Fine Grained Classification

Sandeep Nagar
ML Lab, International Institute of Information Techonology, Hyderbad India
Email: *sandeep.nagar@research.iiit.ac.in, Code and results, Click HERE

***Abstract*—In this project, I have defined a multi-class classification model using the Deep Learning model with the imbalanced image instances of the classes varying from 1-80, using PyTorch.**

***Index Terms*—Deep Learning, Classification, Imbalanced Dataset, Confusion-Matrix, Overfitting, Fine Tuneing, Dropout,Data Augementation**

I. Introduction

Classification problems having multiple classes with an imbalanced dataset present a different challenge than a binary classification problem. The skewed distribution makes many conventional machine learning algorithms less effective, especially in predicting minority class examples.

A. Multiclass Classification:

Image classification using a deep learning algorithm is considered the state-of-the-art in computer vision researches.[1] In the deep learning algorithm, the object feature extracted engineering is done by the algorithm automatically.

B. Imbalanced Dataset:

Imbalanced data typically refers to a problem with classification problems where the classes are not represented equally. For example, you may have a 3-class classification problem of a set of fruits to classify as oranges, apples, or pears with a total of 100 instances. A total of 80 instances are labeled with Class-1 (Oranges), ten instances with Class-2 (Apples), and the remaining ten instances are labeled with Class-3 (Pears). This is an imbalanced dataset and the ratio of 8:1:1. Most classification data sets do not have an exactly equal number of instances in each class, but a slight difference often does not matter. There are problems where a class imbalance is not just expected; it is expected. For example, datasets like those that characterize fraudulent transactions are imbalanced. The vast majority of the transactions will be in the "Not-Fraud" class, and a tiny minority will be in the "Fraud" class.

II. experimentation

A. Dataset:

For the task of training and validation for classification, in the dataset, we have 258 classes having a total of 12,607 images, which is divided into the train(10,000) and test(2,607) using the random distribution 'randomsplit()'.

B. Preprocessing:

Building an effective neural network model requires careful consideration of the network architecture and the input data format.

1) Uniform Aspect Ratio:: One of the first steps is to ensure that the images have the same size and aspect ratio.

2) Image Scaling:: To feed a dataset of images to a convolutional network, they must all be the same size. Once we have ensured that all images are square (or have some predetermined aspect ratio), it is time to scale each image appropriately. I have decided to have images with a width and height of 128 pixels.

3) Data augmentation:: To expose the neural network to a wide variety of variations. The existing dataset with perturbed versions of the current images. scaling, rotations and other affine transformations are typical. This makes it less likely that the neural network recognizes unwanted characteristics in the dataset.

4) Mean, Standard Deviation of input data:: We might want to normalize the dataset such that the mean value of each data sample would be equal to the mean given below. It's useful to look at the 'mean image' obtained by taking the mean values for each pixel across all training examples. Observing this could give us insight into some underlying structure in the images. For this to ensure, I have calculated the mean and std of the dataset.

a) DataSet's Mean: tensor([0.4829, 0.4329, 0.3960])

b) Std:: tensor([0.2112, 0.1898, 0.1821])

5) Normalizing image inputs:: This makes convergence faster while training the network. Data normalization is done by subtracting the mean from each pixel and then dividing the result by the standard deviation. The distribution of such data would resemble a Gaussian curve centered at zero.

C. Cross Entropy Loss:

Cross entropy loss function (CEL) is a widespread loss function for training DCNNs in the image classification task. This is because it measures the difference between target class distribution and predicted class distribution and can be reduced by stochastic gradient descent (SGD) methods. In multi-class classification tasks, the binary cross-entropy loss function can be extended to categorical cross-entropy (CCE) loss for one-hot encoding.[2]

D. Optimizer

The Adam optimizer is super easy to use and tends to settle at a reasonable learning rate all on its own. SGD, on the other hand, usually gets you a nice 1–2 percent boost over Adam but is much harder to tune. Stochastic Gradient Descent(used): To overcome the shortcomings of BGD, stochastic gradient descent (SGD) was introduced. SGD allows updating the network weights per each training image. Furthermore, SGD with momentum renders some speed to the optimization and also helps escape local minima better.

AdaGrad: The learning rate is tuned automatically by dividing the learning rate by the sum of squares of all previous gradients. To scale the learning rate for each weight.

Adam Optimizer: To combine the benefits of Nesterov momentum, AdaGrad, and RMSProp algorithms.

E. Deep Learning Model:

Deep Models Matter.

I directly run the constructed 5-layer CNNs classifier as an optimizer for the baseline without any dropout or data augmentation to be compared with the following models as a shallow model. The baseline perform not well that has only about 70 percent test accuracy and there exists string overfitting since training accuracy is much better than validation accuracy and validation loss value is more significant than training loss value.

InceptionResNet[3]: Inspired by the performance of ResNet, Google presents InceptionResNet. Residual is added to the output of the convolution operation of the inception module. After convolution, the depth is increased, and this model achieves top-5 error on the ImageNet classification. InceptionResNet takes the idea of an inception network and a deep residual network. It accelerates the speed of training and improves accuracy with about 467 layers in total. It shows the power of deep layers as well.

F. Performance Evaluation:

Classification is one of the two sections of supervised learning, and it deals with data from different categories. The training dataset trains the model to predict the unknown labels of population data. [4]

Measuring the area under the ROC curve is also an advantageous method for evaluating a model.

Measurements on Confusion Matric.

G. Confusion-Matrix(Conflicting Classes):

With imbalanced classes, it's easy to get a high accuracy without actually making useful predictions. So, accuracy as an evaluation metrics makes sense only if the class labels are uniformly distributed. In the case of imbalanced classes, confusion-matrix is a good technique for summarizing the performance of a classification algorithm.

When we closely look at the confusion matrix, we see that the classes with very few samples indeed have very few scores compared to the classes with a higher number of samples. Thus looking at the confusion matrix, one can see how the model performs on classifying various classes. We have many classes(258), which makes it challenging to find the conflicting classes.

A confusion matrix is at the end of 'finegrained-logQ298700txt' in the folder of output files.

III. Results

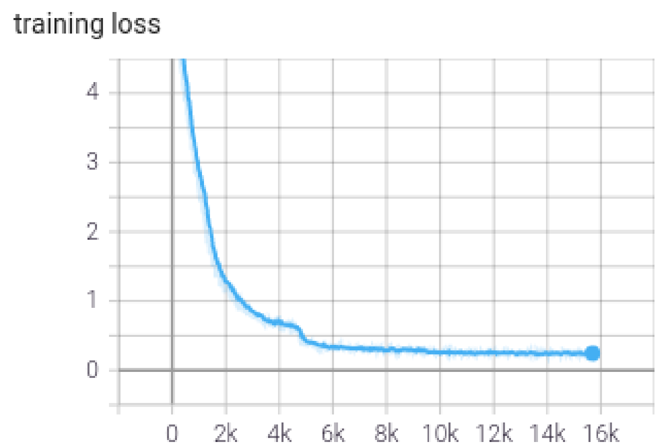
I have trained the Resnet50[5] Deep Learning model.

The classwise accuracy of the classes and the accuracy for the classes with images less than 20 is in the result(output) file. "finegrainedlogQ298550txt".

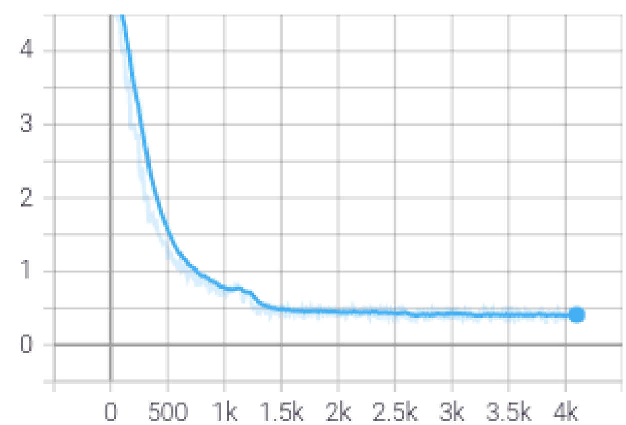
In the table below, there is the accuracy of the test and train with top1, top5, avg(for the 10000 and 2607 images)

The loss for the Train and Test is shown in fig(a), fig(b) below, respectively.

acc	top1	top5	avg.
train	95	99	95
test	91	95	92



(a) train loss



(b) test loss

IV. Conclusion

The objective of this project is to show how deep models like VGG16, VGG19, Inception V3, InceptionResNet V2 can be used on minimal size data with a total of 12,607 images, including 10,000 training data, 2,607 validation data, without severe overfitting and with outstanding performance. All of the models are without pre-trained on Resnet50. According to the experiments, all of the models perform well. Specifically, I apply data augmentation, dropout, and fine-tune these models. It turns out it barely has any overfitting.

As proved from experiments, deep models can fit a tiny dataset as long as the suitable model is chosen and proper modifications are applied. The key to adopting the deep models on the small dataset is data augmentation with dropout and fine-tuning.

Choosing proper deep models to fit in tiny datasets is crucial to determine the performance. Although overfitting is reduced a lot during the experiment process, underfitting occurs for some models. The reason why this happens may be that too many weights are dropped. Overall, the experiments prove that deep models can fit in tiny datasets with proper modifications without severe overfitting.

V. Future Work

Experiments on different models with proper modifications can be carried on in the future. For example, the issue like why fine-tuning these models leads to the increase of validation loss and how underfitting/Overfitting happens in the models.

Modifying more models to better fit in tiny datasets and comparing them is meaningful to work on in the future.

Data augmentation, dropout, and fine-tune the these models for the dataset to minimized the overfitting and improve the accuracy for class with very few image instances.

We can use the Transfer learning/ pre-trained model, which will decrease the training time of the models.

References

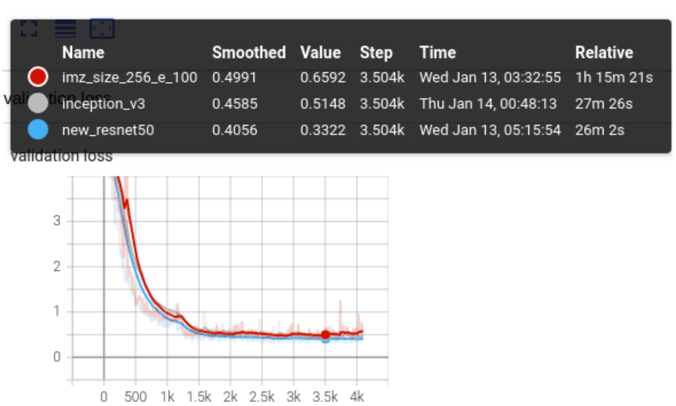
[1] W. A. Ezat, M. M. Dessouky, and N. A. Ismail, "Multi-class image classification using deep learning algorithm," *Journal of Physics: Conference Series*, vol. 1447, p. 012021, jan 2020. [Online]. Available: https://doi.org/10.1088/1742-6596/1447/1/012021

[2] K. Zhang, X. Wang, Y. Guo, D. Chang, Z. Zhao, Z. Ma, and T. X. Han, "Competing ratio loss for discriminative multi-class image classification," *CoRR*, vol. abs/1912.11642, 2019. [Online]. Available: http://arxiv.org/abs/1912.11642

[3] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

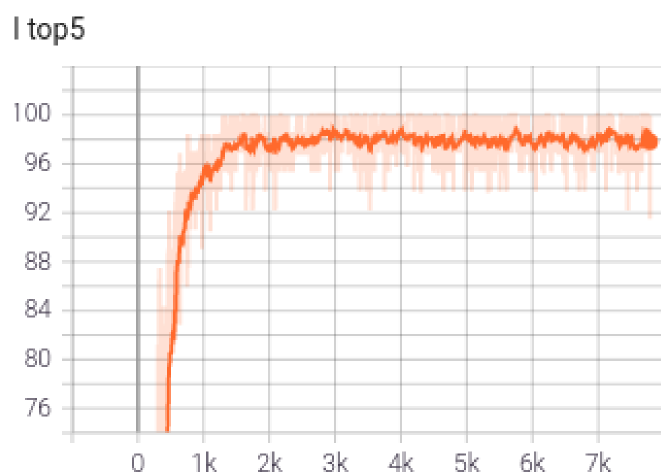
[4] P. Flach, "Performance evaluation in machine learning: The good, the bad, the ugly and the way forward," in *AAAI19*, 2012.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

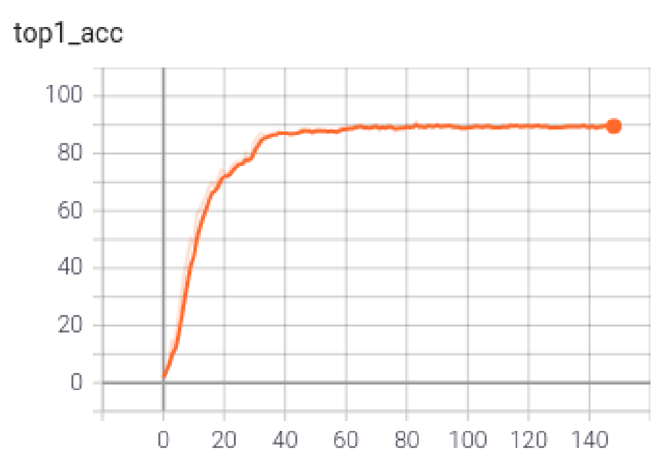


(c) test loss

(d) test acc top1



(e) test acc top5



(f) train acc top1