Multi-Label Classification

Willie Maddox

November 14, 2017

This document serves as the proposal for the final Capstone project for the Machine Learning Engineer Nanodegree offered through Udacity. For this work we will train a model to recognize multiple items in an image. However, unlike traditional classification problems, we will train our models using image datasets with multiple labels per image. Our goal is to achieve as good as if not better results than popular single-label classification models.

1 Domain Background

In machine learning, image classification is the problem of deciding to which category a particular image belongs. 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 classifier that can be trained using images that have multiple labels per image.

2 Problem Statement

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?

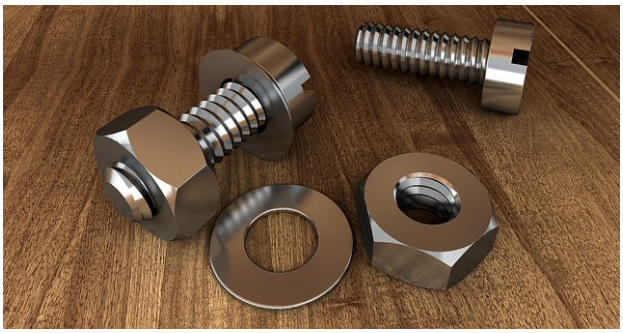

Figure 1: *Nuts n Bolts*

3 Datasets and Inputs

For this study we will use the NUS-WIDE[5] dataset as our primary dataset. We may consider using the MS-COCO[6] dataset as well. These datasets contain images with multiple labels per image making them ideal for this kind of study. They are also large enough that we can use them for training. Both datasets are feely downloadable from the internet. However, we had to explicitly request permission to use and download the image set for the NUS-WIDE dataset. The authors were happy to grant us permission.

The NUS-WIDE dataset contains 269,648 images and 81 labels with 2.4 labels per image on average. The images vary in size (height and width values fall within the range between about 150 pixels to 250 pixels) and shape (portrait and landscape) and all have 3 color channels. We will use the original train/test split of 161,789 and 107,859. The MS-COCO dataset contains a training set of 82,783 images and a test set of 40,504 images. However, the test set does not come with ground truth so we will use a subset of the training set for validation and testing. The MS-COCO dataset has 80 labels with 2.9 labels per image on average. Since we will be fine-tuning a model pretrained on ImageNet, these two datasets are good candidates since all their labels are a subset of the original 1000 ImageNet labels.

4 Solution Statement

The solution is to train a model that is capable of generating (or recommending) a finite set of labels based soley on an input image. For the training we will explore a few algorithms. We will compare between different back propogation optimizers such as rmsprop, Adam and SGD. We will use transfer learning to fine-tune a pre-existing model based on the labels from our datasets. We will measure the performance of our model by comparing the top 3 predicted labels from our model against the known ground truth labels for each of our images[1]. The specific metrics used are discussed in the following sections. We set aside a subset of our training data which will be used exclusively for testing. We will use the Keras python package (with the TensorFlow backend) to create our neural nets and we will train them using a single Nvidia 1080 Ti GPU.

5 Benchmark Model

To benchmark the solution above, we will compare the testing set against the original VGG16 model pre-trained on ImageNet[2] (Just the vanilla VGG16 model with 1000 classes. No fine tuning.) This will be our primary benchmark model. Since we will be using this same base model for transfer learning, we should expect our model to perform better at classifing single-label images from the 81 category NUS-WIDE label set.

For the second model, we will use Clarifai's image recognition API[3]. Clarifai's image recognition systems recognize various categories, objects, and tags in images, as well as find similar images. The company's image recognition systems also allow its users to find similar images in large uncategorized repositories using a combination of semantic and visual similarities.

We will use the evaluation metrics below to quantify how well our model does against state-of-the-art models. These models include KNN[5], WARP[7], CNN-RNN[8], and ResNet-SRN[9].

6 Evaluation 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[10]. Other popular metrics include precision, recall, F_1 score, Jaccard index, 0/1 loss and Hamming loss[11;12;13]. At this point, though, it is unclear which of these metrics will provide the best insight for our results.

[1] since both datasets have ≈ 3 labels per image on average, $k = 3$ seems appropriate

[2] `https://keras.io/applications/`

[3] `https://www.clarifai.com/`

Table 1: *Initial configuration of our proposed top neural network model. The fc's are the fully-connected layers and the ReLU is a rectified linear unit activation layer. In the output shape, "None" is just a placeholder for the batch size. For a model trained on the Pascal VOC data, the final 2 layers would become (None, 80)*

Layer Type	Output Shape
fc	(None, 1024)
ReLU	(None, 1024)
dropout	(None, 1024)
fc	(None, 81)
sigmoid	(None, 81)

7 Project Design

The tenative schedule for solving the given problem will go as follows. First we will read the datasets and split them into training/validation/testing subsets. Each image will need to be resized and pixel values scaled to a range between 0 and 1. We will either create one model out of both datasets or two models, one for each dataset.

Next we will set up our neural network. Since training is expensive we will implement transfer learning to leverage the pretrained weights of a preexisting model. We will use the VGG16 model pretrained on ImageNet. This model is provided to us through the Keras codebase. When we load the model, we will leave off the 3 fully-connected (fc) layers at the top of the network. This will serve as our base model for the entire project. We will run the base model on each input (*i.e.* image) in our dataset and create a set of bottleneck features from the resulting predictions.

We will then use these bottleneck features to train a small fc model which will later be attached to our base model. The number of layers and the number of neurons per layer will be hyperparameters that we will explore, but we will start out with a simple fc network as shown in Table 1. For multi-*class* classification problems it is common to use a softmax layer for the final output layer. This forces the loss function to place bias on a single label during the backpropogation phase. Since our problem is a multi-*label* classification problem, we do not want to give a preference to any particular (ground truth) label since we are treating them equally. For this reason, we will use a standard sigmoid layer is our final output classifier. For the loss function we will use binary cross entropy and for back-propogation we will use rmsprop to update the weights. After training this small fc model, we will attach it to the top of the base model and this will serve as our main model.

Now that we have our main model we can fine-tune the entire network. For this, we will again use binary

cross entropy for the loss function, but since we're fine-tuning, we want to use a slow learning rate so that we do not overfit. For this we will use stochastic gradient descent with a 1e-4 learning rate and 0.9 momentum. We will also freeze most of the bottom layers of the base model so that training can go faster. The number of layers at the end of the base layer that we will leave unfrozen will be a hyperparameter for us to explore.

Training this final model will be significantly slower than training the bottleneck features, especially if we decide to unfreeze some of the top convolutional layers from the base model. This is because file input/output between CPU and GPU is a computational bottleneck. To help speed things up, we will process images from our training set in batches. The size of each batch is also a hyperparameter, but will most likely be 32, 64, 128, or some other power of 2. For each batch, we will augment the images using a number of random transformations (i.e. zoom, rotation, horizontal flip, noise, etc.). This will ensure that our model never sees the same image twice. After each epoch, we will calculate the loss and accuracy of the model using the validation set. This will allow us to detect if our model is overfitting. In addition, we will check if the validation loss is less than the validation loss from the previous epoch and if so, we will create a checkpoint by saving the weights to a file.

Once the model is trained we will run predictions on our holdout test set using the metrics listed in Section 6 and compare them to those currently found in the literature. We will also compare our results to the benchmark models described in Section 5.

References

[1] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):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*, volume 86, pages 2278–2324, 1998.

[4] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL `http://arxiv.org/abs/1409.1556`.

[5] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yan-Tao Zheng. NUS-WIDE: A Real-World Web Image Database from National University of Singapore. In *Proc. of ACM Conf. on Image and Video Retrieval (CIVR'09)*, Santorini, Greece., July 8-10, 2009.

[6] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. *CoRR*, abs/1405.0312, 2014.

[7] Yunchao Gong, Yangqing Jia, Thomas Leung, Alexander Toshev, and Sergey Ioffe. Deep convolutional ranking for multilabel image annotation. *CoRR*, abs/1312.4894, 2013. URL `http://arxiv.org/abs/1312.4894`.

[8] Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. CNN-RNN: A unified framework for multi-label image classification. *CoRR*, abs/1604.04573, 2016. URL `http://arxiv.org/abs/1604.04573`.

[9] Feng Zhu, Hongsheng Li, Wanli Ouyang, Nenghai Yu, and Xiaogang Wang. Learning spatial regularization with image-level supervisions for multi-label image classification. *CoRR*, abs/1702.05891, 2017. URL `http://arxiv.org/abs/1702.05891`.

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

[11] Grigorios Tsoumakas and Ioannis Katakis. Multi-Label Classification: An Overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 2007. URL `http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/jdwm.2007070101`.

[12] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 45(4):427 – 437, 2009. URL `http://www.sciencedirect.com/science/article/pii/S0306457309000259`.

[13] Francisco Herrera, Francisco Charte, Antonio J. Rivera, and María J. del Jesus. *Multilabel Classification*. Springer International Publishing, 2016.