Computer Vision Fall 2018 Project - Kaggle Inclusive Images Challenge

*Aishwarya Budhkar
Courant Institute of Mathematical Sciences
New York University
asb862@nyu.edu

*Radhika Nikam
Courant Institute of Mathematical Sciences
New York University
rn1334@nyu.edu

Abstract

Modern convolutional networks are not known to be robust against distributional skew, i.e. generalizing over different distributions in testing vs. training. However, in real world systems, training data is not inclusive enough of all possible scenarios, and our prediction models usually tend to fail. In this project, we attempt to try out approaches to train classifiers to obtain good results over different geographical distributions.

1. Introduction

Performance of machine learning models is known to be greatly influenced by the specifics of the data-set that it is trained on. A Convolutional Neural Network (CNN) is only capable of identifying features it has already seen in the data-set it trained on, and is usually unreliable at generalizing to different locations. This is a common problem of CNNs [4] and is a major research area today.

One area of interest where we would like neural networks to generalize especially well is across geographical distributions. It is understood that a doctor's clinic in the United States or in Europe is unlikely to resemble a doctor's clinic in India or in Africa, proving to be a problem for our classifiers' predictions. The weights that the classifier will learn for the clinic in the US will be very different from those it would learn from a clinic in Africa, and hence it would have a hard time predicting "clinic" for the latter. To tackle this problem, we need to be able to model our classifiers to perform well on geographical stress tests.

The data-set and problem for this project were created by Google AI, with the aim of encouraging participants to build models that can train on a given data-set, but perform well on test images drawn from an unknown or very different geographical distribution as well. This makes sure the model is robust to blind spots in the training data and be able to get deployed to the benefit of all communities across the globe.This challenge was posted on the Data Science competitive platform Kaggle, and was part of the NIPS 2018 competition track.

1.1. Dataset Description

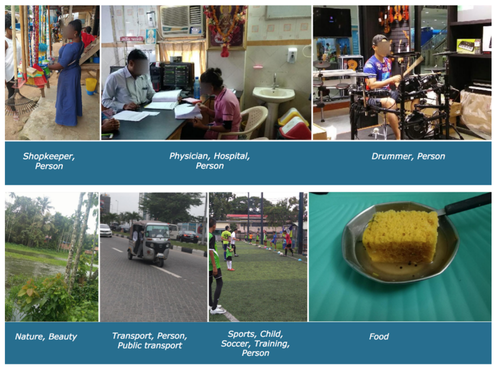
Shopkeeper, Person
Physician, Hospital, Person
Drummer, Person
Nature, Beauty
Transport, Person, Public transport
Sports, Child, Soccer, Training, Person
Food

Figure 1. Examples of Images in the Dataset

The challenge uses a portion of the OpenImages dataset as the training set. The rules of the competition state that no new images can be added while training, but Wikipedia text data may be used to augment the images, if needed.
The images in the OpenImages training dataset includes 1,743,042 training images. Each image in the mix has between 1-10 labels each, and there are 7178 classes in total. There is a definite hierarchy of classes that can be exploited during training for bounding boxes, as shown here. Examples of labels and their descriptions are given in the table below.

Label	Description	Count
/m/01g317	Person	839436
/m/09j2d	Clothing	675650
/m/05s2s	Plant	436288
/m/07j7r	Tree	423757
/m/0dzct	Human face	387233

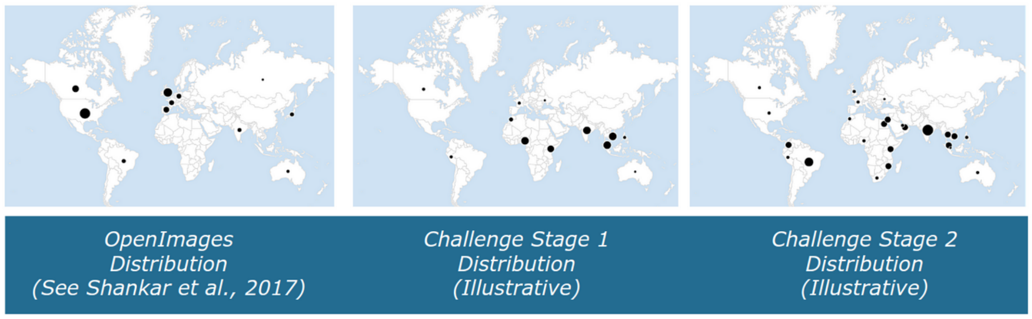
OpenImages Distribution (See Shankar et al., 2017)
Challenge Stage 1 Distribution (Illustrative)
Challenge Stage 2 Distribution (Illustrative)

Figure 2. Geographical distribution of images on the OpenImages dataset is shown in the first panel. In comparison, the next two panels show the image distribution of the test datasets, showing a concentration of images from areas rarely visible in the OpenImages Dataset.

We studied the distribution of the dataset to figure out an effective approach to solving the problem. This involved understanding how the images were distributed across labels and which were the most common labels as seen in the training dataset as opposed to the test dataset.

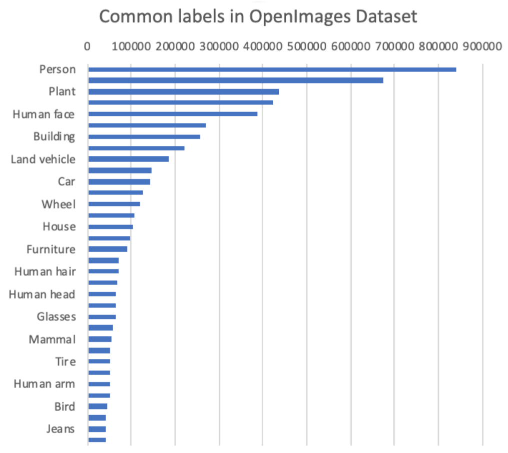

Figure 3. Most common training labels

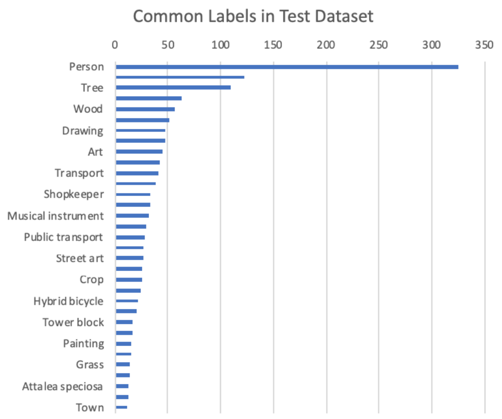

Figure 4. Most common testing labels

1.2. Issues in the Data-set

One major issue in the Dataset was that it was highly unbalanced, i.e. Most of the generic labels- humans, human hair, plant etc. are over-represented in the training set. These labels are not always very useful in capturing geographical nuances across the world. Much of the information regarding a scene is stored in the often rarer labels. For example, a doctor's clinic anywhere would have a few basic elements that would be common across all regions, a stethoscope for example. Because of the heavy dependence of the model on the major labels however, like person and clothing, it is easy for it to not look for or identify stethoscopes in the picture. Hence, the real distinguishing labels such as appliances and environmental cues need to be given much more priority than given currently.
Another issue is that since the distributions among the training and test images are so different, there are cases where the labels in the test images do not have any input training labels at all. Since this is a stress test, there needs to be a way for the model to be immune to such discrepancies and still be able to predict the correct labels regardless of them. As shown in the further sections, we aim to tackle the problems in the dataset and come up with a model that is robust to these blind spots, and compare performance with baseline implementations.

2. Data Preprocessing

For the baseline implementation, we added in data augmentation to the images, introducing scaling, randomized cropping and flipping. This did not get us the results we wanted. Post doing some exploratory data analysis on the images, we decided to go ahead with weighted sampling [3], an importance sampling technique that is a variant of stratified sampling and is used to train on data more efficiently.

In essence, weighted sampling takes an input and scales it by the weights that it has been given. Then the sum of the weights is divided by the number of classes, and these equal partitions are used to sample from randomly, and choose the input it aligns to in the original distribution. This fact is shown in the image below, where the first block shows inputs in boxes, and each box is scaled according to its weight. The samples chosen from the figure below it are lined up with the top figure and those samples are chosen.

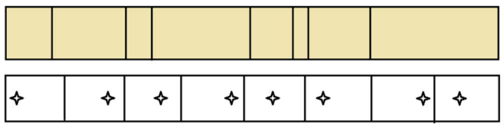

Figure 5. Weighted Sampling. *Image courtesy https://www.sebastiansylvan.com/post/importancesampling/*

In our training data-set, we were able to determine which class labels were rare and how they translated to the classes that would generalize over geographical distributions. All classes were weighted according to their counts over images in the data-set, and the weights of the rarer classes were scaled up. In numbers, this meant that the portion of the dataset that we were able to train on was around 20,000 post sampling. This is due to resource constraints of downloading and storing a large dataset and running them all through a weighted sampler. We were able to sample on a 100,000 images from the dataset. This approach alone, though gave us a big boost in the F2-scores.

In essence, there are 6 classes which are present in the test tuning labels, but are absent in the OpenImages dataset. These labels were ignored for the purposes of tuning on the test dataset.

3. Method

We first performed a weighted sampling on 100,000 images in dataset and created a sampled dataset of 20,000 images to make training labels more balanced, and trained on a subset of 200 labels. These labels were multi-hot encoded. For pre-processing, we scaled the training batch images to 224×224 and 100×100, randomly flipped left or right and cropped, training on a standard ResNet and DenseNet multi-label classifier. We used the sigmoid activation function (with cross entropy loss) in the last layer of the model to express the probability of the image belonging to a particular class.
To account for multilabel classification, probabilities were rounded to get correct prediction and calculate the accuracy. Training was performed in batches of 128 for 20K images for 50 epochs, and the model trained for around 6 hours on every experiment. We used momentum with a learning rate of 0.01, and accuracy and F2- score metrics were used for evaluation. Here F2-score is used for evaluation which is a weighted average of recall and precision, and gives more accurate results for uneven class distribution than accuracy. The model was tested on stage 1 test images on the challenge.
A summary of our experiments can be found in the appendix.

4. Results

We were able to experiment over multiple baselines for the problem. Our final results, trained over 50 epochs are shown below. Training on 50000 images took around 2 days, though tuning hyperparameters didnt improve performance to a great extent. After weighted sampling there was a huge boost in performance with f2 score rising to 0.094 from 0.032. Accuracy here is high, but it might not be a reliable measure since the model is very easily able to predict labels like ’Person’ and ’Clothing’, which are very generic labels and are not as useful in solving the problem.

Also given below are the plots for the accuracy and loss for ResNet and DenseNet with weighted sampling.

Model	Epochs	Accuracy(%)	F2-score
ResNet	50	91.61	0.032
DenseNet	50	93.82	0.028
ResNet+WS	50	98.07	0.093
DenseNet+WS	50	97.8	**0.094**

Table 1. Classification Results

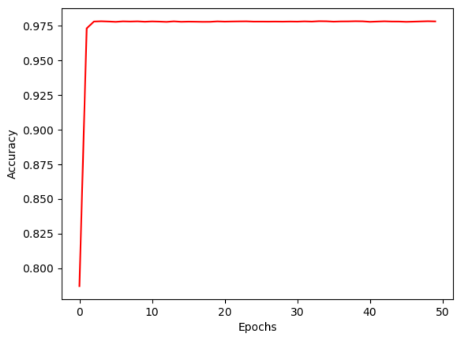

Figure 6. ResNet+WS - **Accuracy**

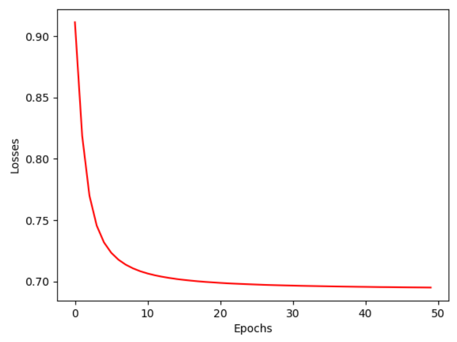

Figure 7. ResNet+WS - **Loss**

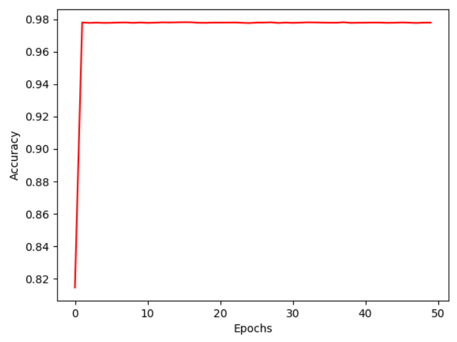

Figure 8. DenseNet+WS - **Accuracy**

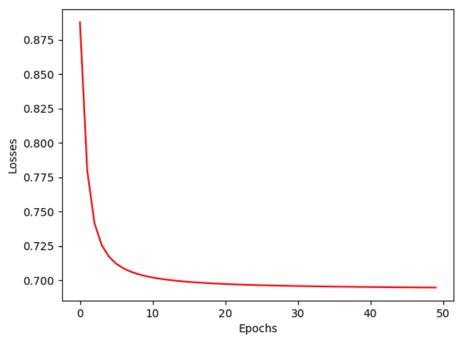

Figure 9. DenseNet+WS - **Loss**

5. Conclusion

We were able to run the problem by baselines and try a variation of importance sampling. However, given the constraints of the size of the dataset and processing and time limitations, we were able to train on a subset of the actual OpenImages Dataset and achieve an F2-score of 0.094. In comparison, the top scores on the Kaggle Leaderboard reach 0.3 after two weeks of training the model on GPUs and on the entire dataset.
There are many variations in the approaches that can be tried, and we would like to continue work on it by including more ablation studies and ensembling techniques. An approach that could be used is using Wikipedia text data to generate word embeddings on the labels, and predict them as well. Another technique, used by fast.ai, is progressive re-sizing [1], which makes the model see smaller images at the beginnning of training, but as the model learns, it is shown bigger images so that it learns more fine-grained distinction. For this problem, it seems the devil is in image preprocessing details before attempting changes to loss functions or modelling techniques.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[2] J. Howard. Now anyone can train imagenet in 18 minutes.

[3] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[4] S. Shankar, Y. Halpern, E. Breck, J. Atwood, J. Wilson, and D. Sculley. No classification without representation: Assessing geodiversity issues in open data sets for the developing world. In *NIPS 2017 workshop: Machine Learning for the Developing World*, 2017.

[5] S. Sylvan. Efficient weighted sampling.

[4] [5] [2] [3] [1]

A. Experiment summary

Model	Data Size	Image Size	Epochs	Learning Rate	Optimizer	Accuracy (%)	F2-score
ResNet	50000	100x100	20	0.01	Momentum	78.61	0.025
ResNet	20000	100x100	20	0.1	Adam	75.12	0.020
ResNet	20000	224x224	20	0.1	Momentum	83.24	0.029
ResNet	20000	224x224	50	0.01	Momentum	91.61	0.032
DenseNet	50000	100x100	20	0.01	Momentum	76.43	0.024
DenseNet	20000	224x224	20	0.1	Adam	76.2	0.019
DenseNet	20000	100x100	20	0.1	Momentum	80.12	0.022
DenseNet	20000	100x100	50	0.01	Momentum	93.82	0.031
ResNet + WS	20000	224x224	20	0.01	Momentum	97.1	0.093
ResNet + WS	20000	224x224	50	0.01	Momentum	98.07	0.093
DenseNet + WS	20000	100x100	20	0.01	Momentum	97.5	0.093
DenseNet + WS	20000	100x100	50	0.01	Momentum	97.8	0.094

Table 2. Summary of Experiments