

X. REFERENCES

[1] Neha Sharma, Vibhor Jain, Anju Mishra, An Analysis of Convolutional Neural Networks for Image classification, Procedia Computer Science 132, 2018.

[2] Mingyuan Xin1 and Yong Wang2, Research on image classification model based on deep convolution neural network, EURASIP Journal on Image and Video Processing, 2019.

[3] Junho Yim, Jeongwoo Ju, Heechul Jung, Junmo Kim, Image Classification Using Convolutional Neural Networks with Multi-stage Feature, Robot Intelligence Technology and Applications, pp 587-59.

[4] Cristian Mateos jun-e Liu1 and Feng-Ping An, Image Classification Algorithm Based on Deep Learning-Kernel Function, 2020.

[5] D.lu & Q Weng, A survey of image classification methods and techniques for improving classification performance, 2007.

[6] Abu Sufian, Paramartha Dutta, Farhana sultana, Advancements in Image Classification using Convolutional Neural Network, 2018.

[7] Md Tohidul islam B.M. Nafiz Karim Siddique Sagidur Rahman, Taskeed Jabid, Image Recognition with Deep Learning, 2018.

[8] https://www.cs.toronto.edu/~kriz/cifar.html

[9] Barry J. Wythoff, "Backpropagation neural networks: A tutorial".

[10] Ernest Istook, Tony Martinez, Improved backpropagation learning in neural networks with windowed momentum, International Journal of Neural Systems, vol. 12, no.3&4, pp. 303-318.

[11] C. C. Jay Kuo, Understanding Convolutional Neural Networks with A Mathematical Model, University of Southern California, Los Angeles, CA 90089-2564, 2016.

[12] Yann LeCun, LeonBottou, Yoshua Bengio, and Patrick Haffner, Gradient-Based Learning Applied to Document Recognition, PROC OF THE IEEE, NOVEMBER 1998.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, Microsoft Research, 2015.

[14] Dominik Scherer, Andreas Muller, and Sven Behnke, Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition, 20th International Conference on Artificial Neural Networks (ICANN), September 2010.

[15] Jianxin Wu, Introduction to Convolutional Neural Networks, LAMDA Group, National Key Lab for Novel Software Technology, Nanjing University, 2017.
[16] Y. lecu, B. boseter, J. SDenker, D Henderson, R.E. hovard, W. hubbred, AND L. Djackal, Backpropagation applied to handwritten zip code reconization, Neural Computation, VOL 1, NO.4PP.541-551, 1989.
[17] B. H. Juang, Deep neural networks a developmental perspective, APSIPA Transactions on Signal and Information Processing, 2016.
[18] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi, Learning activation functions to improve deep neural networks. arXiv:1412.6830, 2014.

ABOUT THE AUTHOR

Mrs. Tritva Jyothi Kiran, Assistant Professor in Computer Science Department, AKNU. Previously. I have completed AICTE funding Project on IEEE802.11e. I have been awarded the National Award for Excellence “Adarsh Vidya Saraswathi Rastriya Puraskar” from Glaciour Global Management in 2020. Present I am working in the research domain Deep Learning using TensorFlow.