IDENTIFYING DEFORESTATION IN THE AMAZON USING DEEP CONVOLUTIONAL NEURAL NETWORKS

December 11, 2020

ABSTRACT

This paper investigates the use of transfer learning on a dataset of multi-labeled satellite images from the Amazon rainforest using the deep convolutional neural network (CNN) MobileNetV2 pre-trained on ImageNet. A typical problem in multi-label classification (MLC) that we also encountered is the imbalance of positive and negative labels. To handle this we used a recently developed asymmetric loss (ASL) designed specifically for MLC and confirmed that it also works well for our dataset. As usual in transfer learning with large datasets we fine-tuned the convolutional base and in combination with a careful optimization process using learning rate cycling we achieved an F_2 score of 90.5%

***K*eywords** Deep Learning · Remote Sensing · Multi-Label Classification

1 Introduction

In recent years satellite images have become increasingly accessible. Furthermore, improvements to sub-meter resolution and accuracy means that techniques from the field of computer vision such as CNNs for object detection can be used which was not possible earlier. In this paper we explore the specific problem of land-cover and land-use classification of satellite images over the Amazon basin. This is a problem that has been solved well using large neural networks [1] and as such this paper investigates if similar results can be achieved with a much smaller network such as the MobileNetV2 network pre-trained on ImageNet. Our findings show that it is indeed possible through simple techniques to achieve an F_2 score that surpasses 90% which is comparable to the 93,3% score of the winner of the Kaggle competition that provided the dataset [2]. We achieved this without significant tuning of hyperparameters of e.g. the loss function, which could in principle have brought it closer to the scoring function. Additionally, our results were achieved in a small number of epochs, which indicates that the feature detectors of the pre-trained MobileNetV2 transfer well to satellite data as well, even switching from a one-hot classifier to predicting multiple labels at once. Our final model takes as input a satellite image and outputs a probability indicating the presence of each of 17 labels with a precision of 78.1% and 94.2% recall.

2 Related Work

2.1 Remote Sensing

The field of remote sensing revolves around the enormous amounts of data that is generated by satellites and other remote sources. It is then analysed for a wide range of applications from monitoring the effects of climate change on the ice sheet [3] to crop yield forecasting which enable earlier responses to famines [4].

The first satellite images of the earth were taken in 1959 and in the beginning the low resolution of the images often led to the objects of interest being smaller than a single pixels. As such, per-pixels based classification or labelling was common but as the resolution increased the field moved to object based (multi-pixel) methods [5]. A standard strategy for classification is to work in the feature space instead of the full pixel space of the images. At first these features were hand-crafted by domain experts and they include global features from color histograms [6] and local features from the SIFT descriptor [7]. A major disadvantage of the hand-crafted features however is that it remains non-trivial to design the individual feature extractors as well as choosing how to combine them. To remedy these concerns another approach is to use unsupervised feature learning by e.g. PCA or Autoencoders which both have been applied in remote sensing [8]. Finally, deep learning based approaches using CNNs have been rising in popularity in recent years. Concretely, in 2015 for the UC Merced land-use dataset [9] with 2100 images and 21 classes Castelluccio et al. were able to achieve a new state of the art beating all previous more traditional methods by a margin of 3% [10]. They did this by fine-tuning a GoogLeNet pre-trained on ImageNet [11] and overall there is a lot interest in using transfer learning for remote sensing, since it is expensive to obtain a large multi-labelled dataset which is needed when training a large neural network. Our work now extends this exploration by combining deep learning and remote sensing and applying it to the specific domain of land-use and land-cover classification of the Amazon.

2.2 Multi-Label Classification

Classification normally means determining which class an instance belongs to out of a number of classes. Multi-label classification is a little different, as each instance can now have several different labels at the same time. This type of problem can be solved using both deep learning techniques and as well as more traditional machine learning methods. One of the early approaches to multi-label classification was done by McCallum, A. K., who looked at the problem of solving multi-label document classification using a Bayesian Classifier [12].

When solving multi-label classification the model usually tries to determine the existence of each class based on shared features. While this have given a lot of good results, Zhang et al. argues that this might be suboptimal, and suggests a new strategy that makes use of label-specific features [13]. The paper shows through comparative experiments on different datasets, that this kind of label-wise feature extraction enhances the performance of the learner compared to other multi-label classification approaches.

For classification of multi-label images, a lot of different approaches have been used. One of these is deep convolutional neural-networks, which have made a lot of progress in the area. A framework that combines recurrent neural networks (RNNs) and convolutional neural networks was proposed by Wang et al. [14]. This framework utilizes RNNs with long-short term memory to investigate label-cooccurrences to model the interdependencies of the labels. This achieves a much better accuracy and beats earlier state-of-the-art multi-label classification models.

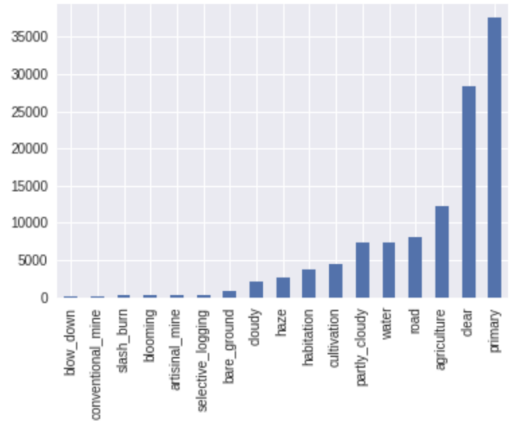

Figure 1: Number of occurrences of each label in the dataset.

3 Methods

3.1 Data

The Planet[15] dataset consists of 40,479 chips (images) extracted from larger scenes shot by satellites over the Amazon basin in 2016-2017. Each chip has a ground-sample distance of 3.7m, which is reduced to 3m after orthorectification (technique for image pre-processing [16]). Images are 256x256 pixels, and we used the .jpg versions which we converted from the CMYK format to RGB. We decided to split the images into 32,000 training images, 4,000 validation images and 4,479 test images. Images are labeled in three categories (atmospheric conditions, common land cover/land use phenomena,

Figure 2: Four example images from the dataset and the assigned labels

and rare land cover/land use phenomena). There are two important labelling constraints which are that every image has exactly one atmospheric label and the *cloudy* label always appears alone indicating that the image was too cloudy to interpret meaningfully. Furthermore, the label distribution is significantly imbalanced as seen in Figure 1 with *primary* being the most common label appearing more than 37513 times and *blowdown* being the least common label only appearing only 98 times. Figure 2 shows a few example images from the dataset along with the assigned labels.

3.1.1 Data Preprocessing

Since we are using a pre-trained version of MobileNetV2, it is important to make sure the exact dimensions of the inputs matches the dimensions expected by the network. As such, we reshaped the images to 224x224 pixels by performing a center crop (small chance of losing features in the images). Furthermore, we normalized our images using the mean and standard deviation from ImageNet which is the same normalization used to pre-train MobileNetV2.

3.1.2 Data Augmentation

To prevent overfitting we used real-time data augmented by randomly flipping, rotating, and altering the brightness of the image. This ensures that the model only sees the same image once which can greatly improve the accuracy and helps regularize the model [17]. We chose some very simple augmentation (rather than something like proposed in aforementioned paper) as our number of epochs made our time better spent elsewhere. Also, it is important to mention that for satellite images of a forest it is reasonable to assume that flipping, rotating, and cropping an image does not make the image fall out of the original data distribution. This is unlike for example written text or numbers where a flipped image is not present in the original data distribution.

3.2 Model Evaluation

When evaluating our model we scored it based on the F_2 score which is a combination of precision and recall. Precision is calculated as the ratio of true positives (tp) to all predicted positives ($pp = tp + fp$ where fp denotes false positives) and recall is the ratio of true positives to all actual positives ($ap = tp + fn$ where fn denotes false negatives). Many traditional loss functions consider them equally important, but as mentioned we used the F_2 score since it was the one chosen by the organisers of the Kaggle competition. The F_2 score essentially rewards models with high recall rather than ones with high precision and is given by the formula:

$$F_2 = \frac{5 \cdot p \cdot r}{4 \cdot p + r} \quad \text{where} \quad p = \frac{tp}{pp} \ , \ r = \frac{tp}{ap}$$

3.3 Sampling with Weights

Rare labels tend to be neglected when training the model as their total contribution to the loss is unimportant compared to the more common labels. One way to get around this problem is through oversampling. This effectively means

that their rarity is decreased by some factor depending on the weights used when sampling. It can however prove very cumbersome to develop a satisfying weighting strategy in a multi-label setting, where correlations between labels and differences in how many other labels each label typically occurs alongside makes it hard to even out all label occurrences in a smart way. Specifically, for our dataset the *primary* label occurs on almost all images making it non-trivial to design an oversampling strategy that balances the *primary* label with the others.

3.4 Model Architecture and Loss Function

Since our data consists of images we choose a CNN as our machine learning model. Specifically, we used the MobileNetV2 architecture introduced in [18].

We only had limited access to proper hardware and it was thus important for us to chose an architecture that would be feasible to train and run on our dataset in a reasonable amount of time. Here MobileNetV2 was a good choice given its ability to combine good accuracy with few parameters and efficient computation of forward passes as described in [19, 20]. In figure 3 we show that MobileNetV2 is is mainly composed of *bottleneck* layers (see figure 4) which consists of depth-wise separable convolutions, inverted residual connections between low-dimensional feature maps, and batch normalization layers. We used the PyTorch implementation and pre-trained weights from and replaced the final layer with a single fully-connected layer with 17 outputs corresponding to each label. When predicting we used a sigmoid layer to convert the values into probabilities for each label. To train the model we used the Asymmetric Loss (ASL) described in [21]. The loss is specified using hyperparameters m,γ_-, and γ_+ and we define $p_m(x) \coloneqq \max(p(x) - m, 0)$. Now letting l be the number of labels in the dataset the loss for a single input x is given by:

$$ASL = \sum_{i=1}^{l} \begin{cases} (1 - p(x))^{\gamma_+} \log(p_m(x)), & \text{if } y_i = 1 \\ p_m(x)^{\gamma_-} \log(1 - p(x)), & \text{if } y_i = 0 \end{cases}$$

In our setup we used the default values for the hyperparameters setting $m = 0.05$,$\gamma_- = 4$, and $\gamma_+ = 1$. The idea behind the loss is that setting $\gamma_- > \gamma_+$ emphasizes the positive examples which allows the network to learn even though there are always much fewer positive than negative labels. The parameter m defines a hard threshold which discards easy negative examples allowing the network to focus the harder and positive examples. Finally, datasets for MLC are often mislabelled [22] and ASL mitigates this by greatly reducing the error when the model has a strong prediction opposite of the label which could indicate that the label is incorrect. This is also shown in figure (5).

3.5 Transfer Learning

Transfer learning allows us to benefit from previous training already performed on huge amounts of other data. For this project we opted for a MobileNetV2 model already trained on ImageNet, partly to investigate if we could achieve similar results to much larger networks that competed in the competition hosted by Kaggle. We started out by training a new classifier on top of the network, but due to the somewhat different nature of our images, we also decided to unfreeze the rest of the layers in order to fine-tune the network further. Although 40.000 images is not a huge amount of data,

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Figure 3: MobileNetV2 architecture

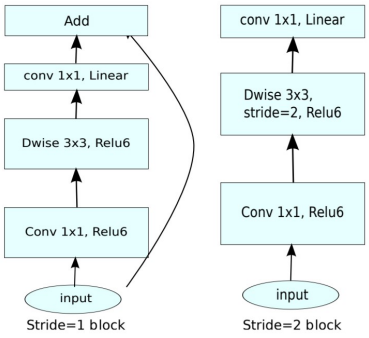

Figure 4: Residual bottleneck layer (left) and bottleneck layer (right) [18]

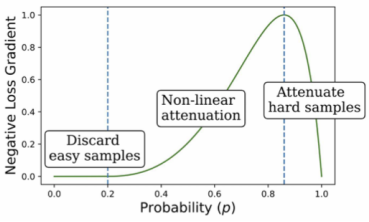

Figure 5: Loss of ALS with respect to model confidence of a label.

excluding a few rare labels it should be enough to prevent overfitting and actually help improve the feature detectors of the network. This is also what we observed during training as seen in figure 7.

3.6 Optimization

3.6.1 Learning Rate Cycling (LRC)

Finding an optimal learning rate can be a time consuming task, but implementing learning rate cycling has proven able to perform at least as good as a fixed optimal learning rate [23]. We determined the learning rate boundaries to be $\left[1 \times 10^{-4}, 1 \times 10^{-1}\right]$ based on the loss at different learning rates shown in Figure 6. However, as we will explain further in section 5.4 in practice we found the learning rate boundary $\left[1 \times 10^{-4}, 1 \times 10^{-3}\right]$ to work better. Furthermore, as recommended, the cycle length should be at least between 4 times the number of batches per epoch and and we chose cycle length of 2250 batches and the mode *triangular2* which halves the interval of the learning rate boundary after each cycle. It turns out that LRC performs well in combination with an adaptive learning rate optimizer like Adam at almost no overhead in terms of training. Additionally, the low initial learning rates work as a good *warmup*, which helps the model to converge faster [24].

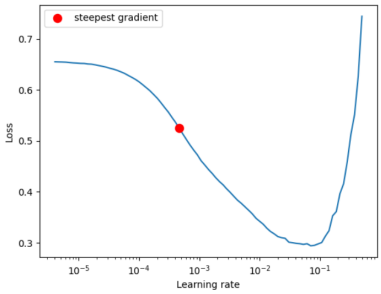

Figure 6: Loss at different learning rates used to decide the boundaries for LRC.

3.6.2 Adam

For our optimization algorithm we used Adaptive Momentum Estimation (Adam). Unlike stochastic gradient descent (SGD) Adam computes an adaptive learning rate for each parameter, whereas SGD only maintains a single learning rate, which does not change during the training. Similar to momentum, the adaptation of the learning rates for each parameter is done using the average of the past squared gradients as in RMSProp, but also the average of past gradients, both of these are computed using exponential decay rates.

4 Results

Figure 7 shows how the training of our final model progressed through 16 epochs. As mentioned in 3.6.1 we perform learning rate cycling with a cycle length of 2250 batches. As expected we observe the loss decreasing between iterations around 1000 to 2000 when the learning rate starts decreasing from its maximum value in the cycle. However, from the other maximum values in the cycle (red lines) the learning rate cycling is not too evident which suggest that we could have used larger interval of learning rate boundaries. Additionally, from iteration around 4000 to 6000 the loss is almost constant indicating that the decoder has converged. At this point we unfroze the convolutional base increasing the capacity of the model and the loss decreased further. From the plot we can also observe that a decrease in the loss correctly translates to a higher F_2 score. Finally, from the plot we see that there no gap between the training and validation loss which shows that the model is not overfitting and that the regularization from batch normalization and data augmentation is sufficient to prevent this issue.

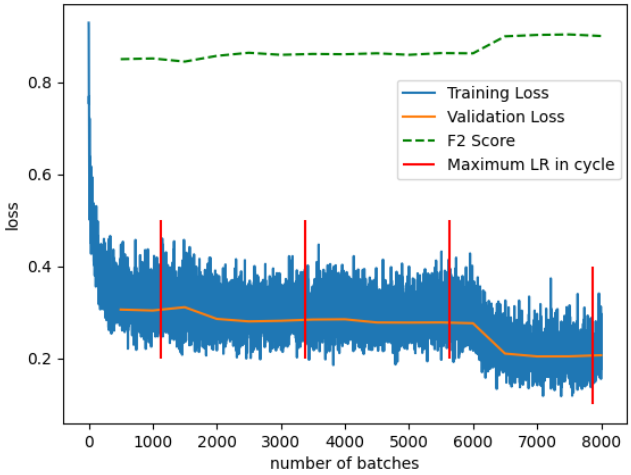

Figure 7: Model scores after 16 epochs.

Table 1: Precision, recall and F_2 score of each label in the final model.

Label	Precision %	Recall %	F2 %
Agriculture	71.9	95.2	89.4
Artisinal Mine	89.3	59.5	63.8
Bare Ground	33.3	42.5	40.3
Blooming	27.3	28.1	28.0
Blow Down	35.7	38.5	37.9
Clear	90.2	99.3	97.3
Cloudy	68.7	94.5	87.8
Conventional Mine	26.7	57.1	46.5
Cultivation	46.2	70.6	63.8
Habitation	54.3	79.4	72.7
Haze	50.2	87.0	75.9
Partly Cloudy	85.6	94.5	92.6
Primary	94.4	99.9	98.8
Road	64.5	91.4	84.4
Selective Logging	20.8	30.3	27.8
Slash Burn	8.33	13.3	11.9
Water	58.0	88.0	79.7
Total	**78.1**	**94.2**	**90.5**

In table 1 we have calculated the precision, recall, and F_2 score of the individual labels on the test set as well as the total across all labels. Here we reach an overall F_2 score of 90.5%. Figure 8 shows an overview of the positives of the final model, and figure 9 shows a few concrete images and the corresponding predictions of our model.

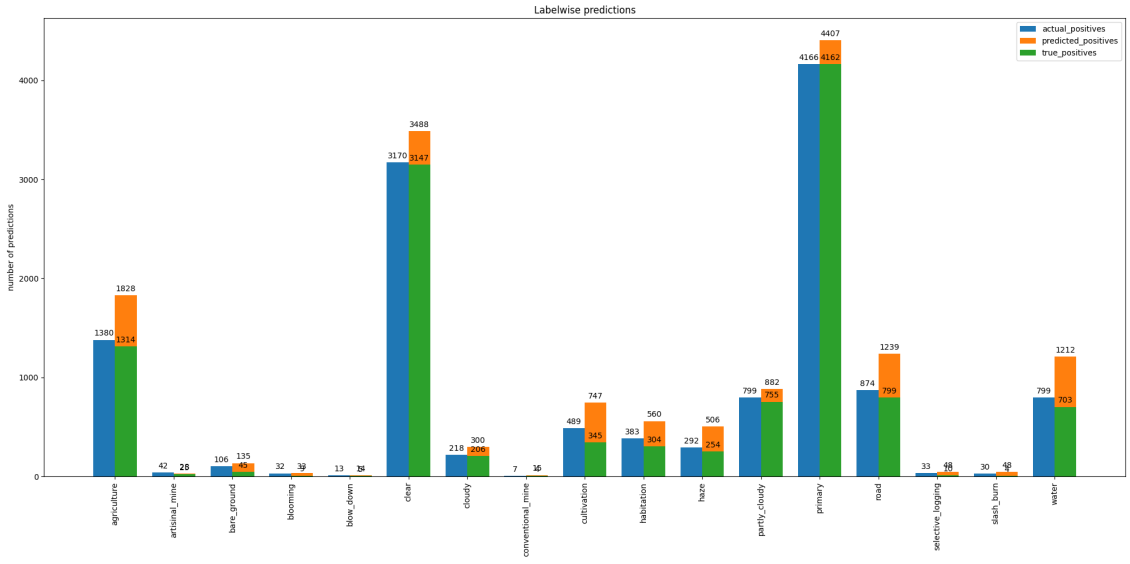

Figure 8: Actual (blue), predicted (orange) and true (green) positives of the final model across all labels.

Figure 9: Examples of label predictions compared to actual labels.

5 Discussion

In this section we will show how we ended up at our final model through an iterative approach, and present intermediate results.

5.1 Binary Cross Entropy with Logits (BCE)

Initially, we used BCE and the Adam optimizer to train a new classifier on top of the pre-trained MobileNetV2 model with the remainder of the model frozen. This produced some very disappointing results, especially on the F_2 score (77%). One problem became very evident as we studied the numbers for pp, ap, and tp on the evaluation of the model - as seen in figure 10 the model basically refuses to predict any label that does not have a massive presence in the data.

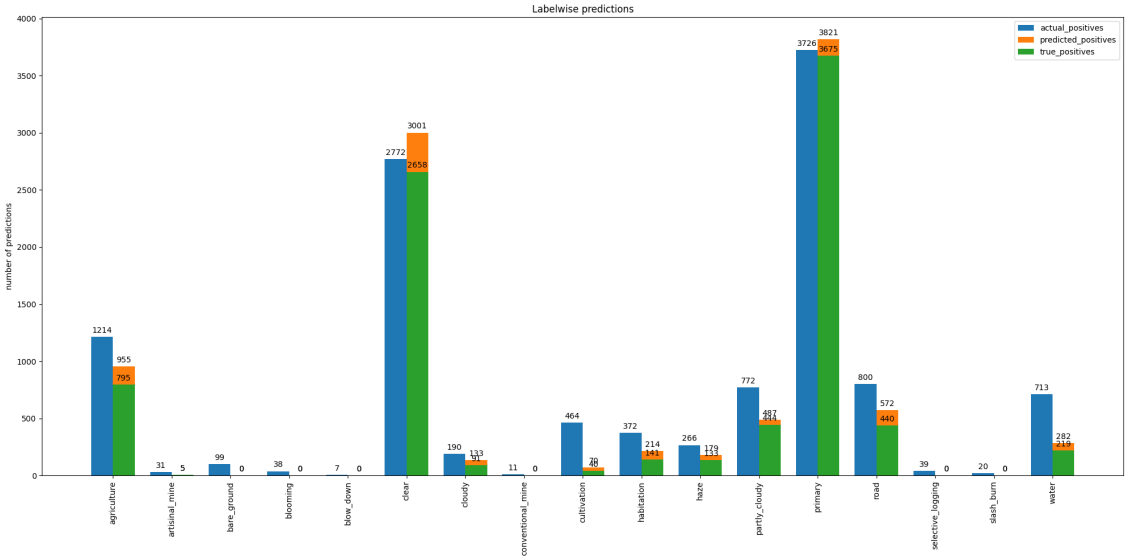

Figure 10: Number of predicted, actual and true positives on validation data using BCE.

5.2 Weighted Sampling and Data Augmentation

Our first attempt in pushing the model to actually guess for the more uncommon labels was to introduce weighted sampling from the training data, i.e. sampling the images with rare labels more often. In order to determine the weight of each image (the probability of choosing it), we calculated for each label the number of occurrences in the data compared to the total number of occurrences of all labels. Then each image was assigned a weight corresponding to the sum of the inverse of all these label-ratios which were present in the image. Or more intuitively - rare labels contributed more weight to the image. This strategy turned out to produce slightly worse results (F_2 score of 69%), as seen in figure 11.

One key observation is that label cooccurrences prevent this strategy from working, as there are big differences in the average number of other labels that each label occur with. For instance the *cloudy* label never occurs alongside any other label, which means that it is the only source of weight for the pictures in which it occurs. On the other hand, a label like *conventional mine* often occurs alongside other labels like *clear* and *primary*, which helps boost its weight even further.

Also, as we introduced weighted sampling, we realized that this might significantly increase the number of times that the model trained on certain images. For this reason we introduced simple data augmentation to make it unlikely for the model to see the exact same image more than once during each epoch.

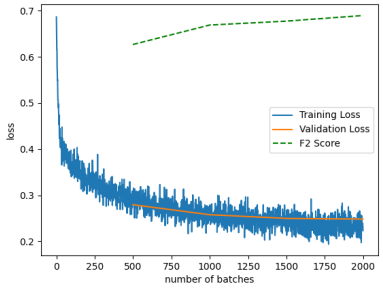

Figure 11: Training loss for BCE with weighted sampling.

5.3 Asymmetric Loss

Although the overall performance is generally more important than predicting well across all labels (this is easily seen in the fact that the scoring function does not keep track of which label a positive comes from), we felt that it was important to improve the models performance both generally and especially on the uncommon labels. This is when we tried to adopt the ASL loss function, which immediately produced some much better results ($F2$ score starting at 83.8% after 1 epoch), as seen in figure 12. As previously described, this means increased focus on positive labels, which also means that the model prioritizes uncommon labels' positives rather than the common labels' negatives. Aside from general improvement, it clearly makes the model guess for all labels in a more accurate way as seen in figure 13. Using ASL instead of BCE yielded an F_2 score of 86.4% after 5 epochs where the improvements stagnated.

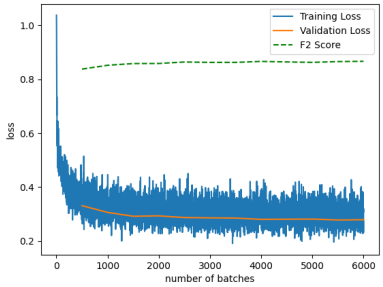

Figure 12: Training loss for ASL

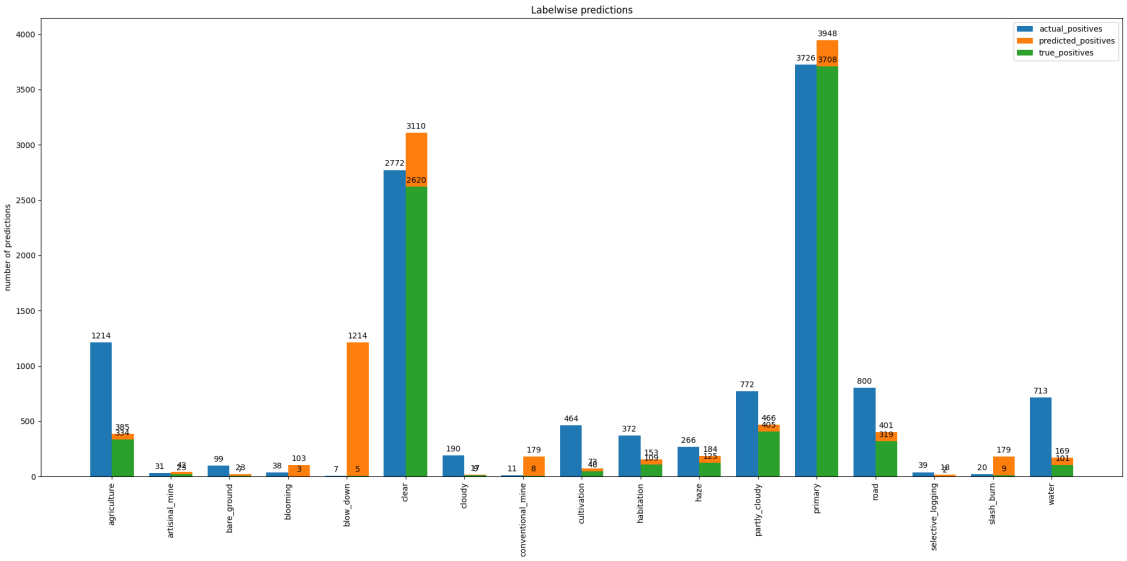

Figure 13: Number of predicted, actual and true positives on validation data using ASL.

5.4 Learning Rate Cycling

Before changing the loss function from ASL to BCE we tested the impact of enabling LRC. As described in section 3.6.1 we first tested the recommended learning rate boundaries of $\left[1 \times 10^{-4}, 1 \times 10^{-1}\right]$. However, as seen in figure 14 this caused our loss to diverge indicating that the learning rate is too high. We then switched to an interval of $\left[1 \times 10^{-4}, 1 \times 10^{-3}\right]$ which similarly to ASL resulted in a significant improvement with an F_2 score of 81.1% as seen in figure 15. In our final model combined ASL and LRC which gave further improvements.

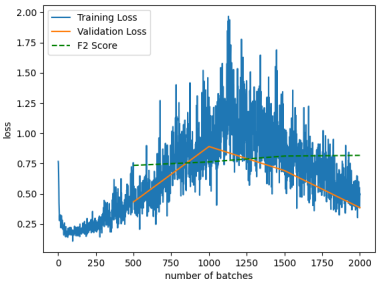

Figure 14: Training loss using LRC with large gap in the learning rate boundaries of $\left[1 \times 10^{-4}, 1 \times 10^{-1}\right]$.

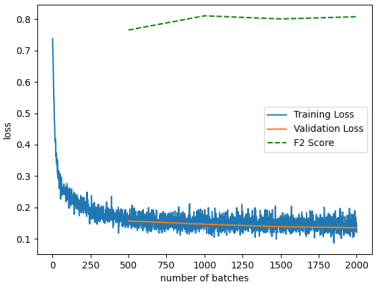

Figure 15: Training loss using LRC with smaller gap in the learning rate boundaries of $\left[1 \times 10^{-4}, 1 \times 10^{-3}\right]$.

5.5 Optimal Thresholds

After the model finished training we were curious if it was possible to push the F_2 additionally by manually optimizing the thresholds of the classifier, such that each label has a unique threshold rather than all of them sharing the same one of 0.5. Intuitively, it seemed unlikely as the classifier receives input from a bias that should be able to shift the results in an optimal way. But the fact that our loss function and our final scoring function are different, we decided to put the idea to the test.

We first calculated the individual F_2 score of each label across every possible threshold with three decimals precision on the training data. An example of the F_2 scores of these thresholds is shown in figure 16.

These new thresholds spanned from 0.380 to 0.546, but as expected the threshold of the common *cloudy* label remained much closer to 0.5 than the rarer *cultivation* and *artisanal mine* labels. Applying the new thresholds to the training data yielded an improvement of 0.089 on the F_2 score, but testing it on the validation data it turned out to reduce the F_2 by 0.099. These results left us very confident that it was not possible to optimize these thresholds on a trained model to achieve a better F_2 score, and that the deviations from 0.5 were simply due to minor differences in the distributions of the training, validation, and test sets.

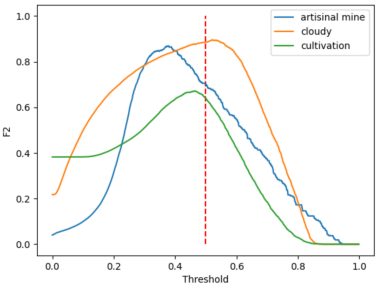

Figure 16: The F_2 score of our final model on the validation set as a function of the threshold value for three different labels.

5.6 Comparison to other Results

As mentioned previously, the top performers on this dataset obtain better results than our model. This is expected however, simply due to the fact that our model is significantly smaller than these state-of-the-art ones. Additionally, some of the top performing teams used ideas such as modelling the interdependencies of the labels that we also left for future work (Section 6).

Comparing to other projects using smaller and more simple networks (a handful of students at Stanford studied the same problem in 2017[25]), which seems like a better benchmark for our model, we find that crossing the 0.9 threshold for F_2 confirms that our model performs quite well, as most of these other results do not quite manage to cross that. Additionally, MobileNetV2 is a small network compared to the networks used by the Stanford students.

5.7 Managing Labelling Constraints

In section 3.1 we described the labelling constraints of the data and when testing our final model we found that these constraints were often violated. To fix these violations we tried the naive approach of adding simple rules to the output of the model such that it satisfies the labelling constraints. Specifically, we tested the following rules:

Rule 0: No changes to the predictions allowing the model to violate the labelling constraints.

Rule 1: Enforce the prediction of exactly one weather label. In our final model this constraint is violated in 15.9% of images.

Rule 2: Enforce that when *cloudy* is the weather label with largest confidence all other labels are removed. Our final model violates this constraint in 7.7% of all images and 82% of images where the *cloudy* label is present.

Table 2 shows how these rules change the precision, recall, and F_2 of the model.

Label	Precision %			Recall %			F2 %		
Clear	90.2	94.3	94.2	99.3	97.9	98.1	97.3	97.1	97.3
Partly Cloudy	85.6	93.0	93.0	94.5	86.1	87.1	92.6	87.4	88.2
Cloudy	68.7	86.0	89.7	94.5	74.2	72.0	87.8	76.3	75.0
Haze	50.2	73.5	77.0	87.0	67.7	67.8	75.9	68.8	69.5
Total	**78.6**	**80.9**	**80.4**	**92.6**	**92.6**	**92.6**	**90.7**	**90.1**	**89.9**

Table 2: Label-wise precision, recall, and F_2 on the validation set for the atmospheric labels when using different rules forcing the predictions to satisfy the labelling constraints. Black, blue, and orange correspond to rules 0, 1, and 2 respectively.

Our first observation is that rules 1 and 2 have a similar effect on the scores where in all cases adding them increases the precision of the model at the cost of lowering the recall significantly and the F_2 score slightly. A difference is that rule 2 lowers the F_2 score more than rule 1 which makes sense since mistakenly predicting *cloudy* now results in a lot of mistakes when all other labels are removed. Based on these observations we can conclude that it is likely possible to further improve the model by making it use the relationships between the labels. Additionally, forcing the predictions to satisfy the labelling constraints does not trivially improve the results because it can give a better F_2 score to predict multiple labels rather than missing one when prioritizing recall over precision.

6 Conclusion and Future Work

In this project we have worked with the important real-world problem of identifying deforestation in the Amazon. Using a deep convolutional network trained with transfer learning we achieved an F_2 score of 90.7% when predicting the land-use and land-cover labels of the satellite images. A specific challenge was that the labels in the dataset are very unbalanced and we explored how different loss functions as well as weighted sampling can help mitigate this issue. For weighted random sampling we found that showing the model more examples of the underrepresented labels improved recall during training but greatly reduced precision during validation. This showed the difficulty of training a model with a different data distribution between training and validation/testing. Given more time we would like to explore different model architectures that more directly exploit the interdependence of labels which is very important in multi-label classification and likewise how to incentives the model to learn the labelling constraints in the dataset.

References

[1] Planet: Understanding the amazon from space, leaderboard.
`https://www.kaggle.com/c/planet-understanding-the-amazon-from-space/leaderboard`
(accessed: 09.12.2020).

[2] Planet: Understanding the amazon from space, 1st place winner's interview.
`https://medium.com/kaggle-blog/planet-understanding-the-amazon-from-space-1st-place-winners-interv`
(accessed: 09.12.2020).

[3] Robert Bindschadler. Monitoring ice sheet behavior from space. *Reviews of Geophysics*, 36(1):79–104, 1998.

[4] Peter Hoefsloot, Amor VM Ines, Jos C van Dam, Gregory Duveiller, Francois Kayitakire, and James Hansen. *Combining crop models and remote sensing for yield prediction: Concepts, applications and challenges for heterogeneous smallholder environments*. 2012.

[5] Thomas Blaschke and Josef Strobl. What's wrong with pixels? some recent developments interfacing remote sensing and gis. *Zeitschrift für Geoinformationssysteme*, pages 12–17, 2001.

[6] Michael J Swain and Dana H Ballard. Color indexing. *International journal of computer vision*, 7(1):11–32, 1991.

[7] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[8] Weixun Zhou, Zhenfeng Shao, Chunyuan Diao, and Qimin Cheng. High-resolution remote-sensing imagery retrieval using sparse features by auto-encoder. *Remote sensing letters*, 6(10):775–783, 2015.

[9] Yi Yang and Shawn Newsam. Bag-of-visual-words and spatial extensions for land-use classification. In *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*, pages 270–279, 2010.

[10] Marco Castelluccio, Giovanni Poggi, Carlo Sansone, and Luisa Verdoliva. Land use classification in remote sensing images by convolutional neural networks. *arXiv preprint arXiv:1508.00092*, 2015.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[12] Andrew Kachites McCallum. Multi-label text classification with a mixture model trained by em. In *AAAI 99 workshop on text learning*. Citeseer, 1999.

[13] Min-Ling Zhang and Lei Wu. Lift: Multi-label learning with label-specific features. *IEEE transactions on pattern analysis and machine intelligence*, 37(1):107–120, 2014.

[14] Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. Cnn-rnn: A unified framework for multi-label image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2285–2294, 2016.

[15] Planet: Understanding the amazon from space, planet dataset.
`https://www.kaggle.com/c/planet-understanding-the-amazon-from-space/data`
(accessed: 23.10.2020).

[16] What is orthorectified imagery?
`https://www.esri.com/about/newsroom/insider/what-is-orthorectified-imagery`
(accessed: 09.12.2020).

[17] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[18] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[19] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*, 2016.

[20] Analysis of deep neural networks.
`https://medium.com/@culurciello/analysis-of-deep-neural-networks-dcf398e71aae`
(accessed: 09.12.2020).

[21] Emanuel Ben-Baruch, Tal Ridnik, Nadav Zamir, Asaf Noy, Itamar Friedman, Matan Protter, and Lihi Zelnik-Manor. Asymmetric loss for multi-label classification. *arXiv preprint arXiv:2009.14119*, 2020.

[22] Thibaut Durand, Nazanin Mehrasa, and Greg Mori. Learning a deep convnet for multi-label classification with partial labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 647–657, 2019.

[23] L. N. Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472, 2017.

[24] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour, 2018.

[25] Stanford cs231n - course project reports: Spring 2017.
`http://cs231n.stanford.edu/2017/reports.html`
(accessed: 09.12.2020).