Multi-Label Classification

Willie Maddox

Abstract—Machine Learning Engineer Nanodegree Capstone Project

Index Terms—Deep Learning, Machine Learning, Neural Networks, Keras, Tensorflow, VGG16, MS-COCO, NUS-WIDE.

1 Introduction

This demo file is intended to serve as a "starter file" for IEEE Computer Society journal papers produced under LaTeX using IEEEtran.cls version 1.8 and later. I wish you the best of success.

mds
December 27, 2012

2 Definition

2.1 Project Overview

In machine learning, image classification is the problem of deciding to which category a particular image belongs or to which categories objects in an image belong. Traditional classifiers, such as binary and multi-class classifiers return as output only one value, the prediction, whereas multilabel classifiers produce a vector of output values. Much of the previous work for solving image classification problems has been focused on using a single label per image to train a classifier. For some of the more popular datasets such as ImageNet [1] and CIFAR-10 [2], each image has associated with it a single label and there are many images per label. This works well for a dataset like MNIST [3] where each instance is a black and white image of a single handwritten digit between 0 and 9. But for images that illustrate the real world such as photographs, there is almost never a single contextual topic in the image. For example, Fig.1 is a picture of bolts, but there are also nuts, washers and a wooden table. So in reality this image has (at least) four tags. The goal of this project is to create a multi-label classifier that can be trained using images that have multiple labels per image.

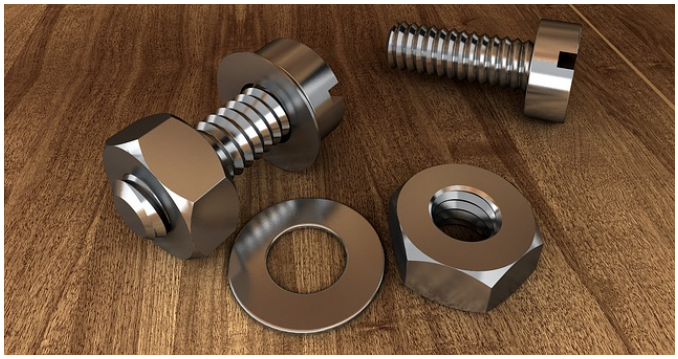

Fig. 1. Nuts n Bolts

2.2 Problem Statement

Classifiers such as AlexNet and VGGNet which are typically trained on single class image data such as the 1000 class ImageNet dataset are also able to classify multiple objects in images even though they weren't specifically trained to do so. This happens because the output of the classifier is simply a list of probabilities for every possible class label. These probabilities range from zero to one and the probability whos value is closest to one is typically regarded as the *winning* prediction. However, if we take the top 3 or even the top 5 highest probable labels we will often find that these *runner-up* predictions can also be found in the image.

• *W. Maddox is currently not affiliated with an academic institution.*

Manuscript received November 21, 2017; revised November 26, 2017.

The problem that we will try to solve is finding multiple features (or labels) in a single image. We would also like to know how much better our trained multi-label classifier is at predicting the top k labels for a given image as compared to a standard single-label classifiers like VGG16 [4] trained on ImageNet? Can we train a classifier using multi-labelled image data to perform at least as well as existing models trained on single-class image data. A model trained on multi-labelled image data should, in theory, require fewer data samples since more information per image is avaliable to training. Is this also true? Let's find out.

2.3 Metrics

After reading through previous literature on multi-label classification, we found that there are quite a few metrics appropriate for this problem. The mean average precision (mAP) is a widely used metric for comparing between trained models and has been regarded as the best metric for classification problems [5]. Other popular metrics include precision, recall, F_1 score, Jaccard index, 0/1 loss and Hamming loss [6], [7], [8].

For multilabel classification, particularly, the precision, recall, and F_1 score have three different variants which we will use [9], [10], [11], [12]. *Macro-averaging* measures the average classification performance across labels.

$$\mathrm{MP} = \frac{1}{L}\sum_{j=1}^{L}\frac{\sum_{i=1}^{N} y_{ij}\hat{y}_{ij}}{\sum_{i=1}^{N} y_{ij}} \tag{1}$$

$$\mathrm{MR} = \frac{1}{L}\sum_{j=1}^{L}\frac{\sum_{i=1}^{N} y_{ij}\hat{y}_{ij}}{\sum_{i=1}^{N} \hat{y}_{ij}} \tag{2}$$

$$\mathrm{MF}_1 = \frac{1}{L}\sum_{j=1}^{L}\frac{2\sum_{i=1}^{N} y_{ij}\hat{y}_{ij}}{\sum_{i=1}^{N} \hat{y}_{ij} + \sum_{i=1}^{m} y_{ij}} \tag{3}$$

This metric treats all classes equal regardless of their sample size, so focusing on getting rare classes right can result in a significant increase in performance. To counterbalance this bias we also perform *instance-averaging* which measures average classification performance across examples,

$$\mathrm{iP} = \frac{1}{N}\sum_{i=1}^{N}\frac{\sum_{j=1}^{L} y_{ij}\hat{y}_{ij}}{\sum_{j=1}^{L} y_{ij}} \tag{4}$$

$$\mathrm{iR} = \frac{1}{N}\sum_{i=1}^{N}\frac{\sum_{j=1}^{L} y_{ij}\hat{y}_{ij}}{\sum_{j=1}^{L} \hat{y}_{ij}} \tag{5}$$

$$\mathrm{iF}_1 = \frac{1}{N}\sum_{i=1}^{N}\frac{2\sum_{j=1}^{L} y_{ij}\hat{y}_{ij}}{\sum_{j=1}^{L} \hat{y}_{ij} + \sum_{j=1}^{l} y_{ij}} \tag{6}$$

and *micro-averaging* which measures average classification performance across both labels and samples.

$$\mu\mathrm{P} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{L} y_{ij}\hat{y}_{ij}}{\sum_{i=1}^{N}\sum_{j=1}^{L} y_{ij}} \tag{7}$$

$$\mu\mathrm{R} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{L} y_{ij}\hat{y}_{ij}}{\sum_{i=1}^{N}\sum_{j=1}^{L} \hat{y}_{ij}} \tag{8}$$

$$\mu\mathrm{F}_1 = \frac{2\sum_{i=1}^{N}\sum_{j=1}^{L} y_{ij}\hat{y}_{ij}}{\sum_{i=1}^{N}\sum_{j=1}^{L} \hat{y}_{ij} + \sum_{i=1}^{N}\sum_{j=1}^{L} y_{ij}} \tag{9}$$

For both of these, the more frequent classes will be dominant and have a greater impact on performance.

Equations 1 through 9 require the $\hat{y}_{ij}$ values to be binary 1/0 predictions. However, most classifier models including ours return an output of predictions in the form of floating point values on the interval $[0, 1]$. The process of turning these raw predictions into binary predictions is commonly referred to as *label decision* and there are two common approaches to this type of decision: top-k and thresholding. [13]

In the top-k approach, for each sample, the k labels with the highest prediction value are set to 1 and the rest are set to 0. This approach works very well when working with datasets that have nice evenly distributed labels across samples but less so when working with unbalanced datasets. One variation of the top-k approach is to use a *per-sample* top-k, the value of which is determined by the number of ground truth labels in each sample. For example, if a sample has 3 ground truth labels, then we assign a 1 to the top 3 predictions from that sample and 0 to the rest of the predictions. The next sample might only have 1 ground truth label in which case only the highest predicted

value will be set to 1. And so on for the rest of the dataset. We have not yet seen this approach in the literature but think it might be interesting to investigate.

The second type of label decision is thresholding. Using this approach the label is predicted as present if the raw prediction exceeds some predefined threshold τ.

$$\hat{y}_{ij} = \begin{cases} 1 & \text{if} \quad h_{ij} \geq \tau \\ 0 & \text{if} \quad h_{ij} < \tau \end{cases} \tag{10}$$

Typically this value is set to 0.5 and is a logical choice for multiclass classification where the raw predictions come from the output of a softmax layer. For multilabel classification it is more common to see a sigmoid function used for the final layer, hence the separation between positive and negative predictions becomes less obvious. To this end, we choose a threshold that minimizes the difference in label cardinality between the ground truth labels y_{ij} and the predictions $\hat{y}_{ij}$. [14]

$$\text{LCard} = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{L} y_{ij} \tag{11}$$

$$\tau = \text{argmin} \left\| \left(\text{LCard} - \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{L} 1_{h_{ij} \geq \tau} \right) \right\| \tag{12}$$

When comparing our model to the benchmark models, we will use Eqns 1 through 9. When comparing the results of our model to those previously reported in the literature, we will restrict ourselves to using only those metrics for which there is a direct comparison to the previous work.

3 Analysis

3.1 Data Exploration

For this study we will use the (2017) MS-COCO [15] dataset as our primary dataset. The COCO dataset is freely downloadable from the internet and contains images with multiple labels per image making them ideal for this kind of study. The COCO dataset contains a training set of 118,287 images and a validation set of 5,000 images. The images are colored and the size of each image is about 600 * 400 pixels. The MS-COCO dataset has 80 labels with 2.9 labels per image on average. However, both training and validation sets have images with no labels. This is not so much a problem for training as it is for testing. Specifically, when we evaluate our predictions using the metrics from Section 2.3, empty labeled samples could lead to divide by zero errors. Hence, we will only use labelled samples for testing.

3.2 Exploratory Visualization

In Fig. 2, we can see that the dominating class label is *person* with 64115 occurances in the training set and 2693 in the validation set. The least dominating class labels are the *toaster* and *hair drier* with 217 and 189 in the training set and 8 and 9 in the validation set, respectively. In Fig. 3, we show the number of unique tags per sample. The majority of the samples have at least 1 to 4 unique labels with 2 unique labels being the most dominate. We also notice that the training and testing sets have 1021 and 48 samples with no labels. We will keep this fact in mind when running our evaluation metrics. In both figures, the training set has 20 to 30 times more samples per label than the validation set and the ratios between the training and validation sets are consistent (i.e. the heights of the train and valid bars are about the same.)

3.3 Algorithms and Techniques

To solve this problem, we will model our classifier using a deep convolutional neural net. We will not train our model from scratch, but rather we will perform transfer learning using the VGG16 convnet trained on imagenet. We will read the images from disk and perform data augmentation using worker threads. This will make it so the GPU doesn't have to wait on batches thereby maximizing our training efficiency. The final output of the trained model will be a sigmoid layer. We will use binary cross entropy for our loss function [16] and mini-batch stochastic gradient descent with momentum for backpropogation. Table 3.3 outlines the hyperparameters that will be tuned to optimize the classifier. We will also use some Keras callbacks during our training. ModelCheckpoint

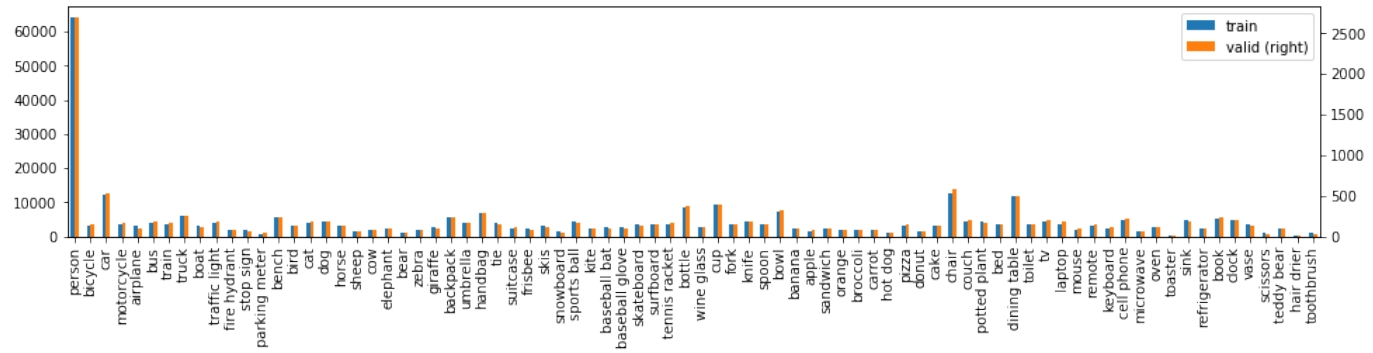

Fig. 2. Number of samples containing at least one instance of a label.

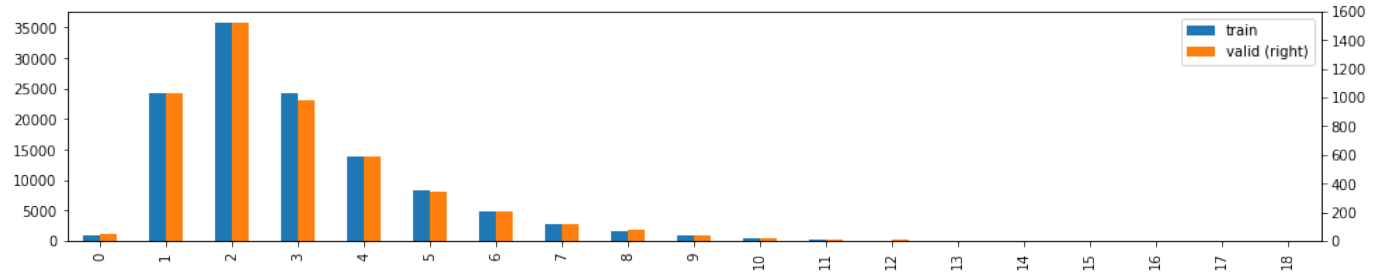

Fig. 3. Number of unique labels per sample.

TABLE 1
List of hyperparameters to tune to optimize the classifier. Stage 1 is the image preprocessing stage. inet: use a global scaling factor, zmuv: use per-image zero-mean unit-variance. Stage 2 is where the top model gets trained with bottleneck features. BN stands for Batch Normalization. We train the entire model (base + top) in Stage 3. Please refer to Sec. 4 for more details.

Stage	Hyper-parameter	Variants
1	Image Preprocessing	inet, zmuv
2	Classifier Design	77, avg, max
2	Regularization Method	BN, Dropout
2	Number of Hidden FC Layers	0, 1, 2
2	Number of Nodes per Layer	1024, 2048, 4096
2	Learning Rate	
2	Batch Size	32, 64, 128, 256
3	Number of Frozen Layers	15, All
3	Learning Rate	
3	Batch Size	32, 64, 128, 256
3	Data Augmentation	h, w, r, s, z, f

to save only the best models at each epoch, EarlyStopping to stop training when we stop making progress and CSVLogger for plotting purposes.

3.4 Benchmark

I was unable to find a VGG16 model trained on COCO to use for benchmarking purposes. I was however able to find multiple results of multilabel classifiers trained on COCO. I will use these results when I evaluate the performance of my model in 5.1.

4 Methodology

4.1 Data Preprocessing

There are 2 types of preprocessing we do on the image data. The first is image resizing and normalizing. Before passing any image through the network it must first be resized to match the height and width of the input layer and its values must be converted (normalized) from uint8's to float32's in the range of -1 and 1. The method by which the images are normalized became a tunable hyperparameter. Keras comes shipped with a preprocess_image[1] func-

1. https://github.com/keras-team/keras/blob/master/keras/applications/imagenet_utils.py

tion that is intended to be used for data normalization on VGG16. If using Keras with the Tensorflow backend this normalization boils down to dividing by 127.5 and then subtracting by 1 for every image. As an alternate option, the various flavors of model generators in Keras provide to option to normalize images by either featurewise or samplewise zero-mean unit-variance (or ZMUV). These three normalization methods each provide slightly different results and so we leave the decision as to which one is best as a tunable hyperparameter.[2]

The other type of image preprocessing involves various forms of data augmentation. Going by the codes in Table 3.3 we will augment the image data by (h) shifting along the height direction, (w) shift along the width direction, (r) performing random rotation, (s) skew transform, (z) multiplying by a zoom factor, and (f) performing horizontal flips. We will also shuffle the training data after each epoch. We will only perform data augmentation during training, not during top model training as the input images to the top model aren't images but rather features of a deep layer.

4.2 Implementation

In this section, the process for which metrics, algorithms, and techniques that you implemented for the given data will need to be clearly documented. It should be abundantly clear how the implementation was carried out, and discussion should be made regarding any complications that occurred during this process. Questions to ask yourself when writing this section:

Is it made clear how the algorithms and techniques were implemented with the given datasets or input data?

Were there any complications with the original metrics or techniques that required changing prior to acquiring a solution?

Was there any part of the coding process (e.g., writing complicated functions) that should be documented?

We will load the VGG16 model but without the last 3 fully connected layers. These layers are specific to the 1000 ImageNet classes that the VGG16 model was originally trained on. N

4.3 Refinement

In this section, you will need to discuss the process of improvement you made upon the algorithms and techniques you used in your implementation. For example, adjusting parameters for certain models to acquire improved solutions would fall under the refinement category. Your initial and final solutions should be reported, as well as any significant intermediate results as necessary. Questions to ask yourself when writing this section:

Has an initial solution been found and clearly reported?

Is the process of improvement clearly documented, such as what techniques were used?

Are intermediate and final solutions clearly reported as the process is improved?

5 Results

5.1 Model Evaluation and Validation

In this section, the final model and any supporting qualities should be evaluated in detail. It should be clear how the final model was derived and why this model was chosen. In addition, some type of analysis should be used to validate the robustness of this model and its solution, such as manipulating the input data or environment to see how the models solution is affected (this is called sensitivity analysis). Questions to ask yourself when writing this section:

Is the final model reasonable and aligning with solution expectations? Are the final parameters of the model appropriate?

Has the final model been tested with various inputs to evaluate whether the model generalizes well to unseen data?

Is the model robust enough for the problem? Do small perturbations (changes) in training data or the input space greatly affect the results?

Can results found from the model be trusted?

2. This part of the study raised an interesting question. Should the input image normalization method be part of the (or be supplied as) supplemental information to the saved model? In other words, if I download and use a trained model, should I not also expect the model to somehow tell me in what format the model expects the input data to be? There seemed to be conflicting opinions as to which normalizing method should be used.

5.2 Justification

In this section, your models final solution and its results should be compared to the benchmark you established earlier in the project using some type of statistical analysis. You should also justify whether these results and the solution are significant enough to have solved the problem posed in the project. Questions to ask yourself when writing this section:

Are the final results found stronger than the benchmark result reported earlier?

Have you thoroughly analyzed and discussed the final solution?

Is the final solution significant enough to have solved the problem?

6 Conclusion

6.1 Free-Form Visualization

In this section, you will need to provide some form of visualization that emphasizes an important quality about the project. It is much more free-form, but should reasonably support a significant result or characteristic about the problem that you want to discuss. Questions to ask yourself when writing this section:

Have you visualized a relevant or important quality about the problem, dataset, input data, or results?

Is the visualization thoroughly analyzed and discussed?

If a plot is provided, are the axes, title, and datum clearly defined?

6.2 Reflection

In this section, you will summarize the entire end-to-end problem solution and discuss one or two particular aspects of the project you found interesting or difficult. You are expected to reflect on the project as a whole to show that you have a firm understanding of the entire process employed in your work. Questions to ask yourself when writing this section:

Have you thoroughly summarized the entire process you used for this project?

Were there any interesting aspects of the project?

Were there any difficult aspects of the project?

Does the final model and solution fit your expectations for the problem, and should it be used in a general setting to solve these types of problems?

6.3 Improvement

In this section, you will need to provide discussion as to how one aspect of the implementation you designed could be improved. As an example, consider ways your implementation can be made more general, and what would need to be modified. You do not need to make this improvement, but the potential solutions resulting from these changes are considered and compared/contrasted to your current solution. Questions to ask yourself when writing this section:

We take a very naive approach to creating the labels for the COCO dataset. Specifically each image has associated with it an 80 element vector of 0's or 1's where the 1's denote that the particular class is found in the image. However the label vector does not take into account multiple instances of the same class, nor does it take into account how much of the actual image the labeled object occupies. It might be interesting to use segmentation maps or object bounding boxes to obtain a percent coverage of the image and use those values instead of a simple binary vector.

Were there algorithms or techniques you researched that you did not know how to implement, but would consider using if you knew how?

It would have been interesting to explore hierarchial classification with using RNN's to learn... zcf whitening with large datasets.

If you used your final solution as the new benchmark, do you think an even better solution exists?

———

Before submitting, ask yourself...

- Does the project report youve written follow a well-organized structure similar to that of the project template?
- Is each section (particularly **Analysis** and **Methodology**) written in a clear, concise and specific fashion? Are there any ambiguous terms or phrases that need clarification?

- Would the intended audience of your project be able to understand your analysis, methods, and results?
- Have you properly proof-read your project report to assure there are minimal grammatical and spelling mistakes?
- Are all the resources used for this project correctly cited and referenced?
- Is the code that implements your solution easily readable and properly commented?
- Does the code execute without error and produce results similar to those reported?

example [17]

MS-COCO [15]

Add back in the negative images and further improve evaluation metrics to handle zero label images.

ACKNOWLEDGMENTS

The authors would like to thank...

REFERENCES

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[2] Krizhevsky, A. and Hinton, G., "Learning Multiple Layers of Features from Tiny Images," *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proc. of the IEEE*, vol. 86, 1998, pp. 2278–2324.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[5] V. Lavrenko, "Evaluation 12: mean average precision," https://www.youtube.com/watch?v=pM6DJ0ZZee0&list=PLBv09BD7ez_6nqE9YU9bQXpjJ5jJ1Kgr9&index=12, 2014.

[6] G. Tsoumakas and I. Katakis, "Multi-Label Classification: An Overview," *International Journal of Data Warehousing and Mining (IJDWM)*, 2007. [Online]. Available: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/jdwm.2007070101

[7] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing and Management*, vol. 45, no. 4, pp. 427 – 437, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306457309000259

[8] F. Herrera, F. Charte, A. J. Rivera, and M. J. del Jesus, *Multilabel Classification*. Springer International Publishing, 2016.

[9] G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. Deroski, "An extensive experimental comparison of methods for multi-label learning," *Pattern Recognition*, vol. 45, no. 9, pp. 3084 – 3104, 2012, best Papers of Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA'2011). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320312001203

[10] X. Wu and Z. Zhou, "A unified view of multi-label performance measures," *CoRR*, vol. abs/1609.00288, 2016. [Online]. Available: http://arxiv.org/abs/1609.00288

[11] O. Koyejo, P. Ravikumar, N. Natarajan, and I. S. Dhillon, "Consistent multilabel classification," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 3321–3329. [Online]. Available: http://dl.acm.org/citation.cfm?id=2969442.2969610

[12] Y. Gong, Y. Jia, T. Leung, A. Toshev, and S. Ioffe, "Deep convolutional ranking for multilabel image annotation," *CoRR*, vol. abs/1312.4894, 2013. [Online]. Available: http://arxiv.org/abs/1312.4894

[13] Y. Li, Y. Song, and J. Luo, "Improving pairwise ranking for multi-label image classification," *CoRR*, vol. abs/1704.03135, 2017. [Online]. Available: http://arxiv.org/abs/1704.03135

[14] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Mach. Learn.*, vol. 85, no. 3, pp. 333–359, 12 2011. [Online]. Available: http://dx.doi.org/10.1007/s10994-011-5256-5

[15] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[16] P.-T. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Annals of Operations Research*, vol. 134, no. 1, pp. 19–67, Feb 2005. [Online]. Available: https://doi.org/10.1007/s10479-005-5724-z

[17] K. Sechidis, G. Tsoumakas, and I. Vlahavas, *On the Stratification of Multi-label Data*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 145–158. [Online]. Available: https://doi.org/10.1007/978-3-642-23808-6_10

PLACE PHOTO HERE

Willie Maddox Biography text here.