DeepRob Final Project Report: Classification Mislabeling (ClaM)

Chancellor Day
College of Engineering
University of Michigan
Ann Arbor, Michigan
dchance@umich.edu

Zack Vega
College of Engineering
University of Michigan
Ann Arbor, Michigan
zvega@umich.edu

Meha Goyal
College of Engineering
University of Michigan
Ann Arbor, Michigan
mehag@umich.edu

Tucker Moffat
College of Engineering
University of Michigan
Ann Arbor, Michigan
moffatuc@umich.edu

***Abstract*—In this paper, we aim to reproduce and extend the paper *Rethinking the Inception Architecture for Computer Vision* by Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, and Jonathon Shlens [2]. This work builds on the Inception architecture of the GoogleNet deep neural network model with the goal of optimizng the network by scaling it up while constraining the increase in computational costs. In this report, our team will reproduce and validate the model on a new dataset, and develop ClaM (Classification Mislabeling) as an attempt to extend upon the work by incrementally mislabeling parts of the input data with the goal of reducing bias and limiting overfitting. This study shows that mislabeling increasing proportions of the dataset does not improve overall performance of the Inception-v3 model defined in the original paper as evaluated on the CIFAR-10 dataset by measuring training and validation accuracy over several epochs.**

I. Introduction

This work presents a novel method for image recognition using Convolutional Neural Networks(Cnns) [1]. Recognizing and understanding its environment is crucial for a robots functionality.

Since 2012, CNN architectures have evolved rapidly becoming deeper and wider, resulting in improved performance across diverse computer vision tasks. A key landmark in the field of computer vision and Convolutional Neural Networks alike was the Alex Net victory in the 2012 ImageNet competition [2], ushering in a huge wave of research dedicated to enhancing the performance of CNNs across a broad spectrum of visual recognition tasks. Among the architectures developed in this time, Inception V3 [1], specifically, represents a pivotal milestone in efficient and effective CNNs.

The Inception architecture series is characterized by its unique utilization of inception modules. These modules enable the network to capture multi-scale features efficiently while being cost effective. Notably, the computational cost of Inception is much lower than that of VGGNet and the other more efficient models.[3] (Figure 1)(Figure 2)

Due to this feat, the Inception module has made an impact in big-data scenarios[4][5] allowing for CNNs to be used in situations that require large levels of data with little computational power.

One challenge caused by the complexity of the deep convolutions of the Inception architecture is it is difficult to modify the architecture to be used for different tasks while preserving its advantages. Specifically noted by the paper, is when scaling up the architecture, careful consideration must be used to prevent exponential increases in the computational cost and parameter count. This could render the architecture hard to use for real world algorithms.

The main conclusion from the Inception architecture is its novelty in its relatively high efficiency and accuracy for its computational cost compared to other algorithms. However at the cost of this efficiency, it becomes hard to scale up the architecture for real world applications due to risk of over

Network	Crops Evaluated	Top-1 Error	Top-5 Error
GoogLeNet [20]	10	-	9.15%
GoogLeNet [20]	144	-	7.89%
VGG [18]	-	24.4%	6.8%
BN-Inception [7]	144	22%	5.82%
PReLU [6]	10	24.27%	7.38%
PReLU [6]	-	21.59%	5.71%
Inception-v3	12	19.47%	4.48%
Inception-v3	144	**18.77%**	**4.2%**

Fig. 1. Single-model, multi-crop experimental results comparing the cumulative effects on the various contributing factors. Szegedy et. al.[1] compare their numbers with the best published single-model inference results on the ILSVRC 2012 classification benchmark.[1]

Network	Models Evaluated	Crops Evaluated	Top-1 Error	Top-5 Error
VGGNet [18]	2	-	23.7%	6.8%
GoogLeNet [20]	7	144	-	6.67%
PReLU [6]	-	-	-	4.94%
BN-Inception [7]	6	144	20.1%	4.9%
Inception-v3	4	144	**17.2%**	**3.58%***

Fig. 2. Ensemble evaluation results comparing multi-model, multi-crop reported results. Our numbers are compared with the best published ensemble inference results on the ILSVRC 2012 classification benchmark. *All results, but the top-5 ensemble result reported are on the validation set. The ensemble yielded 3.46 percent top 5 error on the validation set. [1]

fitting, lower accuracy, etc.

Our extension aims to attempt to improving upon the over fitting problem when scaling up the architecture. This extension tries to allow for the architecture to train longer with different data sets without the sever issue in over fitting.

II. Related Work

"Rethinking the Inception Architecture for Computer Vision" addresses the scaling of deep convolutional neural networks to utilize the additional computation efficiently by leveraging factorized convolutions and regularization, with an initial focus on Inception architectures [2]. The primary concern highlighted in the study is optimizing architecture designs such that convolutional neural networks can be scaled up in terms of depth and width without a significant corresponding increase in computational cost and number of parameters. The paper outlined several overall design considerations for large scale architectures for neural networks, including avoiding representational bottlenecks early in the network, using higher dimensional representations to increase network training speed, performing spatial aggregation over lower dimensional embeddings, and balancing the width and depth of the network to optimize performance and computational budget.

The paper used these design considerations as well as factorization into smaller convolutions, auxiliary classifiers, and efficient grid size reductions to propose a new architecture. The team evaluated the model on the ILSVRC-2012 validation set, and found that their highest quality Inception-v3 model reached 21.2% top 1 error and 5.6% top 5 error for single crop evaluation with a modest 2.5x increase in computational cost compared to existing models. Overall, the paper found that the scaling techniques used significantly less computation than the best models at the time. Factorizing convolutions and aggression dimension reductions helped develop high quality networks with relatively lower computational cost while the combination of low parameter count, additional regularization, batch normalized auxiliary classifiers, and label-smoothing allowed the networks to be trained on modestly sized training sets while maintaining quality.

A. Method

The main advancements this paper made to improve the architecture of the inception model can be broken down into several steps:

1) Factorizing the traditional 7x7 convolution into three 3x3 convolutions
2) Implementing a set of 3, 5, and 2 inception modules with efficient grid size reduction for filtration
3) Performing label smoothing to regularize the classifier layer of the model

A high level overview of their overall architecture can be seen in figure 3.

To implement the factorization of the convolutions, the paper introduced an interesting conversion method, finding that any $n \times n$ convolution could be replaced by first a $1 \times n$ convolution followed by an $n \times 1$ convolution, such that the computational cost savings increase as n grows. From their experiments, they found that this worked well on medium sized grids, such as the 7x7 grid used for the model. An example of this factorization is depicted in figure 4 below.

type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

Fig. 3. The outline of the proposed network architecture. The output size of each module is the input size of the next one.

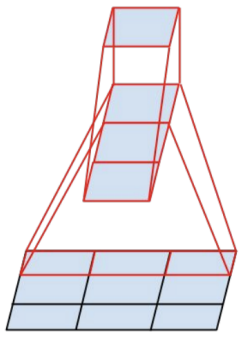

Fig. 4. The figure demonstrates an example of a mini network in which a 3x3 convolution is factorized. The lower layer is comprised of a 3x1 convolution with 3 output units

B. Reproducing Results

To reproduce these results we utilized the Tensorflow and Keras python modules to recreate the environment to train the Inception architecture. Due to hardware limitations we had to use the Cifar-10 Dataset, a relatively smaller dataset compared to the massive ImageNet dataset used by the paper. It is important to keep this in mind as the Cifar-10 Dataset has only 10 labels and a relatively small size compared to the 20,000 labels of ImageNet that the original Inception v3 architecture is trained on. To prepare the Cifar-10 dataset for training with

Inception V3, pre-processing steps are undertaken. Initially the images are resized to meet the minimum input size of Inception V3, which is 75x75 pixels. TensorFlow's image manipulation library and the skimage model are then used to transform the images into the proper format. After this, the images are then normalized to ensure pixel values lie between -1 and 1 as well as the labels being converted to one-hot encoding. Using Keras, the Inception V3 model, pre-trained on the ImageNet dataset, is employed as a base model. The fully connected layers of the Inception V3 model are then addapted to match the number of classes in the CIFAR-10 dataset. Specifically, a Global Average Pooling layer followed by two dense layers with ReLU and soft max activation's are added to the model. We then trained using this to recreate what was done in the paper.

III. Algorithmic Extension

A. The Problem

After reproducing the paper's results using the CIFAR-10 dataset, we found that the model was overfitting significantly, and at epochs 3 and 4, the training accuracy increased while the validation accuracy began decreasing. Therefore, for our extension, we aimed to reduce the overfitting by randomly mislabeling training data before each attempt to train the model. Since the Inception v3 model already included bias, modifying the training set specifically of the labels allowed us to introduce noise and attempt to reduce overfitting overall.

B. Dataset

The dataset that we used to test our model and extension was the CIFAR-10 dataset. The dataset contains 60,000 32x32 images with 10 classes and 6,000 images per class. Due to the limited access to the original dataset, and limits on our own systems' computational power and memory, we chose this dataset as despite the more limited dataset, it provided similar information and was able to be processed efficiently on our systems.

C. Implementation

We began by writing our own python script to access and use the Inception V3 model described in the paper. We cut of the top of the model archtiecture and replcaed it with our own, and set it up to use the 10 classes as represented in our CIFAR-10 dataset rather than the large number of classes from the paper's original dataset. Then, we performed two experiments. First we randomly mislabeled 1, 5, and 10% of the y-train data (the labels used for training) only once before training the model, and for the second approach, we had the model mislabel the same percentages of data before every epoch. We graphed the changes in training accuracy, validation accuracy, and validation loss per percentage of mislabeled data per epoch.

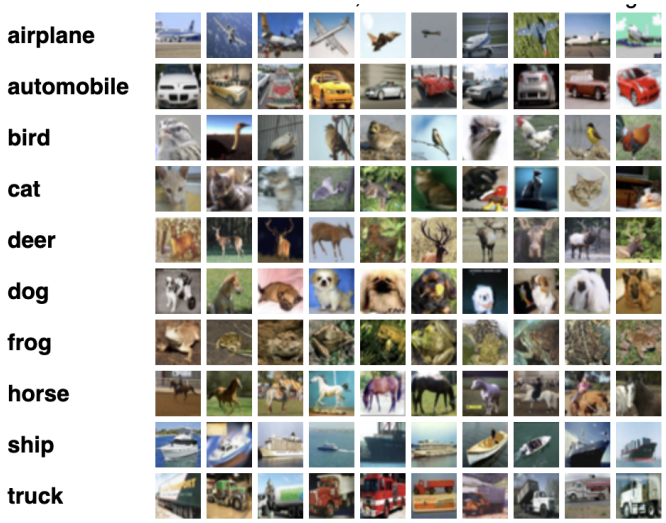

Fig. 5. This image displays each of the 10 classes and 10 example images from each class of the CIFAR-10 dataset

IV. Experiments and Results

When recreating the paper, it was noticed that while using the CIFAR-10 dataset that the data was being over fitted. At epoch 3 or 4, the validation accuracy would start decreasing, while the training accuracy would continue to increase. In an attempt to combat this over-fitting, and since the inception V3 architecture already has bias included in their model, it was decided to implement a novel form of noise involving randomly mislabeling training data. We implemented this in two distinct ways. The first way that was tested was randomly mislabeling a percentage of the dataset, and then training all 10 epochs on that data with a portion mislabeled. The second way that was tested was randomly mislabeling a percentage of the dataset, and then training one epoch on that data. This mislabeling process was repeated between every epoch so the same percentage of data was mislabeled, but different sets of data was mislabeled each epoch.

A. Experimental Setup

For this paper, all software was run utilizing an RTX 3060 Ti on Ubuntu 22.04. Due to hardware limitations, when recreating the paper *Rethinking the Inception Architecture for Computer Vision*[2] instead of training the entire inception V3 model on the imageNet dataset, we imported the pre-trained model without the top, and instantiated our own top. This top was identical to *Rethinking the Inception Architecture for Computer Vision*[2] except it had 10 output classes, one for each class in the CIFAR-10 dataset. This model was trained using a batch size of 32, equal to *Rethinking the Inception Architecture for Computer Vision*, and 10 epochs, which is different than the 100 that *Rethinking the Inception Architecture for Computer Vision* used, but much quicker.

B. Results

Each 10 Epoch trial took $\sim$18 minutes. The resulting graphs were produced in matlab.

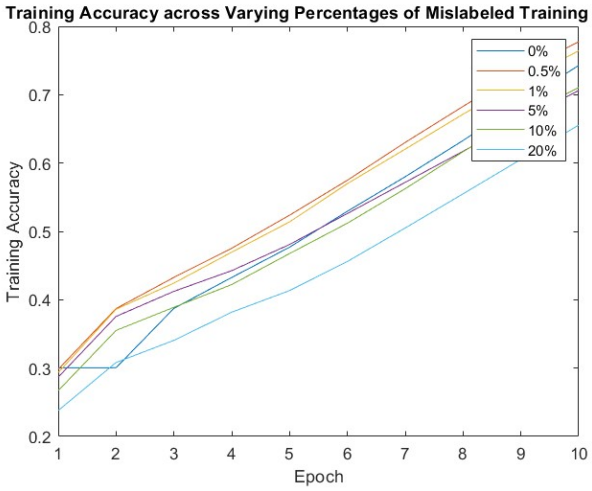

Fig. 6. This graph displays the training accuracy per epoch for method one, mislabeling a percentage of the training data before training starts

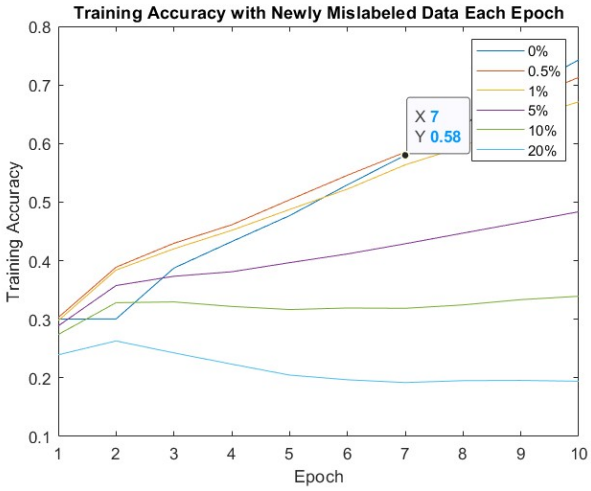

Fig. 8. This graph displays the training accuracy per epoch for method two, mislabeling a percentage of the training data before each new epoch

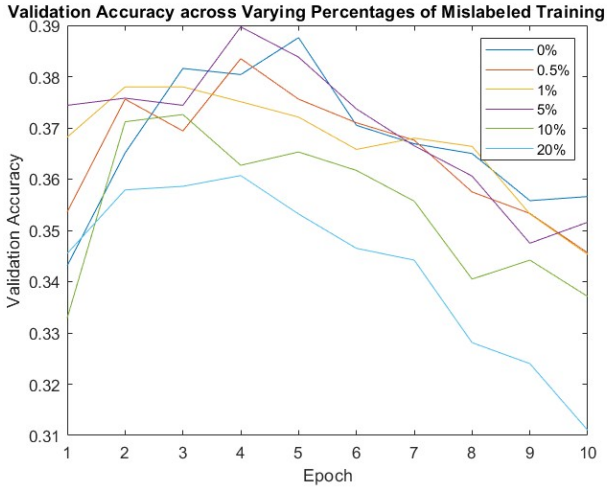

Fig. 7. This graph displays the validation accuracy per epoch for method one, mislabeling a percentage of the training data before training starts

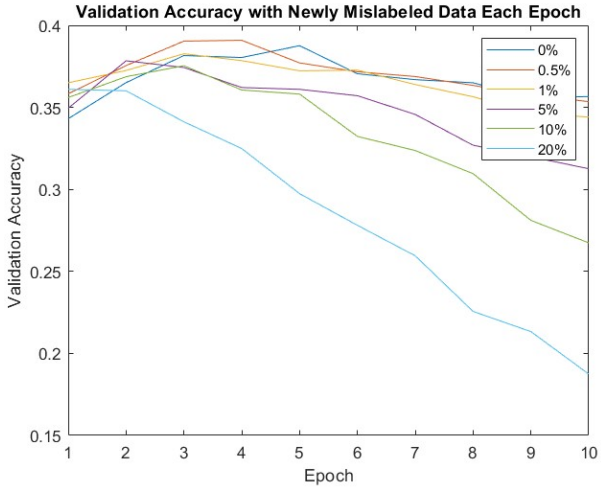

Fig. 9. This graph displays the validation accuracy per epoch for method two, mislabeling a percentage of the training data before each new epoch

Fig 6. shows that our classification using inception V3 is successfully being trained on the training data. Moreover low amounts of initial mislabeling actually increased the training accuracy over zero mislabeling. High amounts of training data mislabeling decreased accuracy.

Despite Fig 6. showing that the model was being trained on the initially mislabeled training data. Fig 7. shows there is a large amount of over-fitting occurring. after the 3rd or 4th epoch of every trial, validation accuracy starts decreasing.

Fig 8. shows that training with a low amount of mislabeling each epoch allows for accurate training than the baseline of 0%, but training with a high amount of mislabeling each epoch does not allow for accurate training. In comparison to method 1, training accuracy in method two is worse, and having a low amount of mislabeling each epoch does not increase training accuracy. Our hope was that this lower training accuracy in method two would have the trade off of introducing noise into the system to reduce over-fitting.

Shown in figure 9. the validation results of method two are relatively similar to method one, although all validation accuracy starts higher at epoch one, and has a more gradual decrease in validation accuracy over each epoch. This shows that neither method one or method two successfully introduced a successful form of noise preventing data over-fitting.

V. Conclusions

This paper found that classifying a small dataset with inception V3 resulted in over-fitting. Moreover, two different methods of mislabeling data were explored to decrease this over-fitting. it was found that low levels of initial mislabeling of the data increased training accuracy, but both methods did

not increase validation accuracy, nor did they decrease overfitting.

REFERENCES

[1] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 2818-2826, doi: 10.1109/CVPR.2016.308.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, and Y. Jia, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2016. [Online]. Available: https://www.tensorflow.org/

[3] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. arXiv preprint arXiv:1502.01852, 2015.

[4] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pages 2155–2162. IEEE, 2014

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.