ResNet-50 Architecture for Multi-Label Classification of ISS Satellite Images

TERC 1
Sarah Ferry, Neela Kaushik, Dimitri Makrigiorgos
ferrys@bu.edu, nkaushik@bu.edu, dmak1112@bu.edu

Figure 1. Sample images from the ISS. The present objective is to recognize these images and classify them with one or multiple tags provided by the TERC Windows on Earth team.

1. Project Task

TERC is a non-profit organization whose mission is to improve math and science education across the nation. Their Windows on Earth program is an educational project that provides free public access to thousands of photographs taken by astronauts on the ISS. The objective of this project is to provide the TERC Windows on Earth team with a program that can analyze the images from the ISS and automatically tag them with 12 basic tags with a high level of accuracy. TERC receives thousands of images every day but are only able to manually tag less than 1% of these images. A machine learning, image recognition implementation that achieves high multi-label tagging accuracy would greatly facilitate their efforts to build a searchable database of images for research, education and public outreach.

2. Related Work

This project hinges on achieving an image recognition model that is capable of multi-label classification for learning and tagging an image with multiple labels. Every real-world image can be described using multiple labels, all of which are essential to the rich semantic information and attributes of the object, scene or action depicted. Multi-label image classification has therefore been an important task in machine learning and computer vision for many years, to achieve greater image understanding during the classification process.

Early work in multi-label classification centered around identifying objects in different subsections of an image [1]. More recently, many results have shown that weights from deep neural networks pre-trained on single label classifications can be extended to multi-label classification with good results, and the success of deeply-learned features for single-label image classification have increased the accuracy of multi-label classification [3][5][8].

CNN architectures in particular are growing increasingly popular for multi-label classification tasks. Traditional methods for adapting single-label classifiers to multi-label learn independent classifiers for each label and employ ranking or thresholding on the classification results. Gong et al. evaluated various loss functions to use for multi-label classification tasks using CNNs, and reported strong results for a weighted approximate ranking loss as well as a cross-entropy log loss [2].

Recent work by Zhu et al. and others uses ResNet, a deep CNN architecture, as a structure on which to base a deep neural network for performing multi-label classification [7]. Common architectures include ResNet-50 and Resnet-101, which contain 50 and 101 residual layers, respectively. Results from Krizhevsky, Sutskever & Hinton have demonstrated that CNN models, including those based on ResNet architectures, can be pre-trained on large datasets with an existing wide diversity of data, such as ImageNet, to extract CNN features for other images for which diverse training data might not exist [3]. Pre-training the model can also greatly expedite the training and fine-tuning process.

3. Approach

We build off of these successful methods for multi-label classification and use a ResNet architecture with ImageNet pre-training for our implementation. The neural network we use is based on an out-of-the-box ResNet-50 implementation pre-trained on ImageNet [6]. The architecture is implemented in Python using the Keras library built on top of TensorFlow. We freeze the standard top layers for ResNet-50 and replace the output layer with a new layer of density twelve to account for the twelve categories we are interested in classifying (Figure 2). The classification model we employ uses the sigmoid activation function and a binary cross-entropy loss defined as:

$$L(p) = -\frac{1}{N}\sum_{k=1}^{N}\left(\frac{1}{n}\sum_{i=1}^{n} y_{ki}\log(p_{ki}) + (1-y_{ki})\log(1-p_{ki})\right)$$

Where N is the number of tags, n is the number of samples, p_{ki} is the predicted probability that tag k for sample i is 1 (i.e. exists) and y_{ki} is the true value of tag k for sample i (i.e. exists or does not exist).

Given an image, the model predicts the probability of the image having each of 12 tags. For each tag, if the predicted probability is higher than 50% then the model assigns the tag to the image; otherwise, it predicts the tag is not associated with the image.

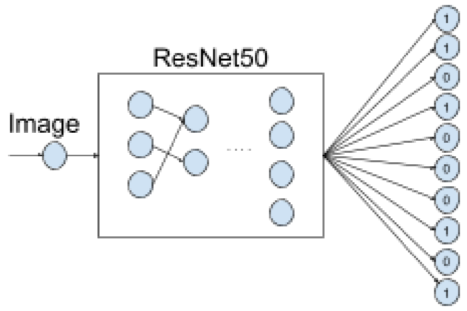

Figure 2. A high-level architecture of our approach using a fine-tuned version of the ResNet-50 CNN.

4. Dataset and Metric

The full dataset consists of 13,026 tagged images provided by TERC. Images are shuffled and randomly assigned to training, validation and testing sets at a 60/20/20 split, resulting in 7,816 training examples, 2,605 validation examples and 2,605 test examples. Each image is resized to 224 x 224, the standard input size for ResNet-50. Image labels are obtained from the EXIF metadata and stored with a categorical encoding scheme (1=tag exists, 0=tag does not exist) for each of 12 tags of interest. Images and their labels are vectorized and passed into the neural network for processing.

Category	Number of Occurrences			
	Training	Validation	Testing	Total
Volcano	38	10	6	54
Sunrise/Sunset	166	64	85	315
ISS Structure	277	70	79	376
Stars	29	14	10	53
Night	295	89	96	480
Aurora	110	36	46	192
Movie	417	151	135	703
Day	3340	1108	1062	5510
Moon	393	138	135	666
Inside ISS	228	76	78	382
Dock/Undock	148	51	47	246
Cupola	144	44	46	234

Table 1. Total number of occurrences for each tag among our training, validation and testing data.

Our metrics for model evaluation include computing binary accuracy and F1 scores for the model's predicted tags on the validation and testing data against the corresponding, real tags. Binary accuracy is a metric predefined in the Keras package that reports the mean accuracy over all labels:

$$Accuracy = \frac{1}{n}\sum_{i=1}^{n}\frac{TP_i + TN_i}{P_i + N_i}$$

Where n is the number of tags, TP_i , TN_i are the number of true positive predictions and true negative predictions across all images for a given tag i and P_i and N_i are the total number of positive and negative occurrences for that tag across all images.

The F1 score is a popular metric used for evaluating classification models that takes into account the model's precision and recall. Precision is the fraction of relevant instances among all retrieved instances. Recall is the fraction of relevant instances retrieved among the total number of relevant instances. The F1 score is the harmonic mean of precision and recall. We implement the Keras precision, recall and F1 metrics, which are computed as the batch-wise average over each set of images in n training batches:

$$Precision = \frac{1}{n}\sum_{i=1}^{n}\frac{TP_i}{TP_i + FP_i} \qquad Recall = \frac{1}{n}\sum_{i=1}^{n}\frac{TP_i}{TP_i + FN_i}$$

$$F1 = \frac{1}{n}\sum_{i=1}^{n} 2 * \frac{Precision_i * Recall_i}{Precision_i + Recall_i}$$

TERC provided us with a baseline goal of achieving 80% classification accuracy per tag. As it turns out, the distribution of tags among the provided training is highly skewed in a manner such that this benchmark is easy to achieve even with a naive classifier (see Table 1 for tag counts in the dataset). One baseline method we consider operates by identifying the tags that occur in a majority of the images, and assigning only those tags to all images. Because most tags do not occur in the majority of images, with the tag "day" being the only notable exception, this naive baseline would correctly document true negatives and actually achieve very high accuracies of >93% for most of the tags. The tag "volcano," for instance, only occurs in 6 of the testing images and would have a 99.65% accuracy using the naive baseline and not assigning any image the volcano tag (see Table 2 for per-tag accuracies using this baseline method).

Therefore, due to the nature of the dataset and the relative ease with which we are able to achieve TERC's proposed performance objectives for the model, we instead set our goal to outperform the accuracy of the naive baseline on all tag categories.

5. Evaluation

We performed cross-validation to determine the best hyper-parameters for fine-tuning the model. We trained the model for 10 epochs on the training data, using a batch size of 16 and the learning rates α=0.1, α=0.01, α=0.001 and α=0.0001. We then tested each model on the validation dataset and evaluated its accuracy and loss to determine the optimal learning rate. Accuracy and loss curves for each learning rate are graphed in Figure 3.

Based on these experimental results, the best learning rate for our model and data was determined to be α=0.01 as the model trained with this α yielded the the highest F1 score and lowest log loss when tested on the validation data at the end of training. We then retrained the model for 100 epochs using these hyper-parameters to provide TERC with a highly trained model. After the model was trained, we ran it on the test data to evaluate our performance.

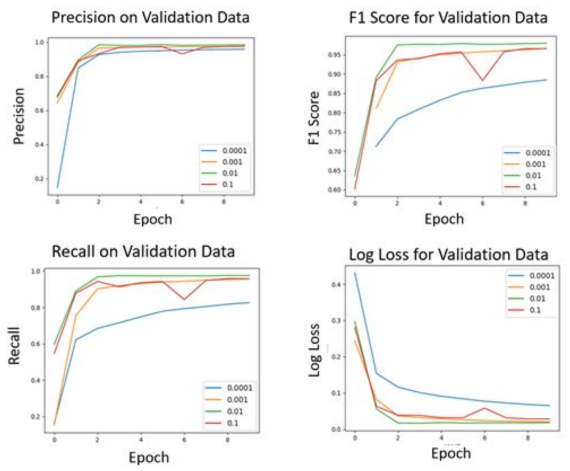

Figure 3. Experimental results of varying learning rate α on trained model's precision, recall, F1 score and log loss when evaluated on validation data.

Category	Accuracy			
	Baseline	$\alpha = 0.001$	$\alpha = 0.01$	Test
Volcano	0.9965	0.9977	0.9985	0.9992
Sunrise/Sunset	0.9555	0.9965	0.9977	0.9992
ISS Structure	0.8879	0.9916	0.9950	0.9962
Stars	0.9954	0.9977	0.9988	0.9992
Night	0.9501	0.9981	0.9942	0.9965
Aurora	0.9800	0.9992	0.9992	0.9992
Movie	0.9466	0.9934	0.9896	0.9965
Day	0.6190	0.9873	0.9919	0.9961
Moon	0.9389	0.9988	0.9985	0.9988
Inside ISS	0.9558	0.9977	0.9988	0.9985
Dock/Undock	0.9585	0.9962	0.9977	0.9985
Cupola	0.9697	0.9988	0.9977	0.9981
Overall	N/A	0.9647	0.9658	0.9827

Table 2. Performance of naive baseline method, cross-validation experiments and final trained model on test data. The trained model achieves strong results of >0.99% accuracy per tag.

Table 2 displays our trained model's per-category performance on the test data. We define overall accuracy to be the proportion of images for which the model classifies every tag correctly, i.e. with no false positive or negative labels. The trained model yields a very high overall accuracy of 98.27% on the test data. Additionally, the model outperforms the naive baseline for every category, fulfilling both TERC's and our desired objectives (Figure 4).

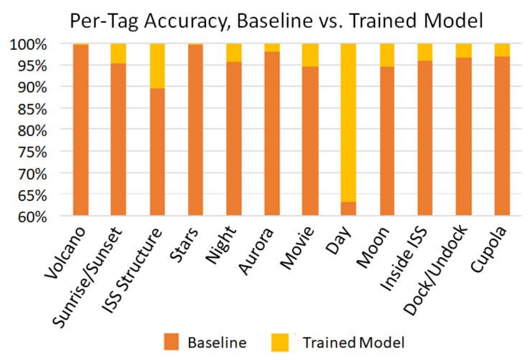

Figure 4. Accuracy of trained model versus naive baseline for each tag. The trained model outperforms the baseline in every instance.

6. Conclusion

Our task was to provide the TERC Windows on Earth team with an accurate, easy-to-use trained CNN model to perform multi-label classification and tag images from the ISS with high levels of accuracy. To achieve this objective, we implement a pretrained ResNet-50 architecture with an output layer of density twelve to provide a classification for each of the twelve tags.

Upon fine-tuning and training this model, we obtain very high per-tag and overall image tagging accuracies of >98%. At present, the model is trained and tested on a relatively sparse and unbalanced data set. However, we believe it is robust enough to hold up to the tasks TERC requires of it at present, and the model can always be retrained on a more complete dataset in the future to yield even stronger results.

As our final deliverable, we provide TERC with a Python script that takes in as input a folder of untagged images, loads the trained model and runs the images through the neural network, inserts the predicted tags into the EXIF metadata, and returns a folder of tagged images. Overall, we feel that this deliverable is very successful and can be of much value to the TERC Windows on Earth team.

7. Roles

Task (lines of code)	**Lead**
Extract image metadata (40 lines)	Sarah
Preprocessed images (150 lines)	Neela
Create and test model (420 lines)	All
Re-associated image metadata (20 lines)	Neela
Finalize model for TERC (70 lines)	Sarah
Prepare report and presentation	All

8. Link to GitHub repo

https://github.com/ferrys/terc

References

[1] Duygulu, P., Barnard, K., de Freitas, J. F., & Forsyth, D. A. (2002, May). Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *European conference on computer vision* (pp. 97-112). Springer, Berlin, Heidelberg.

[2] Gong, Y., Jia, Y., Leung, T., Toshev, A., & Ioffe, S. (2013). Deep convolutional ranking for multilabel image annotation. *arXiv preprint arXiv:1312.4894*.

[3] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

[4] Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C., & Xu, W. (2016). Cnn-rnn: A unified framework for multi-label image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2285-2294).

[5] Y. Wei, W. Xia, J. Huang, B. Ni, J. Dong, Y. Zhao, and S. Yan, "Cnn: Single-label to multi-label," arXiv preprint arXiv:1406.5726, 2014.

[6] Yu, F. (2016, October 03). A Comprehensive guide to Fine-tuning Deep Learning Models in Keras (Part I). From https://flyyufelix.github.io/2016/10/03/fine-tuning-in-keras-part1.html.

[7] Zhu, F., Li, H., Ouyang, W., Yu, N., & Wang, X. (2017). Learning Spatial Regularization with Image-level Supervisions for Multi-label Image Classification. *arXiv preprint arXiv:1702.05891*.