ECE472 Deep Learning Final Project
Professor Curro

On Optimizing Confidence Level for Small Dataset

Dan Brody
Seyun Kim

2020 Fall

Abstract

With the advance in deep neural network architectures and especially convolutional networks, deep learning has achieved great success in image recognition and classification. While many of the current state of the art convolutional neural networks produce high accuracies, it is often expected to do so under the premise that the dataset being processed has sufficiently enough data to extract information from. Also, deep learning algorithm's outputs are often taken blindly and assumed to be accurate, which is not always the case. Combining the two cases, we propose a novel method to optimize confidence level for small datasets which will alleviate the weakness of small datasets and provide a way to be "confident" for the outputs. To do so, we suggest applying [2] Online Hard Example Mining(OHEM) and add [1]Cosine loss to the existing cross entropy loss. We tested our method on Chexpert, a dataset of chest X-rays, and a variety of other datasets including the Oxford Flowers dataset. The result showed that the regular OHEM works better than our proposed method and we argue that the reason for this is the overconfidence of the model on its predictions. Further research can be done to reduce confidence and produce better results.

Introduction

Constantly, there has been demand for large amounts of data for various types of datasets. However, it is also an obvious fact that developing as much of such huge datasets as it is demanded is almost impossible, both due to the limitations in time and cost. In the case of image labels, it takes considerable time and cost to annotate and label each image accurately and sometimes it takes human input to process all the data so that it is usable in deep learning methods. It is thus not surprising that there are many cases where the limitation is not the lack of knowledge about the topic but the available data to train and test their theory on. However, obviously enough, there are way more small datasets out in the real life than large datasets. For example, one's photo gallery in smartphone can be a useful dataset but is neither annotated nor "large" in the perspective of deep learning where millions of input images is not uncommon (ImageNet contains 14,197,122 images). Then, it would be immensely beneficial to the development of deep learning applicable to real life if we could devise a method that can work well with small datasets and make use of infinitely many small datasets available in our everyday life. Instead of having to search for an appropriately large dataset or make a new dataset that can be impractical and even infeasible to work on a CNN model, using a small dataset that is more readily available than the large one would be the better option to most people.

Also, understanding confidence in deep learning is more important than it is credited for. One reason for a poor performance of a model can be due the model's overconfidence on its predictions. When a neural network makes a prediction, it has a certain degree of confidence that the prediction it gave is correct. Assuming that the confidence is reasonable, the model will update its weights in an optimal direction. However, when the model confidence is not optimal - small confidence for correct predictions and high confidence for wrong predictions - the model there is a high probability that the model output will not be optimal. Therefore, in this paper, the authors also focus on optimizing confidence so that the model confidence on its predictions will not impede the model performance.

To this end, we propose to apply [1] cosine loss function to [2] Online Hard Example Mining(OHEM) to develop a model that can produce good results for classifying small datasets, dubbed as the cosine OHEM:

$Top\ K\ Loss\ =\ \lambda L_{cos}(softmax(x), y)\ + L_{cross}(softmax(x), y)$ (1)

$Top\ K\ Index\ Select\ =\ indexes\ of\ k\ greatest\ losses\ from\ Top\ K\ Loss$ (2)

$k\ =\ ratio\ *\ shape(y)[0]$ (3)

$L_{NLL}(x', y'),\ x', y'\ = train\ and\ test\ examples\ whose\ indexes\ were\ chosen\ from\ Top\ K\ Index\ Select$

(4)

The remainder of the paper is organized as the following. In the related works section in which we talk about Cosine Loss and OHEM we will discuss concepts relevant to the overall content of the paper. In the datasets section, we introduce the datasets used for our experiments in the paper.The experiments section and conclusion section discuss the summary and evaluation of our proposition and results.

Related works

OHEM

Online Hard Example Mining, which will be referred to as OHEM in the following sentences, provides a simple but effective algorithm for training state-of-the-art image processing models based on convolutional neural networks. Broadly, OHEM solves the problem of lazy learning where the ConvNet model does not learn anything significant from the dataset because it is largely composed of easy samples and hard samples are neglected as “unknown.” OHEM solves this problem by sampling hard examples to get trained by the model, rather than multiple easy ones which can only consume memory and time.

As shown in Figure 1, OHEM can be divided into two parts: convolutional network and RoI(Region of Interest) network. The convolutional network is the traditional ConvNet which computes the feature map. The RoI network then uses this feature map and the input RoIs to compute softmax classification loss for each RoI. Then, input RoIs with large loss, the "hard examples," are fed into the RoI network to update the model. In this way, the OHEM algorithm selects examples for which the model is proved to be weak at classifying.

We chose to incorporate OHEM as it does not require too many extra hyperparameters while still yielding a significant increase in accuracy.

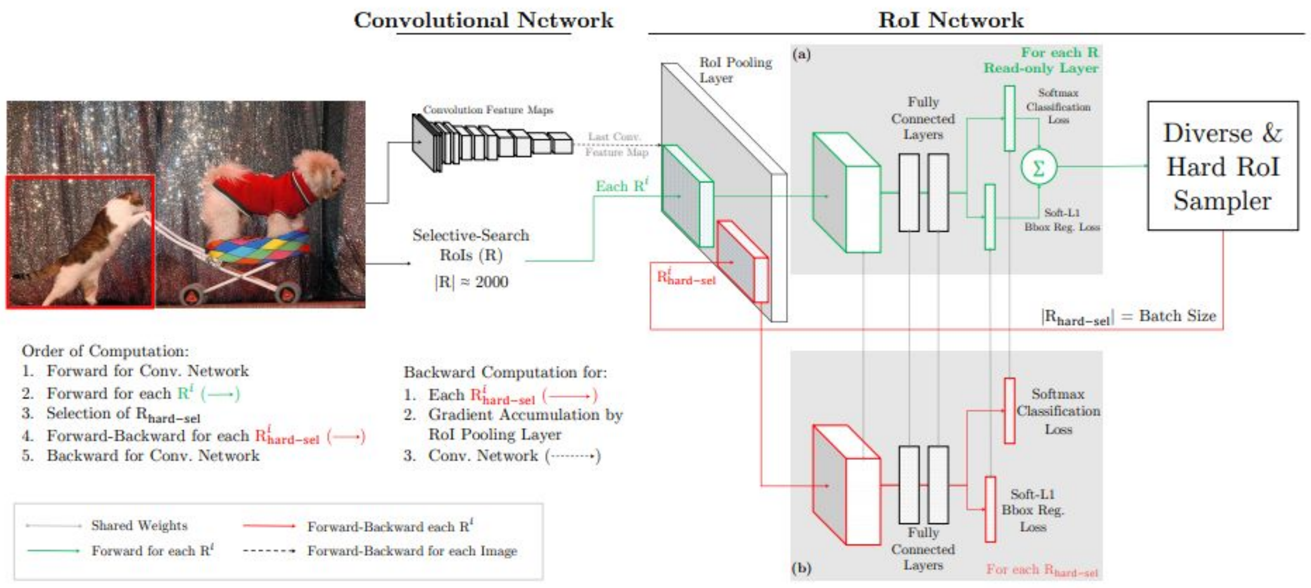
Convolutional Network
RoI Network
Convolution Feature Maps
Last Conv. Feature Map
Each R^i
Selective-Search RoIs (R)
|R| ≈ 2000
RoI Pooling Layer
(a)
For each R
Read-only Layer
Fully Connected Layers
Softmax Classification Loss
Soft-L1 Bbox Reg. Loss
Diverse & Hard RoI Sampler
R^i hard-sel
|R hard-sel| = Batch Size
Order of Computation:
1. Forward for Conv. Network
2. Forward for each R^i
3. Selection of R hard-sel
4. Forward-Backward for each R^i hard-sel
5. Backward for Conv. Network
Backward Computation for:
1. Each R^i hard-sel
2. Gradient Accumulation by RoI Pooling Layer
3. Conv. Network
Shared Weights
Forward for each R^i
Forward-Backward each R^i
Forward-Backward for each Image
(b)
For each R hard-sel

Figure 1. [2] Architecture of Online Hard Example Mining(OHEM). Note that RoI Network is not used when experimenting for this paper.

Cosine Loss

It is widely believed that categorical cross entropy works better than most of the loss functions available nowadays in image classifying and that small datasets often perform not as good as big datasets in a CNN classifier. In paper [1] *Deep Learning on Small Datasets without Pre-Training using Cosine Loss,* the authors, B. Barz and J. Denzler, examine and experiment the performance of cosine loss function when adding it to categorical cross-entropy, for classifying images of datasets that contain a small number of samples per class (They define less than 200 images per class as "small dataset"). In their paper, they were able to show that when cosine loss and cross entropy loss are combined the model performs significantly better than the conventional categorical cross entropy function on small datasets. Specifically, on a CUB-200-2011 dataset, they achieved 30% higher accuracy with cosine loss than with categorical cross-entropy without pre-training.

The cosine similarity between two n-dimensional vectors is defined as the following equation (1), where $a\angle b$ represents angle between vector a and b, $||a||_2$ represents the L^2 norm, and <a, b> dot product between a and b. Then, let x be an instance of an input and y its label or class. Then, the cosine loss can be written as equation (3), where f_θ refers to the output of the neural network with θ as the trainable parameters and phi(y) refers to the one-hot encoding of the ground truth label in equation (2). In the end, the cosine loss function seeks to maximize the cosine similarity between the output of the neural network and one-hot vectors indicating the true class.

$$\sigma_{cos}(a, b) = cos(a\angle b) = \frac{<a,b>}{||a||_2 \cdot ||b||_2} \quad (5)$$

$$\varphi_{onehot}(y) = [0\ (y-1\ times)\ \cdots\ 1\ \cdots\ 0\ (n-y\ times)]^T \quad (6)$$

$$L_{cos}(x, y) = 1 - \sigma_{cos}(f_\theta, \varphi(y)) \quad (7)$$

[1] B. Barz and J. Denzler also compares the cosine loss with two other popular loss functions: categorical cross entropy and mean-squared error. As Figure 2 shows, compared to the other two loss functions, cosine loss exhibits two distinctive properties. First, the loss values are well ranged between 0 and 2, while softmax+cross-entropy reaches until 20 and MSE, even worse, ranges from 0 to more than 200. Such large loss values can cause the model weights to shift too much. Also, another property is that loss values of cosine loss is well distributed within its range [0,2], which can be seen from gradual change in color in its heatmap Figure 2.c. For softmax + cross-entropy, the loss value is mostly less than 0, represented as a dark blue area in the heatmap. Also, the color change in (a) is barely visible, meaning that small variations can cause comparatively large change in loss value. MSE is similar to (a) except that MSE shows a more uneven distribution of loss values with yellow area barely showing in the corners and most of the values concentrated in the ranges 0 to 100 and 200 to 300. Unlike (a), however, MSE exhibits a gradual change in color.

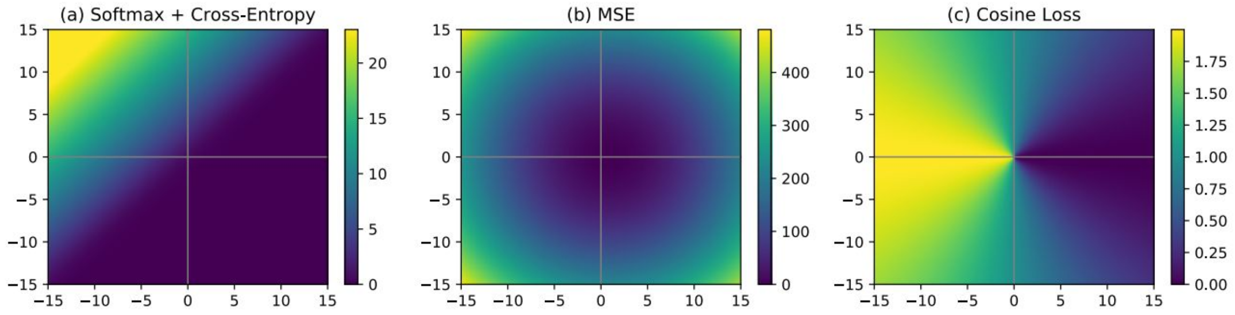

Figure 2. Heatmap of different loss functions

Congenerous Cosine Loss (COCO Loss)

[6] One variation of cosine loss worth discussing in this paper is Congenerous Cosine Loss (COCO Loss) which seeks to minimize inter-class variations and enlarge intra-class

differences to achieve better classification accuracy compared to the previous state-of-the-arts and increase confidence level in classification tasks.

Unlike regular cosine loss which computes the cosine distance(similarity) between the ground truth and predicted value, COCO loss computes cosine distance between two features. Then, the cosine distance computed is directly compared and optimized.

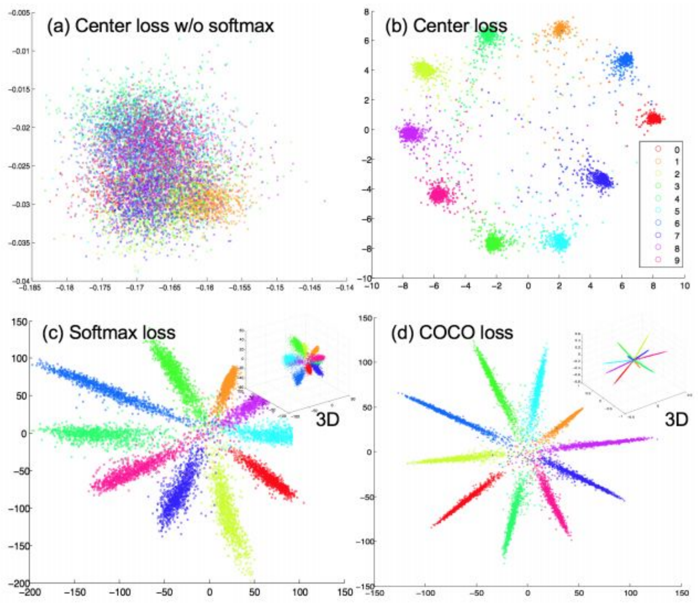

Figure 3. Feature visualization of using different losses, trained on MNIST.

Figure 3 shows the feature visualization of MNIST dataset using different loss functions. Here, we are going to focus on the softmax loss and COCO loss. It can clearly be inferred from the figure that COCO loss is distinctively better at minimizing inter-class variations and maximizing intra-class differences. In COCO loss, each class spreads away from each other as much as possible while remaining dense with its instances. On the other hand, softmax loss is poorer than COCO loss at minimizing inter-class variations - more instances in each class is separated from the class mainstream, looking more sparse. Also, compared to COCO loss, softmax loss is weaker at maximizing intra-class differences as the classes on the right side of the visualization are not as long as those on the left, indicating that intra-class difference is not maximized generally.

Dataset

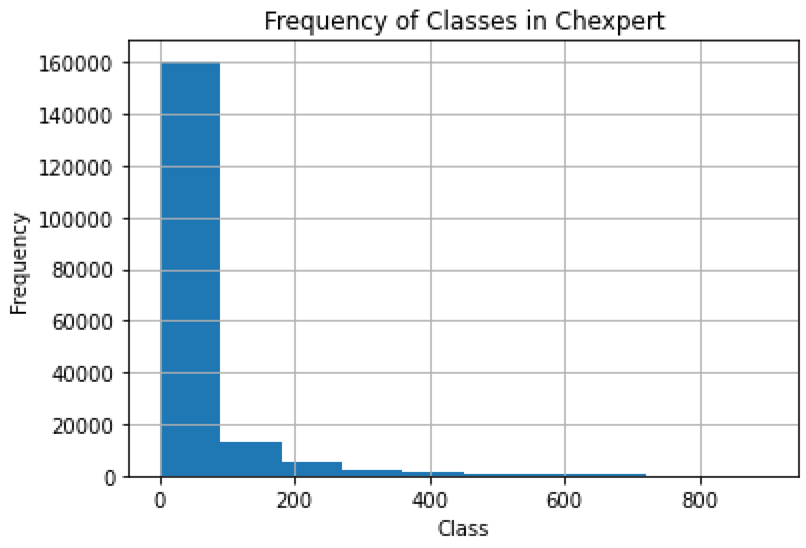

Figure 4. Frequency distribution of classes in Chexpert. The classes are ground into ranges of 100 for better visualization. The dataset is highly imbalanced, where examples in the dataset are concentrated in classes 0-100.

In this paper, we define small datasets as those that have 500 or less examples in each class and are preferably imbalanced. Among the publicly available small datasets, we first explored medical datasets as they are often small and imbalanced. [3] Chexpert is a chest radiograph interpretation published by Stanford University School of Medicine, consisting of 224,316 chest radiographs of 65,240 patients. The reason we chose this dataset was because, As Figure 4 shows, the dataset is highly imbalanced and small (in terms of features per class). This dataset needed some preprocessing, however, before it could be used. The dataset had 13 columns with different attributes that were identified by the chest radiograph such as Enlarged Cardiomediastinum,Cardiomegaly, and Lung Opacity and we later narrowed this down to 12 after finding that one of the columns, Support Devices, was very common among the rows and not a very useful diagnosis. From these 12 attributes the rows in the table had a 1 if the patient had that attribute, a -1 if the result was unknown, and a 0 if the attribute was not contained in the radiograph. We decided to then create unique classes out of the 12 attributes by making a set a class if its set of ones was not already denoted as a class e.g. {Cardiomegaly, Lung Opacity, Cardiomediastinium}. The authors have also made it possible to specify the number of classes, maximum features per class, and minimum features per class to adapt to different use cases. Doubly, because the training dataset is so large,223,414 training examples, the authors split the training dataset in the function into a train and test set. Best results on Chexpert was when the number of features per class is between 100 and 500 and the number of classes is 40. This subset has 7076 training examples and 3521 training examples. We also tried to use [4] MRNet, also distributed by Stanford University Medical Center, consisting of knee MRI examinations. There are a total of 1,370 knee MRI tests of which contains 1,104 (80.6%) abnormal exams, with 319 (23.3%) ACL tears and 508 (37.1%) meniscal tears. Ground truth values of the exams were obtained by manually extracting test results from clinical reports. This dataset, in its original form, had three classes - abnormal, ACL tear, or meniscal tear - which the authors concluded to be too few to demonstrate the functionality of our model. Therefore, a pre-processing was done to the dataset to increase the number of classes. First, each original three classes - abnormal, ACL tear, and meniscal tear - was given a binary value 1 or 0 depending on whether or not it was diagnosed to the patient. As a result, there were a total of eight classes in the end, representing every possible combination of the original three classes. For example, if a patient was diagnosed with abnormal knee but neither of ACL or meniscal tear, that patient is class 3. However, this dataset was not applied eventually to our model for a set of reasons. First, as Figure 3 shows, there were no instances of several classes in the dataset, making it no more different from MRNet before pre-processing. Also, although a minor problem compared to the first reason, because MRNet is a medical dataset, a concern was raised that it might be redundant with Chexpert which is also a medical dataset. Overall, the authors decided not to use MRNet as it contained too few classes even after pre-processing and was similar to Chexpert.

Abnormal	ACL tear	Menisc	Label
0	0	0	0
0	0	1	1
0	1	0	2
1	0	0	3
0	1	1	4
1	0	1	5
1	1	0	6
1	1	1	7

Table 1. Classes and features of MRNet dataset after pre-preprocessing to increase the number of classes.

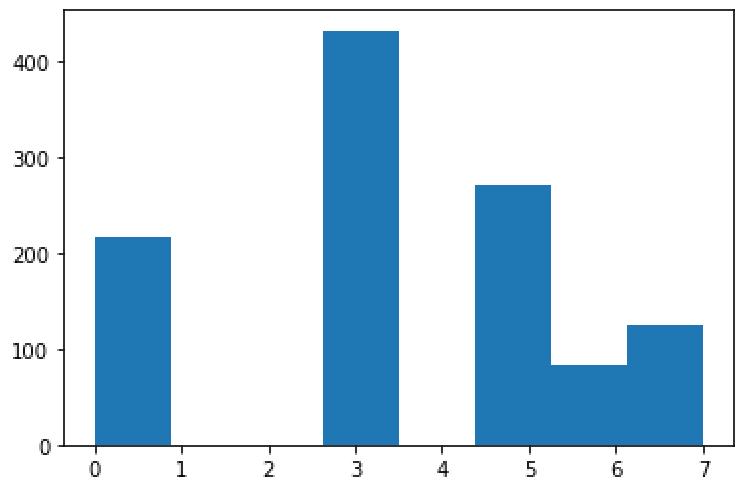

Figure 5. Frequency distribution of different classes in MRNet after pre-processing. It can be inferred that the dataset has no instances of classes 1,2, and 4, reducing it to basically five classes. Vertical axis represents frequency and horizontal axis represents class numbers.

Lastly, to fill in the vacancy of MRNet, we chose to use CIFAR100 for several reasons. First, CIFAR100 is easily available to the public which makes it easier for the readers to reproduce the results in this paper. Secondly, it requires far less memory capacity compared to Chexpert and MRNet. Also, CIFAR100 is one of the datasets used by the authors of [1] *Deep Learning on Small Datasets without Pre-Training using Cosine Loss*. The fact that we apply one of the datasets used to test the model introduced in [1] will provide a good benchmark to compare and analyze the similarity and differences between the way cosine loss is used in our paper and [1]. Another dataset that we planned on using from [1] is the Oxford Flowers dataset because the dataset satisfies our criteria of a low number of samples per class and multiple classes. Although these two aforementioned datasets are not as imbalanced as medical data, we chose them to find out if our loss had other use cases past the imbalanced small datasets.

Dataset	# Classes	# Train	# Test	Avg Samples/Class
Chexpert	*903 (total)	*Variable	*Variable	*203 (total)
MRNet	3(8)[1]	1,130	120	456(171)[1]
CIFAR100	100	50,000	10,000	500
Oxford Flowers	102	1020	6148	80

Table 2. Image dataset statistics. Avg sample/class refers to the average number of images in each class in the dataset. [1]Number outside the parentheses indicate statistics before pre-processing MRNet and number in the parenthesis indicate statistics after pre-processing *these inputs are controlled by the user.

Experiments

For our model we propose to train on a modified resnet 18 with a one channel input layer for Chexpert, and a multi-channel layer for other datasets including the Oxford Flower Dataset. The scheduler that we use is the exponential learning rate scheduler as in [7] except that this learning rate scheduler is more aggressive with a gamma of 0.9. This means that every epoch the learning rate of each parameter group is decayed by 0.9. The starting learning rate that we use is 0.002 with a weight decay of 0.00001. We also use random affine transformations on some of the experiments because they preserve collinearity and can correct for any geometric distortions or deformations that occur in the image. Additionally, affine transformations can make the model harder to fit since one picture may be transformed and the other may not, leaving the model with more work to do. With this, we could expose what happens if overfitting happens later than normal. It should also be noted of what variables we keep track of while the model is training. We measure the loss on the training examples, percent of training samples correct from the prediction, sensitivity, and specificity every 10 batches, we evaluate the loss on the test examples and percent correct on the test examples every 3 epochs, and we measure the softmax values, cosine loss, and topk loss (cosine loss + cross entropy) for every batch. Our cosine OHEM loss consists of the following:

$$Loss \ = \ L_{cross}(x, y) \ + \ \lambda \ \ L_{cos}(x, y)$$

$Top - k \ = \ k \ max(\ Loss \ )$ (k is not a variable here, it just indicates how many of the largest values should be plucked from the set)

$$k \ = \ shape(k)[0] * ratio$$

$$L_{NLL}(x', y'), \ x', y' \ = \ train \ and \ test \ examples \ whose \ indexes \ were \ chosen \ from \ Top - k$$

Below are our best results:

ratio	lambda	affine?	# of epochs	Chexpert Accuracy (%)	Flowers Accuracy (%)

0.9	-0.2	YES	70	20.54	
0.9	0	YES	70	19.56	
0.9	-0.2	NO	30		69.1
0.9	0	NO	30		70.0

Table 3: Accuracy on the training examples for certain conditions using the cosine OHEM loss where affine indicates any affine transformation, ratio and lambda represent changeable variables in cosine OHEM

ratio	lambda	affine?	# of epochs	Chexpert Accuracy (%)	Flowers Accuracy (%)
0.9	-0.2	YES	70	9.45	
0.9	0	YES	70	9.51	
0.9	-0.2	NO	30		26.1
0.9	0	NO	30		28.1

Table 4: Accuracy on the testing examples for certain conditions using the cosine OHEM loss where affine indicates any affine transformation, ratio and lambda represent changeable variables in cosine OHEM

Overall, the regular OHEM (without the added cosine loss) outperformed the cosine OHEM that we had devised. We had already known that adding cosine loss would add confidence and possibly lead to overfitting but we had hoped that in using it, not as an actual metric, but a metric to find the topk hardest losses, the cosine loss + cross entropy combination, would choose losses with more confidence to penalize by or choose losses that a confident loss would not choose (based on lambda). In other words, we were hoping that the cosine loss would behave similar to the NLL loss which is known to predict confidence by its magnitude. NLL loss is a powerful confidence metric. Unfortunately, looking at the results, there seems to be a higher training accuracy for cosine OHEM than the original OHEM (at least for the largely imbalanced dataset, Chexpert, but OHEM always outperforms cosine loss in the evaluation on the test set. This leads us to believe that our model has overfit, or just overfit earlier than the original OHEM, causing our results to be unsuccessful. The cosine loss metric appears overconfident. What else this leads us to believe is that perhaps using the max top k was an incorrect choice since we were sending losses to the output loss (NLL) that were overly confident.

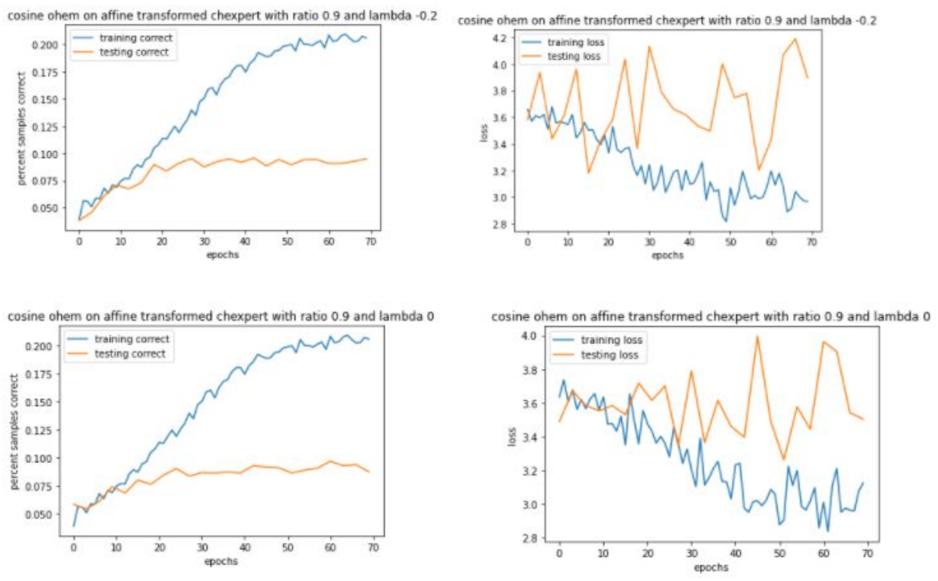

Figure 6: Accuracy curves on the train and test examples at the end of the model with a ratio of 0.9 and a lambda of -0.2 (top row) juxtaposed to a model with conditions of ratio of 0.9 and lambda of 0 (bottom row)

As Figure 6 shows, adding cosine loss to cross entropy in an OHEM loss results in a larger amount of confidence, or in other words overconfidence. For the graphs that add cosine loss the curves vary much more than those without cosine loss, with the highest change in accuracy between 3 epochs from 3.8 to 3.2 as compared to that of the OHEM without cosine loss added that has a highest change in accuracy between 3 epochs from 4.0 to 3.5. This shows that the weights, when adding cosine loss, changed dramatically as opposed to a less confident model whose weights increase steadily.

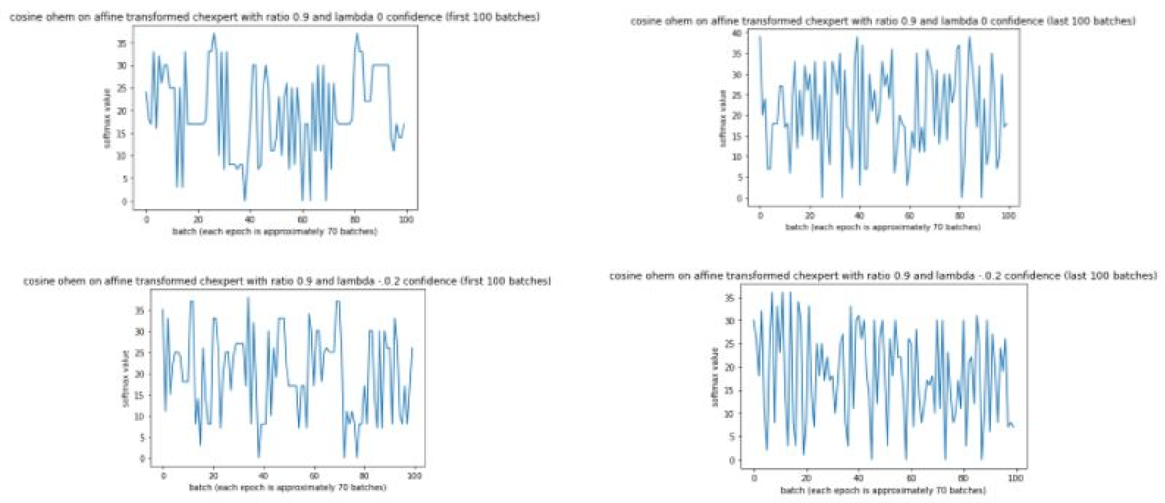

Figure 7: Values of the softmax of the last layer of the Resnet18 model for the conditions of ratio as 0.9 and lambda as 0 (top row) and ratio as 0.9 and lambda as -0.2

Furthermore, more evidence of overconfidence can be seen in the representations of the softmax layer in the model from the first 100 batches to the last 100 batches in Figure 7. When cosine loss is added to the cross entropy in OHEM the model starts with relatively rounded curves and emerges in the last 100 batches to sharp curves at every batch, making sharp increases and decreases. For example, just in the first twenty batches of the last 100 beaches there are three curves that are straight lines with at least a 32 value difference before the next line/curve.

Conclusion

In this paper we proposed a novel approach to optimize for confidence levels using a combination of Online Hard Example Mining and Cosine loss to increase classification accuracy on small datasets. We described the underlying theories of our method - such as OHEM, Cosine loss, and confidence level in deep learning - and the experiment process in detail. The results of the experimentation, however, showed that our approach does not perform better than the original OHEM. We argue that the reason for the result is that the model was overconfident on its predictions, which is indicated by the high accuracy in training phase and low accuracy in testing phase for our modified OHEM. We believe that further work can be done to refine this method by alleviating the overconfidence problem and improving the results. This could perhaps be resolved with hyperparameter optimization and cross validation, a larger number of epochs, or a different scheduler optimized to the specific problem at hand.

References

[1] B. Barz and J. Denzler. Deep Learning on Small Datasets without Pre-Training using Cosine Loss. *IEEE Winter Conference on Applications of Computer Vision (WACV) 2020.*

[2] A Shrivastava and A. Gupta and R. Girshick. Training Region-based Object Detectors with Online Hard Example Mining. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. (oral)*

[3] Jeremy Irvin,1,* Pranav Rajpurkar,1,* Michael Ko,1 Yifan Yu,1 Silviana Ciurea-Ilcus,1 Chris Chute,1 Henrik Marklund,1 Behzad Haghgoo,1 Robyn Ball,2 Katie Shpanskaya,3 Jayne Seekins,3 David A. Mong,3 Safwan S. Halabi,3 Jesse K. Sandberg,3 Ricky Jones,3 David B. Larson,3 Curtis P. Langlotz,3 Bhavik N. Patel,3 Matthew P. Lungren,3,† Andrew Y. Ng

[4] Nicholas Bien *, Pranav Rajpurkar *, Robyn L. Ball, Jeremy Irvin, Allison Park, Erik Jones, Michael Bereket, Bhavik N. Patel, Kristen W. Yeom, Katie Shpanskaya, Safwan Halabi, Evan Zucker, Gary Fanton, Derek F. Amanatullah, Christopher F. Beaulieu, Geoffrey M. Riley, Russell J. Stewart, Francis G. Blankenberg, David B. Larson, Ricky H. Jones, Curtis P. Langlotz, Andrew Y., Matthew P. Lungren. MRNet: Deep-learning-assisted diagnosis for knee magnetic resonance imaging. CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison. 21 Jan 2019

[5] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009

[6] Y. Liu , H. Li, X. Wang. Learning Deep Features via Congenerous Cosine Loss for Person Recognition. SenseTime Group Ltd., Beijing, China 2 The Chinese University of Hong Kong, New Territories, Hong Kong. 31 Mar 2017

[7] OBELISK – One Kernel to Solve Nearly Everything: Unified 3D Binary Convolutions for Image Analysis by Mattias P. Heinrich et al.