# Identification of human activities in the Amazon forests

1st Aristi Papastavrou
*DSML*
*NTUA*
Athens, Greece
aristipapastaurou@mail.ntua.gr

2nd Marios Xristopoulos
*DSML*
*NTUA*
Athens, Greece
marioschristopoulos@mail.ntua.gr

3rd Vasilis Sioros
*DSML*
*NTUA*
Athens, Greece
vasileiossioros@mail.ntua.gr

***Abstract*—The Amazon rainforest deforestation is a significant environmental issue that has attracted global attention due to its far-reaching impacts on the planet. Deforestation in the Amazon basin is primarily driven by human activities such as logging, mining, and agriculture, which lead to the clearing of vast areas of forest.In order to tackle the global challenge of rapid deforestation, we use satellite imagery to understand land use patterns and atmospheric conditions. In this project, we apply machine learning algorithms and techniques for performing automated multi-label classification of satellite images of the Amazon basin using a dataset provided by Planet Labs, which is hosting a Kaggle competition for this task.**

## I. Introduction

The Amazon is the largest and most bio-diverse tropical rain forest in the world, covering an area of 2.1 million square miles. It is comprised of an estimated 390 billion individual trees divided into 16,000 species. The Amazon has been referred to as the “lungs of the planet”, as it helps to stabilize the earth’s climate and slow global warming by fixing CO2 and producing 20% of the world’s oxygen.

Since 1978 over 289,000 square miles of Amazon rain forest have been destroyed across Brazil, Peru, Columbia, Bolivia, Venezuela, Suriname, Guyana, and French Guiana. Every minute, the world loses an area of forest the size of 48 football fields. There is concern that the destruction of the forest will result in loss of biodiversity, habitat loss, and the release of the carbon contained within the vegetation, which could accelerate global warming. Better data about the location of deforestation and human encroachment on forests can help governments and local stakeholders respond more quickly and effectively.

The goal of this project is to track changes in the Amazon rain forest due to deforestation using satellite image data. The data has been provided by Planet and hosted on kaggle.com as part of a previous competition Planet: Understanding the Amazon from Space [1]. The labels for this task represent a reasonable subset of phenomena of interest in the Amazon basin. The task is a multi-label image classification problem, where each image will have one and potentially more than one atmospheric label and zero or more common and rare labels.

Fig. 1: A map of the Amazon basin [1]

## II. Related work

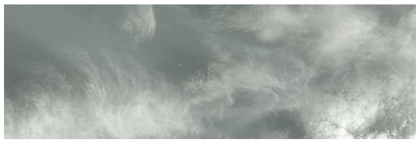
The existing literature on multi-label classification and satellite image classification proved to be very beneficial as it provided us with a baseline framework to build our model off of and some general experimentation ideas. The task performed for the purposes of this current research falls under the category of image classification. The goal of image classification is to train a machine learning model to recognize different patterns or features in the image, and to associate those patterns with specific classes or labels. As Convolutional neural networks (CNNs) are the most commonly used type of model for image classification, due to their ability to learn hierarchical features from the raw pixel values of the input image, we conducted extensive research on different architectures of cnns, transfer learning, data augmentation etc, in order to create our models and provide well thought out experiments.

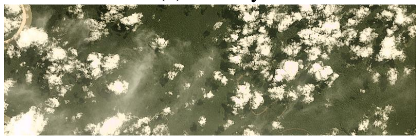
## III. Dataset and Features

The dataset consists of 40,479 training images and 61,191 test images, and each image is a 256x256 sized jpeg. The file train_v2.csv is included which lists the training file names and their accompanying labels. The labels can be broken down into three categories: cloud cover labels, common labels, and less common labels. There are 17 labels in total: clear, partly cloudy, cloudy, haze, primary, water, habitation, agriculture, road, cultivation, bare ground, slash and burn, selective logging, blooming, conventional mining, artisanal mining, and blow down.

### A. Cloud Cover Labels

Clouds are a major challenge for satellite imaging, and daily cloud cover and rain showers in the Amazon basin can significantly complicate monitoring in the area. These labels closely mirror what one would see in a local weather forecast: clear, partly cloudy, cloudy, and haze. Haze is defined as any chip where atmospheric clouds are visible but not so opaque as to obscure the ground. Clear scenes show no evidence of clouds, and partly cloudy scenes can show opaque cloud cover over any portion of the image. Cloudy images have 90% of the chip obscured by opaque cloud cover.


(a) Cloudy


(b) Partly Cloudy

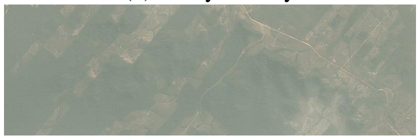

(c) Haze

Fig. 2: Examples of Cloud Cover Labels [1]

### B. Common labels

We show below some examples of common labels in this data set such as: rain forest, agriculture, rivers, towns/cities, and roads.

*Primary Rain forest:* The overwhelming majority of the data set is labeled as ”primary”, which is shorthand for primary rain forest. Generally speaking, the ”primary” label was used for any area that exhibited dense tree cover.

*Water Labels: Rivers & Lakes:* Rivers, reservoirs, and oxbow lakes are important features of the Amazon basin, and we used the water tag as a catch-all term for these features.

*Habitation:* The habitation class contains labels that appeared to contain human homes or buildings. This includes anything from dense urban centers to rural villages along the banks of rivers. Small, single-dwelling habitations are often difficult to spot but usually appear as clumps of a few pixels that are bright white.

*Agriculture:* Commercial agriculture, is also a major driver of deforestation in the Amazon. For the purposes of this dataset, agriculture is considered to be any land cleared of trees.

*Road:* Roads are important for transportation in the Amazon but they also serve as drivers of deforestation. For our data, all types of roads are labeled with a single ”road” label. Some rivers look very similar to smaller logging roads, and consequently there may be some noise in this label.

*Cultivation:* Shifting cultivation is a subset of agriculture that is very easy to see from space, and occurs in rural areas where individuals and families maintain farm plots for subsistence.

*Bare Ground:* Bare ground is a catch-all term used for naturally occurring tree free areas that aren’t the result of human activity.

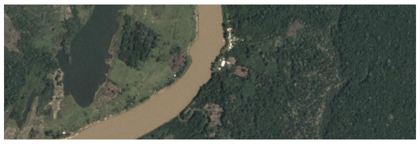
(a) Primary Rain forest

(b) River

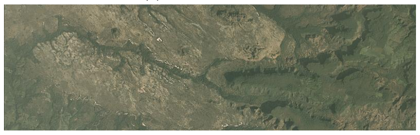
(c) Habitation

(d) Agriculture

(e) Road



(f) Cultivation



(a) (g) Bare Ground

Fig. 3: Examples of Common Labels [1]

### C. Less Common Labels

*Slash and Burn:* Slash-and-burn agriculture can be considered to be a subset of the shifting cultivation label and is used for areas that demonstrate recent burn events and appear to have dark brown or black areas.

*Selective Logging:* The selective logging label is used to cover the practice of selectively removing high value tree species from the rain forest (such as teak and mahogany). From space this appears as winding dirt roads adjacent to bare brown patches in otherwise primary rain forest.

*Blooming:* Blooming is a natural phenomenon found in the Amazon where particular species of flowering trees bloom, fruit, and flower at the same time to maximize the chances of cross pollination.

*Mining: Conventional and Artisinal:* There are a number of large conventional or Artisinal mines in the Amazon basin and the number is steadily growning.

*Blow Down:* Blow down, also called windthrow, is a naturally occurring phenomenon in the Amazon. Briefly, blow down events occur during micro bursts where cold dry air from the Andes settles on top of warm moist air in the rain forest.

A histogram of the label occurrences shows a distribution of the labels, with most frequent label being ‘primary’:

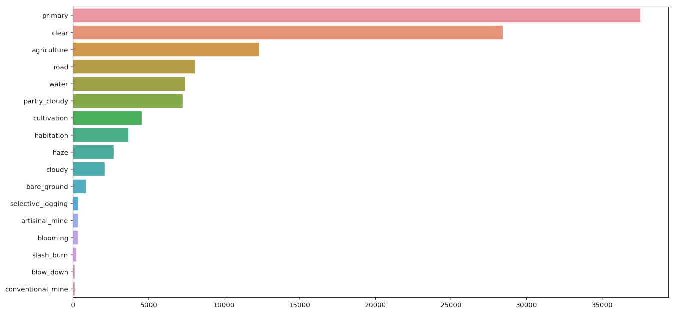
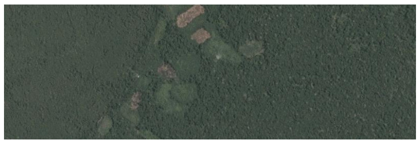
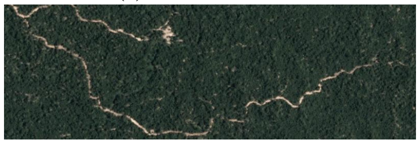


Fig. 5: Histogram of label occurrences



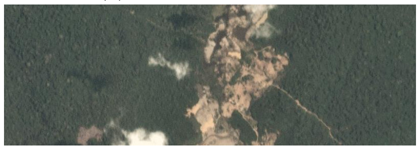
(a) Slash and Burn



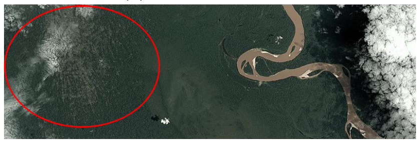
(b) Selective Logging

(c) Bloom

(d) Conventional Mine



(e) Artisinal Mine



(f) Blowdown

Fig. 6: Examples of Less Common Labels [1]

## IV. Methods

### A. Conventional Machine Learning Approaches

We first attempted to tackle the problem using conventional machine-learning techniques using the **sklearn** [2] machine-learning framework. Given the size of the data set, we limited our options to models that supported mini-batch learning. We focused on the linear SVM and Passive-Aggressive classifier models.

Support Vector Machines (SVMs) [3] are a type of supervised learning algorithm used in classification and regression analysis. SVMs work by finding the best possible boundary, or hyperplane, that can separate data points in a high-dimensional space into two or more classes. The basic idea of SVMs is to map the input data to a high-dimensional feature space where it can be separated by a hyperplane. The hyperplane is chosen so that it maximally separates the data points of different classes, i.e., it maximizes the margin between the closest points of each class. The points closest to the hyperplane are called support vectors, hence the name Support Vector Machines. For linear SVMs, the decision boundary is given by:

$$w^T x + b = 0 \tag{1}$$

where $w$ is the weight vector and $b$ is the bias term. The objective function for the linear SVM can be formulated as:

$$\begin{aligned} &\min_{w,b} \frac{1}{2}||w||^2 \\ &\text{s.t. } y_i(w^T x_i + b) \geq 1 \text{ for } i = 1, \dots, n \end{aligned}$$

where $||w||^2$ is the L2 norm of the weight vector, $y_i$ is the class label of the $i$-th data point, and $x_i$ is the feature vector of the $i$-th data point.

The Passive-Aggressive Classifier [4] is a type of online learning algorithm used in machine learning to solve binary classification problems. In this algorithm, the model updates its weight vector based on the observed data in an iterative manner. The "passive-aggressive" name comes from the algorithm's behavior when it encounters misclassified examples. If a new observation is misclassified, the algorithm "aggressively" updates its weight vector to correct the mistake. In contrast, if the observation is correctly classified, the algorithm "passively" does not make any changes to the weight vector. More specifically, the algorithm's weight vector is updated using a formula that depends on the degree of misclassification and the magnitude of the feature vector. The magnitude of the feature vector is used as a regularizer to prevent the weight vector from becoming too large.

$$\begin{aligned} w_{t+1} = \arg\min_{w} \frac{1}{2}|w - w_t|^2 + C \cdot \xi_t \\ \text{s.t. } y_t \cdot w^\intercal x_t \leq \xi_t \\ \xi_t \geq 0 \end{aligned}$$

In this formulation, $w$ is the weight vector being learned, $t$ is the iteration number, $C$ is a hyperparameter that controls the trade-off between maximizing the margin and minimizing the loss, $\xi_t$ is a slack variable that allows for some margin violations, $x_t$ is the input vector at iteration $t$, and $y_t$ is the true label associated with $x_t$. The first line of the formulation defines the update rule for $w$ at iteration $t+1$, which seeks to minimize the distance between $w$ and the current weight vector $w_t$ while penalizing any margin violations by a factor of $C \cdot \xi_t$. The second line specifies the constraints on the optimization problem: $y_t \cdot w^\intercal x_t$ should be less than or equal to $\xi_t$ to ensure that the margin is at least 1, and $\xi_t$ should be non-negative to prevent it from canceling out the penalty term in the objective function.

We experimented with processing the raw image data as well as extracting their respective (Histogram of Oriented Gradients) HOG [5] features. HOG features are a type of feature descriptor used in computer vision to detect and describe the local texture, shape, and edges in an image. HOG features work by calculating the gradient of intensity in small regions of an image, and then grouping the gradient information into histograms. The resulting histograms represent the distribution of gradient orientations and magnitudes within each region of the image. HOG features have been used in a wide range of applications, including object detection, pedestrian detection, and face recognition, and have been shown to be robust to variations in lighting and scale.

We additionally experimented with multiple online dimensionality reduction techniques such as (1) Incremental PCA [6], (2) Mini-batch K-Means Clustering [7], and (3) Online Dictionary Learning [8]. Incremental PCA, dictionary learning, and minibatch K-means are three techniques commonly used in machine learning and computer vision. Incremental PCA is a technique used to perform principal component analysis on large datasets by processing the data in small batches. Dictionary learning is a method of sparse coding that involves learning a set of basis vectors that can represent a given set of signals. Minibatch K-means is a variant of the K-means clustering algorithm that processes small random subsets, or "minibatches," of the input data at a time. These techniques have been widely used in various applications, such as image and speech processing, and can help to reduce the computational cost and memory requirements of traditional algorithms.

Our pipeline consisted of (1) normalizing by removing the mean and scaling to unit variance, (2) applying (or not) one of the aforementioned dimensionality reduction techniques and (3) utilizing one of the aforementioned mini-batch classification approaches. Given that both the linear SVM as well as the Passive-Aggressive classifiers do not natively support multi-target classification, we had to fit 17 separate classifier instances to the data set, one for each element of the 17-element long one-hot encoded vector.

### B. Convolutional Neural Networks (CNNs)

Convolutional neural networks (CNNs) are a class of deep neural networks that have been particularly successful in computer vision tasks such as image classification, object detection, and segmentation. Unlike traditional neural networks, CNNs are designed to recognize spatial patterns in data, such as images, by applying convolutional filters to the input. These filters are learned during the training process and can capture features of different scales and complexities, from simple edges and corners to more complex structures such as eyes, noses, and faces. By stacking multiple convolutional layers, CNNs can learn increasingly abstract representations of the input data, leading to state-of-the-art performance in a variety of visual recognition tasks. Moreover, the use of pooling layers and nonlinear activation functions allows CNNs to achieve translation invariance and robustness to small variations in the input, making them suitable for real-world applications.

CNN's can be described using the below formula: Let $X$ be an input image of dimensions $W_1 \times H_1 \times C_1$, where $W_1$ is the width, $H_1$ is the height, and $C_1$ is the number of channels. The output of the $k$-th convolutional layer is a set of $C_k$ feature maps, each of dimensions $W_k \times H_k \times 1$, where $W_k$ and $H_k$ are the width and height of the output feature maps, respectively. The $i$-th element of the $j$-th feature map in the $k$-th layer is given by:

$$y_{ijk} = f\left(\sum_{l=1}^{C_{k-1}} \sum_{p=1}^{W_f} \sum_{q=1}^{H_f} w_{lpqk} x_{i+p-1,j+q-1,l}\right) \tag{2}$$

where $f$ is an activation function, such as ReLU, $w_{lpqk}$ is the weight of the $p$-th row, $q$-th column, and $l$-th channel of the filter in the $k$-th layer, and $x_{i+p-1,j+q-1,l}$ is the $l$-th channel of the $(i+p-1, j+q-1)$-th pixel in the input image. The filter has dimensions $W_f \times H_f \times C_{k-1}$ and is convolved with the input image using a stride of 1 and padding of 0. The bias term is typically added after the activation function, and the output is then passed through a pooling layer to reduce its size.

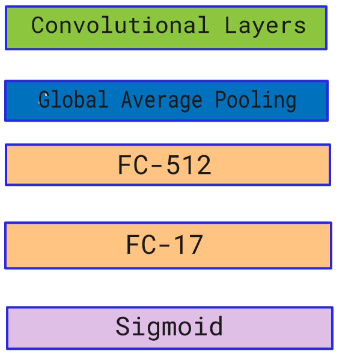


Fig. 7: CNN architecture

In our experiments we created cnns (using both PyTorch and Keras api's), whose architecture consists of 4 (Conv-BN-ReLU) layers with 2 Max Pooling layers followed by two (FC-BN-ReLU) layers with 512 and 17 neurons. The Conv layers use 3x3 filters with stride 1. The number of filters progressively increases as we go down the network with the first Conv layer containing 32 filters and the final one containing 64 filters. We experimented with this model using different input images with 3-channels, while for the training process we used 10 epochs, dataloaders and the Adam optimizer with 0.001 learning rate.

### C. Transfer Learning

Transfer learning [9] is a machine learning technique where a pre-trained model is used as a starting point for a new model. It can save time and computational resources and has been widely used in various applications such as computer vision and natural language processing.

For our purposes, we utilized an 18 layers deep pre-trained ResNet model (ResNet18). ResNet is a neural network architecture that was introduced in 2015 [10] to enable the creation of very deep neural networks while maintaining good performance. The key innovation of ResNet is the use of residual blocks, which consist of two convolutional layers with a shortcut connection that bypasses the second convolutional layer. This allows the network to learn residual functions, i.e., the difference between the input to the block and the output of the block, which helps to mitigate the problem of vanishing gradients in very deep networks.

ResNet18 requires that the input dimensions are multiples of 32 and given that the input images are of size $256 \times 256$, the closest multiple of 32 is 224. We, therefore, re-scaled our input data to conform to this restriction. We additionally normalized our input images based on the pre-trained mean and standard deviation intensity values of the model. During training, we horizontally flip and rotate our input images at random, in order to avoid overfitting to the specific orientation of the data. Our target values are the 17-element long vectors resulting from the one-hot encoding of the tags of each image. We replaced the final average pooling layer as well as the final feed-forward layer of the original architecture in order to conform to the expected 17-element long vector output. More specifically, we utilized an adaptive average pooling layer with an output size of $1 \times 1$. Instead of the default feed-forward layer, we utilized two feed-forward layers $512 \times 128$ and $128 \times 17$ large in order of appearance. Between these two layers, a ReLU [11] activation function as well as a dropout layer [12] with a dropout probability of 0.2. During inference, the final output of our model is passed through a Sigmoid [13] function in order to squash it in the range 0 to 1. These values are then compared with a per-class pre-specified threshold in order to construct the 17-element long one-hot encoded vector prediction.

## V. Experiments/Results/Discussion

Our evaluation metrics consist of accuracy, precision, recall, and the F1 score (samples). Accuracy measures the percentage of correct predictions made by the model. It is calculated as the ratio of the number of correctly predicted instances to the total number of instances in the dataset. Precision measures the percentage of correctly predicted positive instances out of all instances predicted as positive. It is calculated as the ratio of true positives (TP) to the sum of true positives and false positives (FP). Recall measures the percentage of correctly predicted positive instances out of all actual positive instances.

It is calculated as the ratio of true positives (TP) to the sum of true positives and false negatives (FN). The F1 score (samples) is the weighted average of F1 scores calculated for each instance. It gives more weight to instances that belong to minority classes. It is calculated as the average of F1 scores calculated for each instance. The generic formula of the F1 score is as follows:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (3)$$

### A. *Conventional Machine Learning Approaches*

It became immediately apparent that the linear SVM was not able to sufficiently model the data. Extracting and processing the HOG features corresponding to each input image instead of its raw data resulted in a slight increase in performance. Employing dimensionality reduction techniques only worsened the performance of our pipeline. We concluded that processing the HOG image features using the Passive-Aggressive classifier approach without applying any dimensionality reduction resulted in the best possible performance. The aforementioned configuration outperformed all other configurations across all evaluation metrics. It's obvious from "Fig. I" that the performance of our pipeline was not extraordinary. Thus, we decided to explore alternative approaches such as the utilization of Convolutional Neural Networks and Transfer Learning techniques.

TABLE I: The Evaluation Metrics Corresponding To Our Best Performing Combination Of Input Features, Dimensionality Reduction Techniques And Classification Models.

| **Accuracy** | **Precision** | **Recall** | **F1 (samples)** |
|---|---|---|---|
| 0.153 | 0.342 | 0.401 | 0.478 |

### B. *Convolutional Neural Networks (CNNs)*

The second part of the experimental procedures for this projects include the development of CNN models using PyTorch and Keras api's, experimenting with different architectures and layers for each model type, optimizers, loss functions etc. To train CNNs for image classification, pre-processing techniques such as data augmentation , normalization, and image resizing are commonly used. Optimization techniques such as stochastic gradient descent (SGD), AdaGrad [14], and Adam [15] are used to update the weights of the network during training.

TABLE II: Satistics for CNN's (PyTorch and Keras implementation)

| **DL API** | **Metrics** | | |
|---|---|---|---|
| | ***accuracy*** | ***f1-score*** | ***loss*** |
| Keras | 0.95 | 0.82 | 0.13 |
| Pytorch | 0.78 | 0.69 | 0.22 |

As it can be easily observed from the statistics, CNN architecture, implemented with the help of keras api has better results for our data. Although in both cases we used the same architecture (layer wise), keras due to its easier gui, was easier to tune. So therefore the difference in these two implementations could possibly be due to need of better layer tuning on the pytorch api, different activation functions etc.

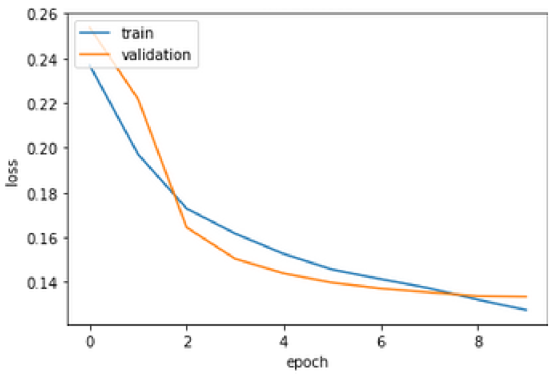


Fig. 8: Training and Validation data loss per epoch for CNN with keras

We also trained our classification model using SGD Nesterov momentum update rule [16] (default learning rate = 0.01, momentum = 0.9) and observed that it resulted in a higher final loss (train loss: 0.2401, validation loss: 0.248) than the model that relied on the Adam update rule. Nesterov's Accelerated Gradient is a clever variation of momentum that works slightly better than standard momentum. The idea behind Nesterov's momentum is that instead of calculating the gradient at the current position, we calculate the gradient at a position that we know our momentum is about to take us, called as "look ahead" position. From physical perspective, it makes sense to make judgements about our final position based on the position that we know we are going to be in a short while.

The implementation makes a slight modification to standard SGD Momentum by nudging our parameters slightly in the direction of the velocity and calculating the gradients there. Ultimately, even further hyperparamter tuning did not yield better results and therefore, we decided to use the Adam update rule for all of our subsequent experiments.

Finally the last experiment that we conducted was Data augmentation. Data augmentation is a technique for generating new training examples by applying transformations such as rotation, cropping, and scaling to the original images. Some of the strategies that we implemented (either solely or in combination with other techniques) where:

Flipping: Flipping the image horizontally or vertically can create a new training example that is similar to the original image but has a different orientation.

Rotation: Rotating the image by a certain angle can create a new example that represents the same object but from a different perspective.

Scaling: Rescaling the image by a certain factor can create a new example that represents the same object but at a different size.

After applying these techniques on the existing dataset, we created new dataloaders, containing the now modified data. However incorrect implementation of the augmentation techniques or their parameters can also lead to poor performance. Due to the high quality of the images of our dataset, applying excessive image transformations or using inappropriate ranges of parameters such as rotation angle or scaling factor can lead to poor results which we noted while trying to tune the new images.

Although neither of the last two techniques, seemed to have any positive effect on the cnn keras model accuracy, we still succeeded , with the help of finetuning and experimenting on architectural variations to achieve high statistical results.

### C. Transfer Learning

We trained our model on the entirety of the training data set for 12 epochs, using the Adam [15] optimizer with a learning rate of 0,001, an epsilon of $1\mathrm{e}{-08}$, and a weight decay of 0,01. We employed an early stopping strategy [17] with a tolerance of 5 and a minimum $\Delta$ of 0,001. We additionally employed a learning rate schedule that decays the learning rate by 0,1 every 5 epochs. These precautions were taken in order to avoid overfitting our model to the data set. The early-stopping criterion was met after only 12 out of 15 total epochs. As seen from “Fig. 9” our model was capable of closely modeling the training data. Additionally, it performed notably well on the validation keep-aside set.

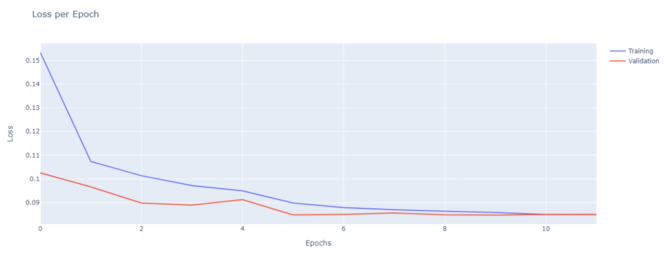


Fig. 9: The learning curves corresponding to our ResNet18 model.

As previously mentioned during inference, we utilize per-class thresholds in order to construct the prediction vector. In order to determine the appropriate per-class threshold values, we constructed a population of 2500 17-long real-valued vectors which were sampled from a Continuous Uniform Distribution of range (0, 0.5) and evaluated our model on the validation set for each threshold vector. Comparing the resulting F1 (samples) scores we concluded that the threshold values resulting in the best-performing model are 0.316, 0.417, 0.420, 0.409, 0.036, 0.235, 0.335, 0.064, 0.449, 0.286, 0.314, 0.119, 0.359, 0.371, 0.094, 0.304, and 0.444. These values resulted in an F1 (score) of 0.91 on the validation set.

We finally evaluated our model on the test data set wherein our model achieves an F1 (samples) score of 0.82. This by no means corresponds to state-of-the-art results but it is impressive nonetheless. The confusion matrices of our model on the validation and the test data set are shown in “Fig. 10” and “Fig. 11” respectively. As we can see our model is mostly capable of efficiently distinguishing different categories. It performs extremely well on some, for instance, distinguishing images labeled as *Cloudy*, while suffering bad performance mostly when it comes to distinguishing ones labeled as *Agriculture*, *Road*, and *Water*.

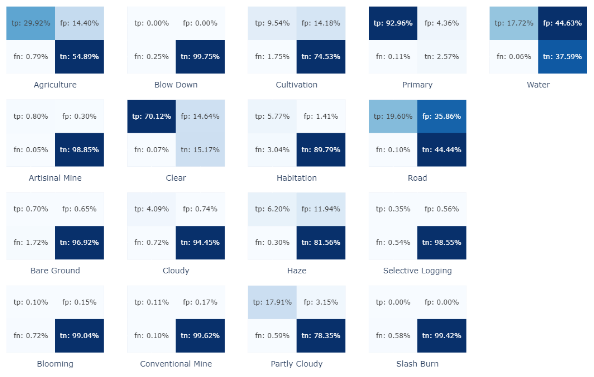


Fig. 10: The confusion matric of our ResNet18 model on the validation set.

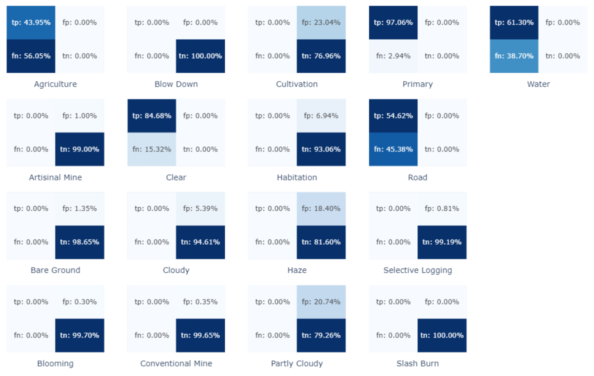


Fig. 11: The confusion matric of our ResNet18 model on the test set.

## VI. Conclusion/Future Work

We determined that given the complexity of the task at hand conventional machine-learning techniques would not be able to efficiently model the data. As a result, we experimented with CNNs and Transfer Learning techniques which resulted in great performance increases. Our methodology does not supersede state-of-the-art approaches but results in noteworthy performance.

We would like to experiment with different architectures such as 34, 50, 101, and 152 layers deep ResNet models as well as other completely different architectures such as VGG [18]. Additionally, we would like to look further into why our model fails to properly model the *Agriculture*, *Road*, and *Water* classes and how we would be able to mitigate that issue. Finally, of major interest to us is whether any preprocessing of the data such as extracting their corresponding HOG features as in the conventional approach case would result in a performance increase. Any further updates will be posted on our github repo [19]

## References

[1] B. Goldenberg, B. Uzkent, C. Clough, D. Funke, D. Desai, grischa, JesusMartinezManso, K. Scott, M. Risdal, M. Ryan, Pete, R. Holm, R. Nair, S. Herron, T. Stafford, and W. Kan, "Planet: Understanding the amazon from space," 2017. [Online]. Available: https://kaggle.com/competitions/planet-understanding-the-amazon-from-space

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[4] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *Journal of Machine Learning Research*, vol. 7, pp. 551–585, 03 2006.

[5] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, pp. 886–893 vol. 1.

[6] D. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *International Journal of Computer Vision*, vol. 77, pp. 125–141, 05 2008.

[7] D. Sculley, "Web-scale k-means clustering," in *Proceedings of the 19th International Conference on World Wide Web*, ser. WWW '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 1177–1178. [Online]. Available: https://doi.org/10.1145/1772690.1772862

[8] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 689–696. [Online]. Available: https://doi.org/10.1145/1553374.1553463

[9] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," *CoRR*, vol. abs/1808.01974, 2018. [Online]. Available: http://arxiv.org/abs/1808.01974

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[11] A. F. Agarap, "Deep learning using rectified linear units (relu)," *CoRR*, vol. abs/1803.08375, 2018. [Online]. Available: http://arxiv.org/abs/1803.08375

[12] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, p. 1929–1958, jan 2014.

[13] J. Han and C. Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning," in *Proceedings of the International Workshop on Artificial Neural Networks: From Natural to Artificial Neural Computation*, ser. IWANN '96. Berlin, Heidelberg: Springer-Verlag, 1995, p. 195–201.

[14] A. F. Agarap, "Liao,luofeng and shen,li and duan,jia and kolar,mladen and tao,dacheng," *CoRR*, vol. abs/2106.10022, 2021. [Online]. Available: https://arxiv.org/abs/2106.10022

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[16] J. Brownlee, "Gradient descent with nesterov momentum from scratch," 2021. [Online]. Available: https://machinelearningmastery.com/gradient-descent-with-nesterov-momentum-from-scratch/

[17] Y. Yao, L. Rosasco, and A. Caponnetto, "On early stopping in gradient descent learning," *Constructive Approximation*, vol. 26, no. 2, pp. 289–315, 2007.

[18] S. Liu and W. Deng, "Very deep convolutional neural network based image classification using small training sample size," in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, 2015, pp. 730–734.

[19] A. Papastavrou, V. Sioros, and M. Christopoulos, "Identification-of-human-activities-in-the-amazon-forests," 2023. [Online]. Available: https://github.com/AristiPap/Identification-of-human-activities-in-the-Amazon-forests