Detecting Palm Oil Plantations Using Neural Networks

NEWTON TRAN

What is palm oil and why?

- *Elaeis guineensis* or the African oil palm
  - Native to Africa but was brought over to Southeast Asia over a century a go as ornamental (decorative) tree crop
- Highly versatile because:
  - Semi-solid at room temperature, keeps its spreadability
  - Resistant to oxidation → longer product shelf life
  - Relatively high smoke point of 235 °C (or 455 °F)
  - Odorless and colorless
  - Highly-efficient and high-yielding crop, doesn't require much land
- About 50% of packaged products contain it
- Over 85% of the global supply is produced in Indonesia and Malaysia (Thailand, Columbia, Nigeria, etc. also)

Consequences of Palm Oil

- Between 2000 to 2018, palm oil accounted for 7 percent of global deforestation
  - Decrease in biodiversity
  - More carbon-rich peat soils, which emit millions of tons of carbon dioxide → further contributing to climate change

Problem Statement

- Given a dataset of satellite images, can we build a model that utilizes neural networks to detect the presence of palm oil plantations?

Data

- Women in Data Science (WiDS) Datathon 2019
- Three .zip folders and their corresponding .csv annotation files:
  - `train_images.zip`, the images used for training
  - `leaderboard_holdout_data.zip`, the images used for the competition's submission / ranking
  - `leaderboard_test_data.zip`, the images used for the competition's submission / ranking, like above
  - `traininglabels.csv`, the annotations that correspond to training images
  - `holdout.csv`, the annotations that correspond to the holdout images
  - `testlabels.csv`, the annotations that correspond to the test images

Data

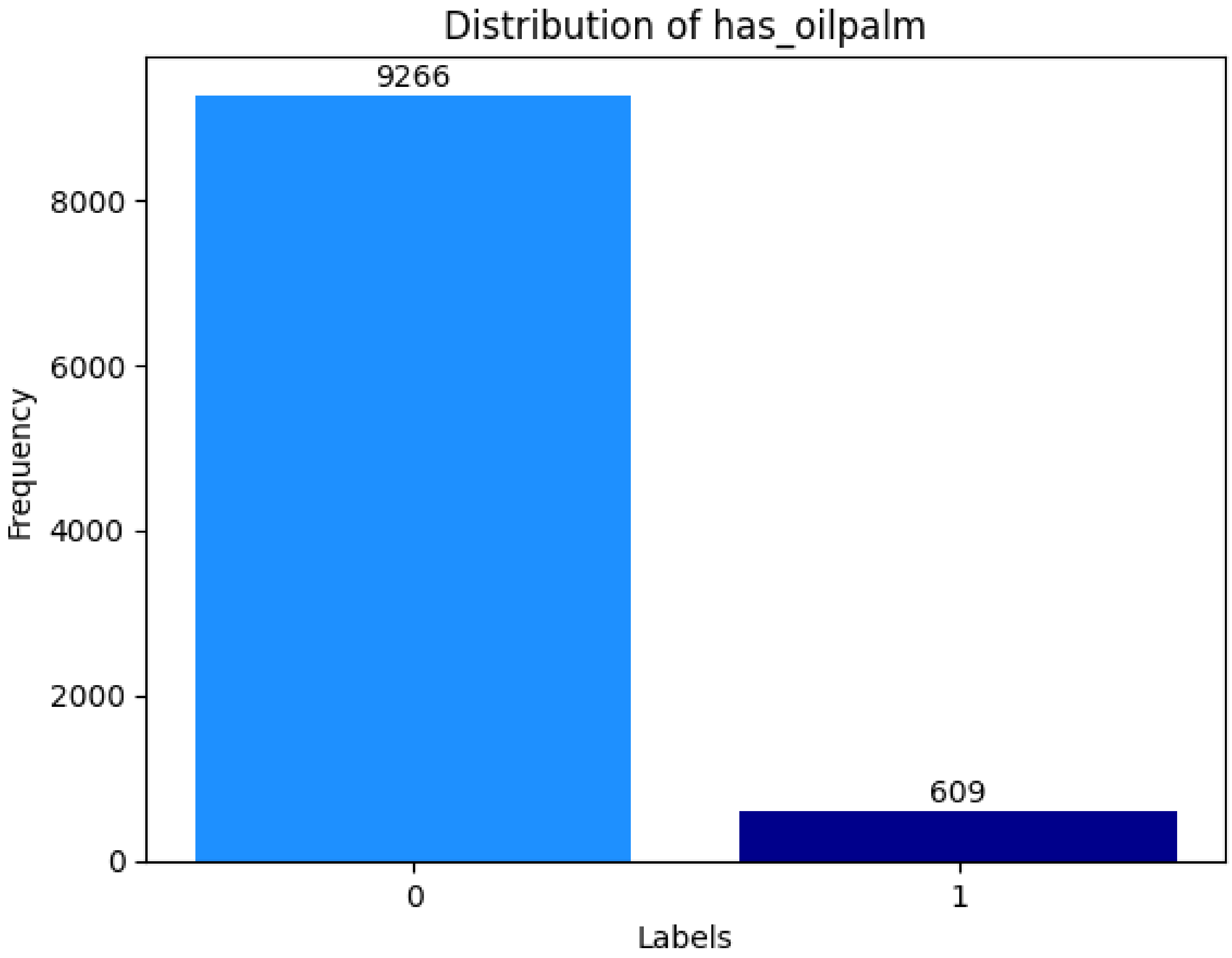
- For each of the annotations, there are three features:
  - `image_id`, the filename of the image
  - `has_oilpalm`, where 0 indicates no presence of a palm oil plantation, 1 indicates otherwise (i.e., the image’s label)
  - `score`, which indicates the likelihood of the image’s label holding true, where values closer to 1 indicate high likelihood
- More images were added after the end of the competition but were not officially documented
  - Test dataset was unusable since there were significantly more images than those listed in the corresponding annotations, and they deviated quite extremely
  - As a result, this model utilizes the training and holdout datasets

Data Preview

	image_id	has_oilpalm	score
0	img_000002017.jpg	0	0.7895
1	img_000012017.jpg	0	1.0000
2	img_000022017.jpg	0	1.0000
3	img_000072017.jpg	0	1.0000
4	img_000082017.jpg	0	1.0000

Exploratory Data Analysis

- First critical thing to address is mismatch between the actual filenames and those listed in the annotations
  - Particularly, remove the 2017 and 2018 appended to the end of the filenames listed in the annotations
  - Subsequent duplicate filenames were created, so any duplicates were removed immediately
  - The annotations were further filtered to include only images whose filenames exactly matched



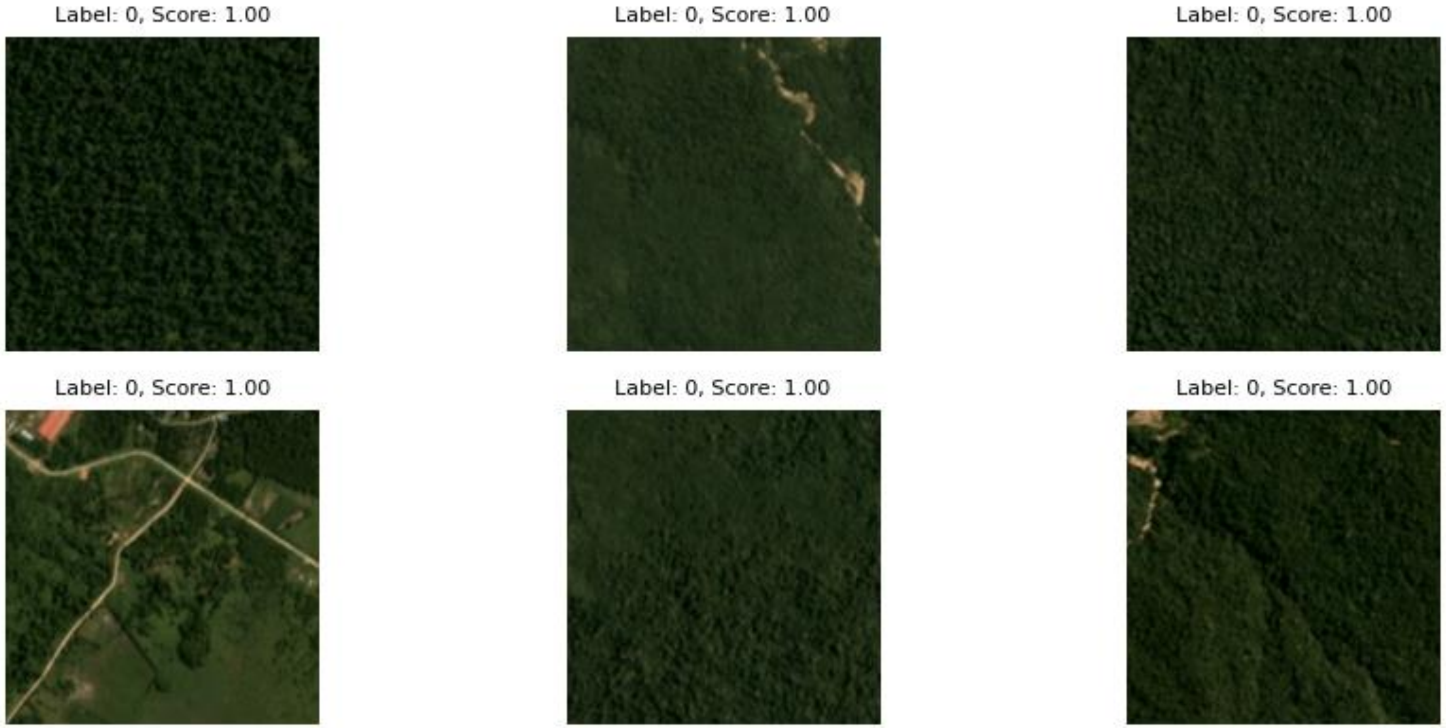
Distribution of the Training Labels, `has_oilpalm`

- After initial preliminary cleaning, we observe that the training labels were highly imbalanced
  - 94 percent of negative class, whereas the remaining 6 percent of positive class

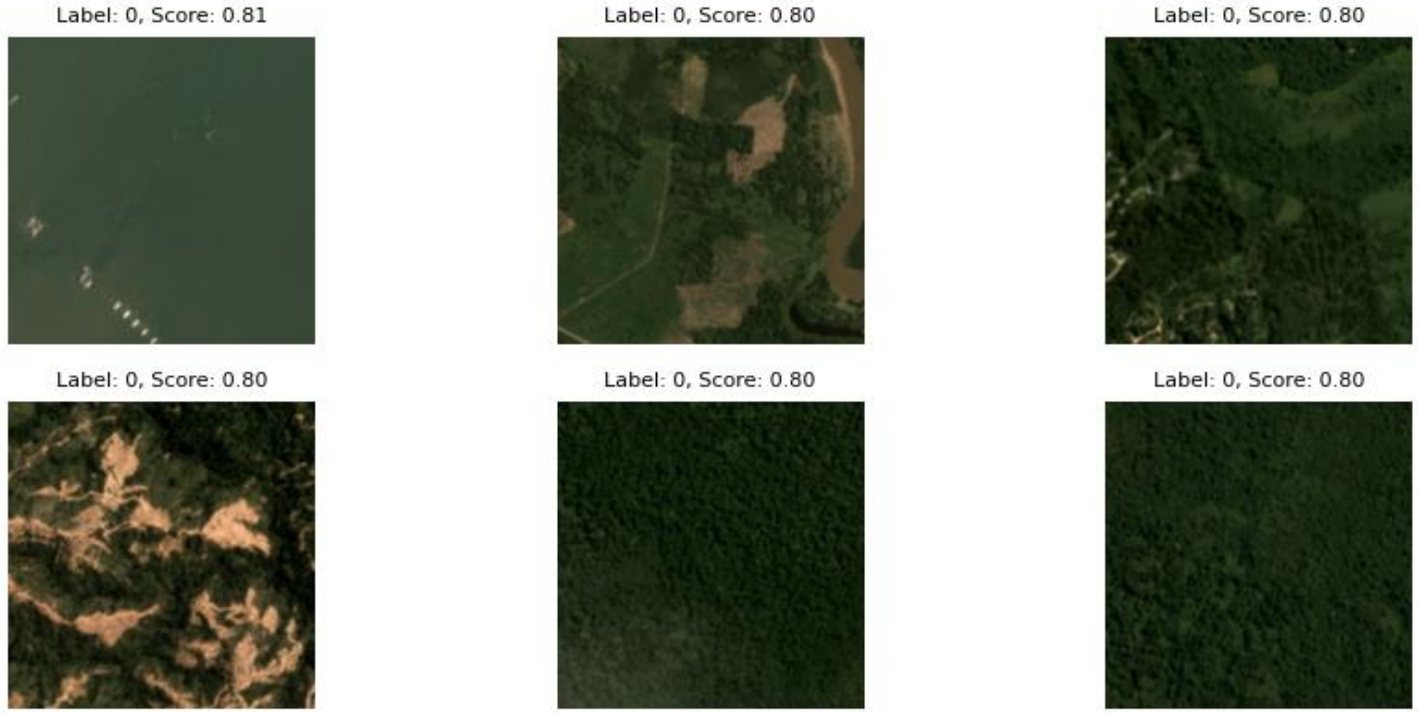
Distribution of **`has_oilpalm`**, Relative to **`score`**

- Among the negative class images, 83 percent of them contain a `score` of 1, indicating high likelihood of no palm oil plantation
- We further investigate how `score` differs by looking at different thresholds:
  - `score` equals 1
  - `score` is between 0.7 inclusive and 1 non-inclusive
  - `score` is between 0.5 inclusive and 0.7 non-inclusive
  - `score` is below 0.5

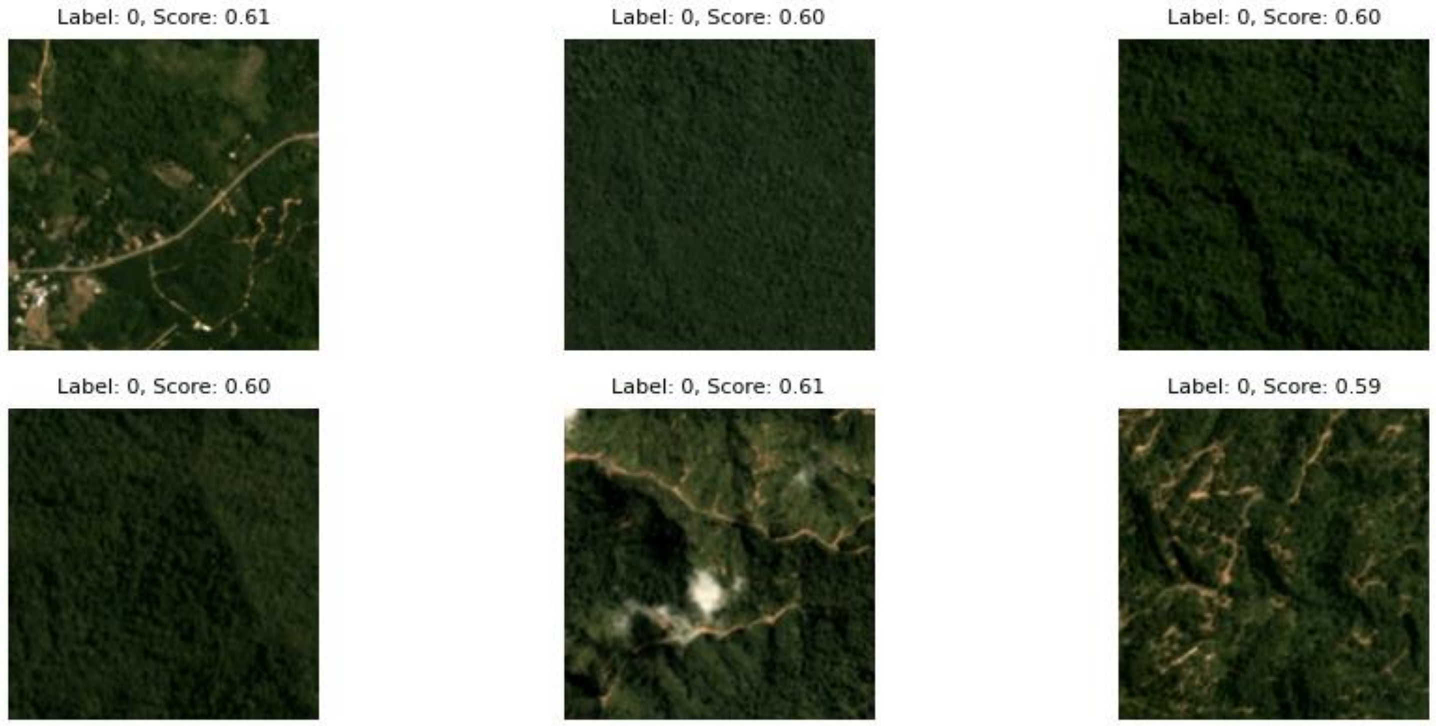
`score` equals 1



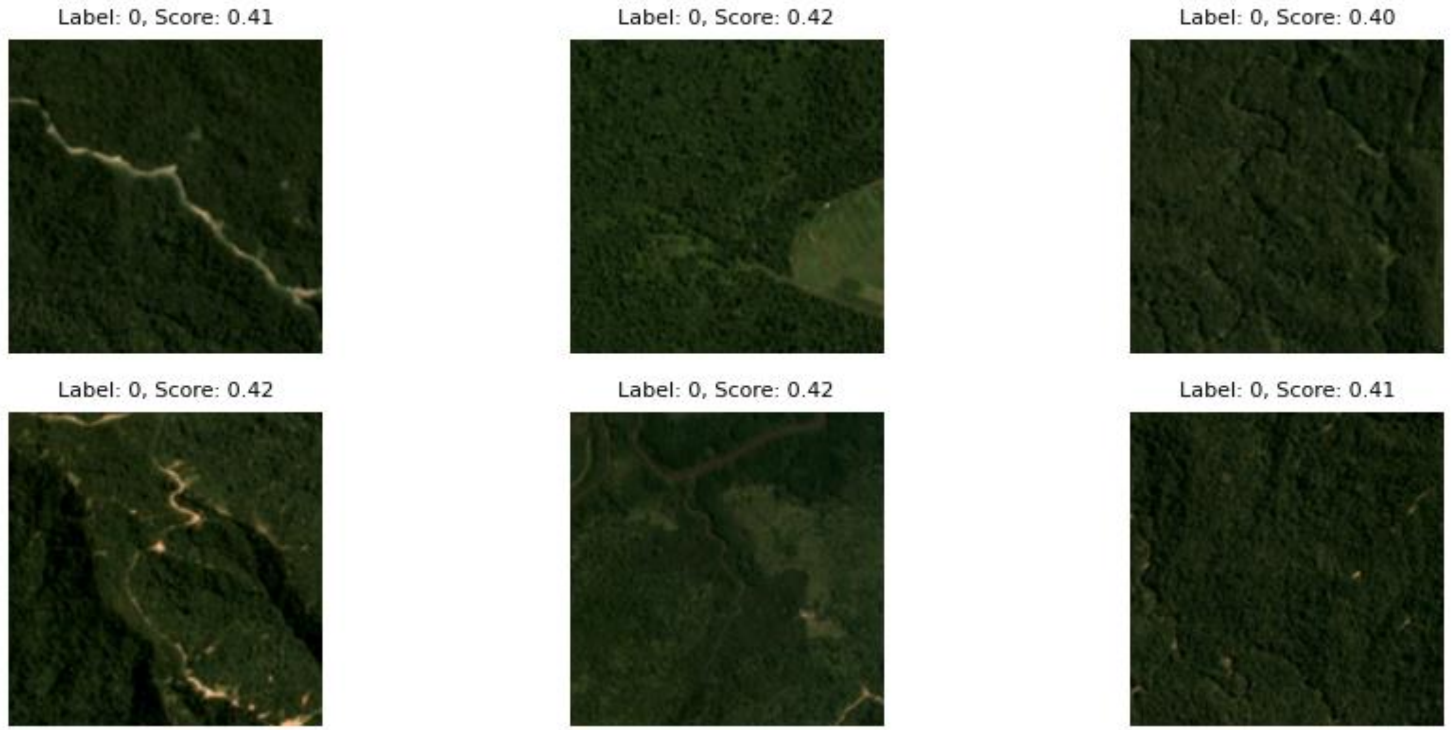
`score` is between 0.7 inclusive and 1 non-inclusive



`score` is between 0.5 inclusive and 0.7 non-inclusive



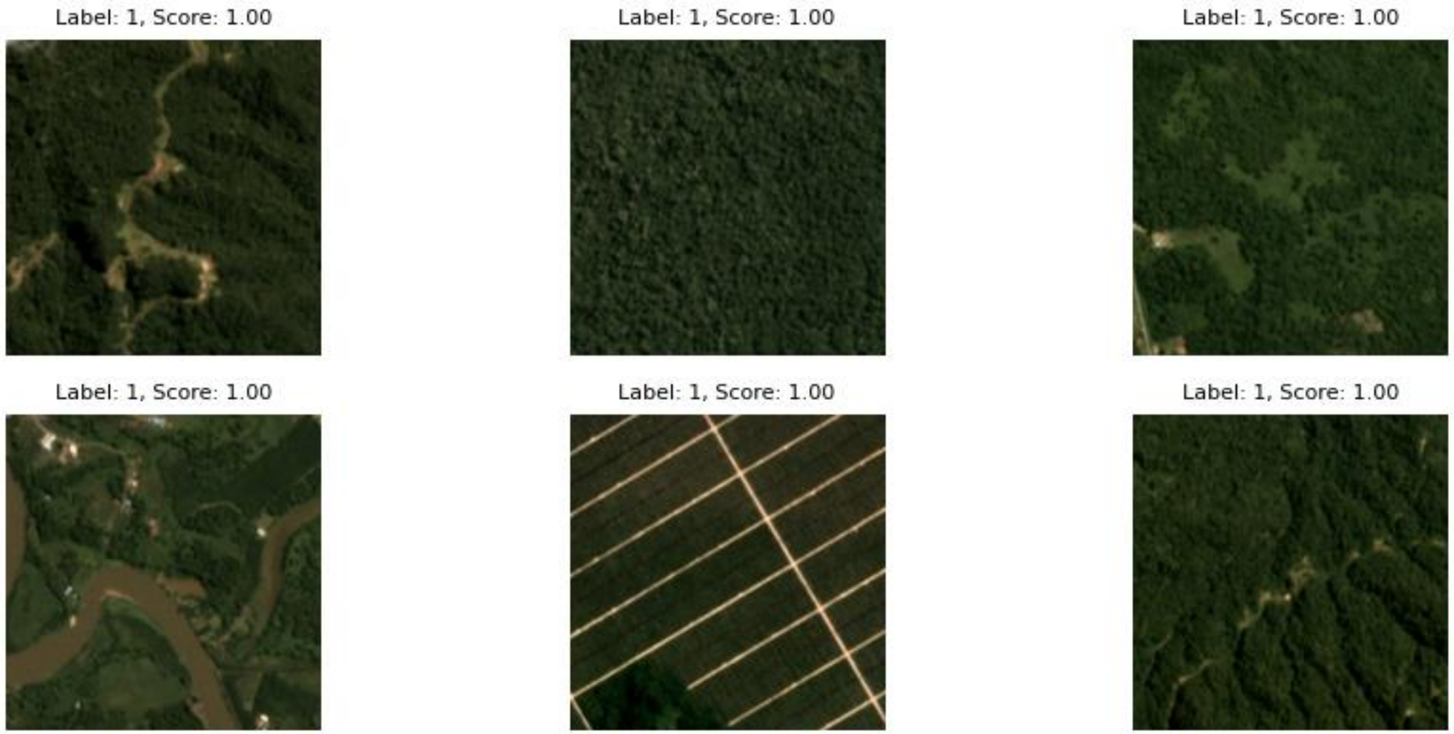
`score` is below 0.5



Distribution of `has_oilpalm`, Relative to `score`

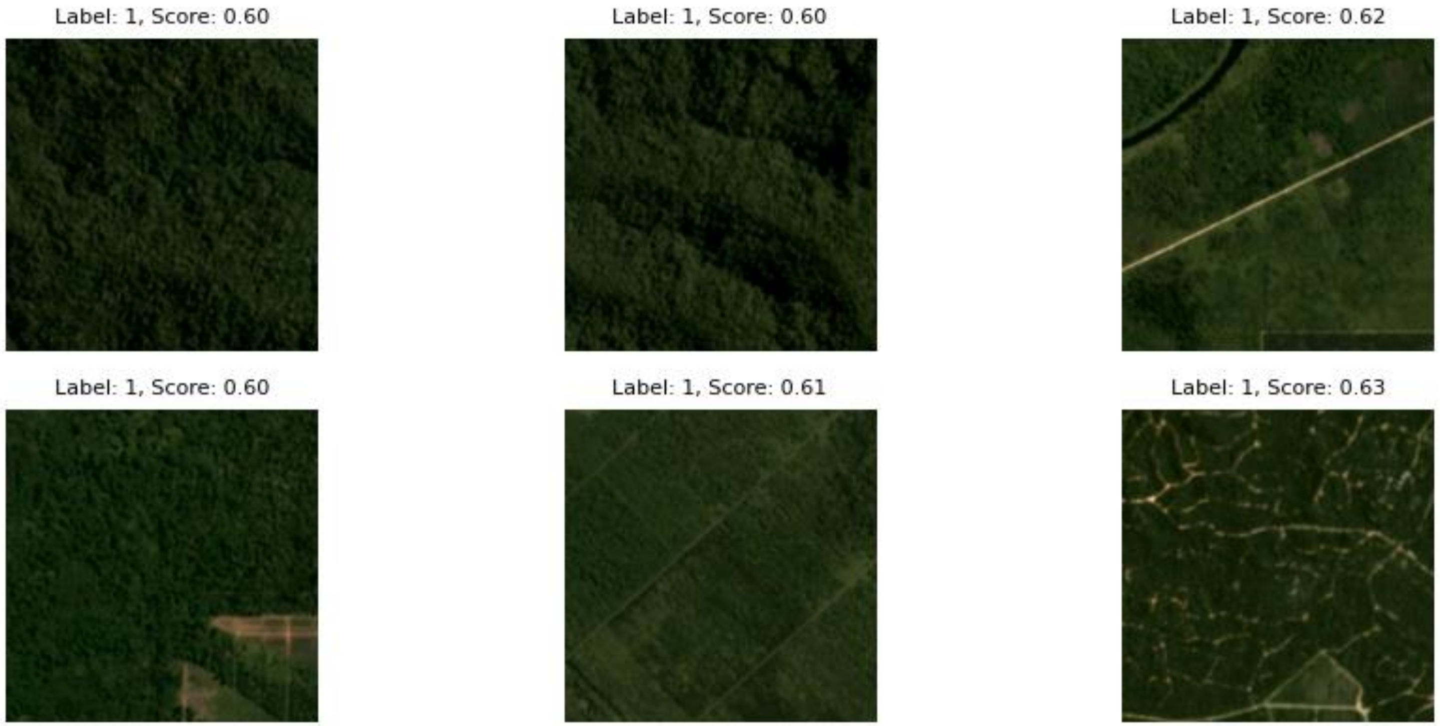
- Among the positive class images, 82 percent of them contain a `score` of 1, indicating high likelihood of a palm oil plantation
- As done previously, we will further investigate how `score` differs by looking at different thresholds:
  - `score` equals 1
  - `score` is between 0.7 inclusive and 1 non-inclusive
  - `score` is between 0.5 inclusive and 0.7 non-inclusive
  - `score` is below 0.5

`score` equals 1

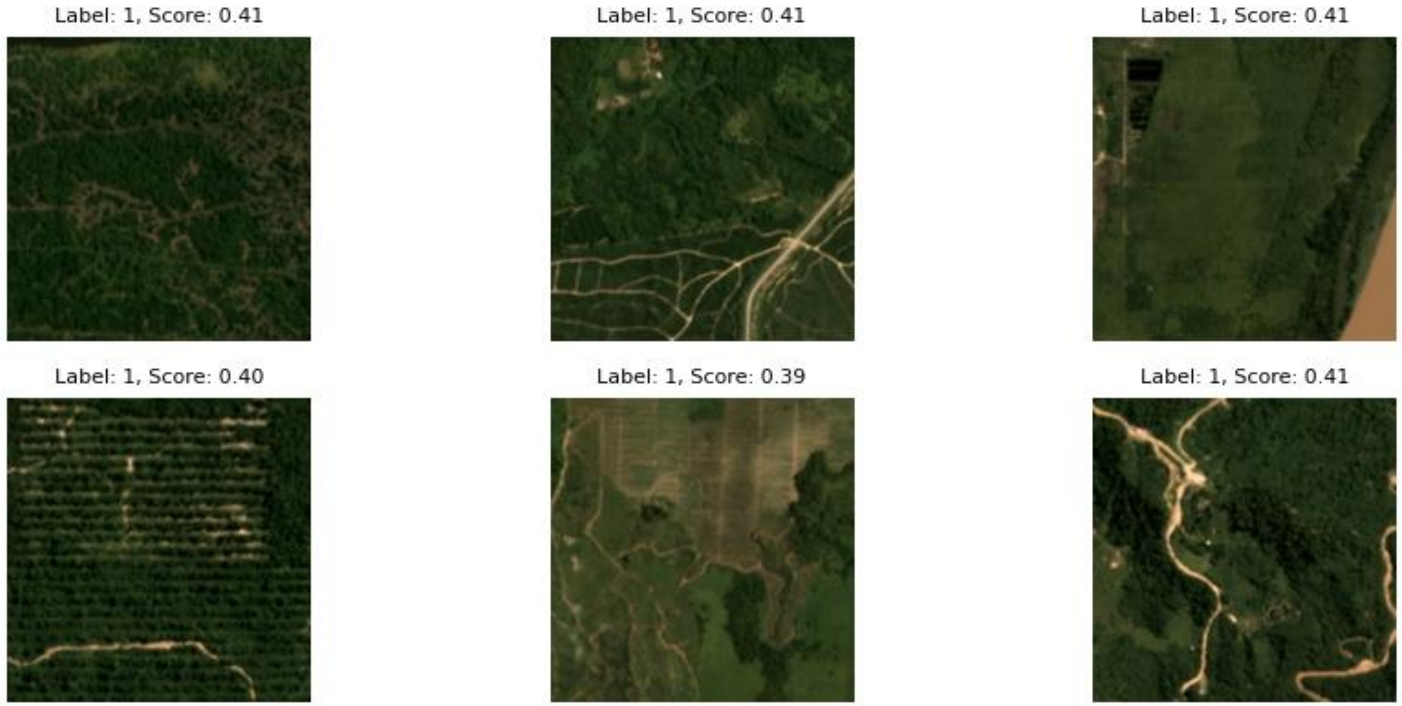


`score` is between 0.7 inclusive and 1 non-inclusive

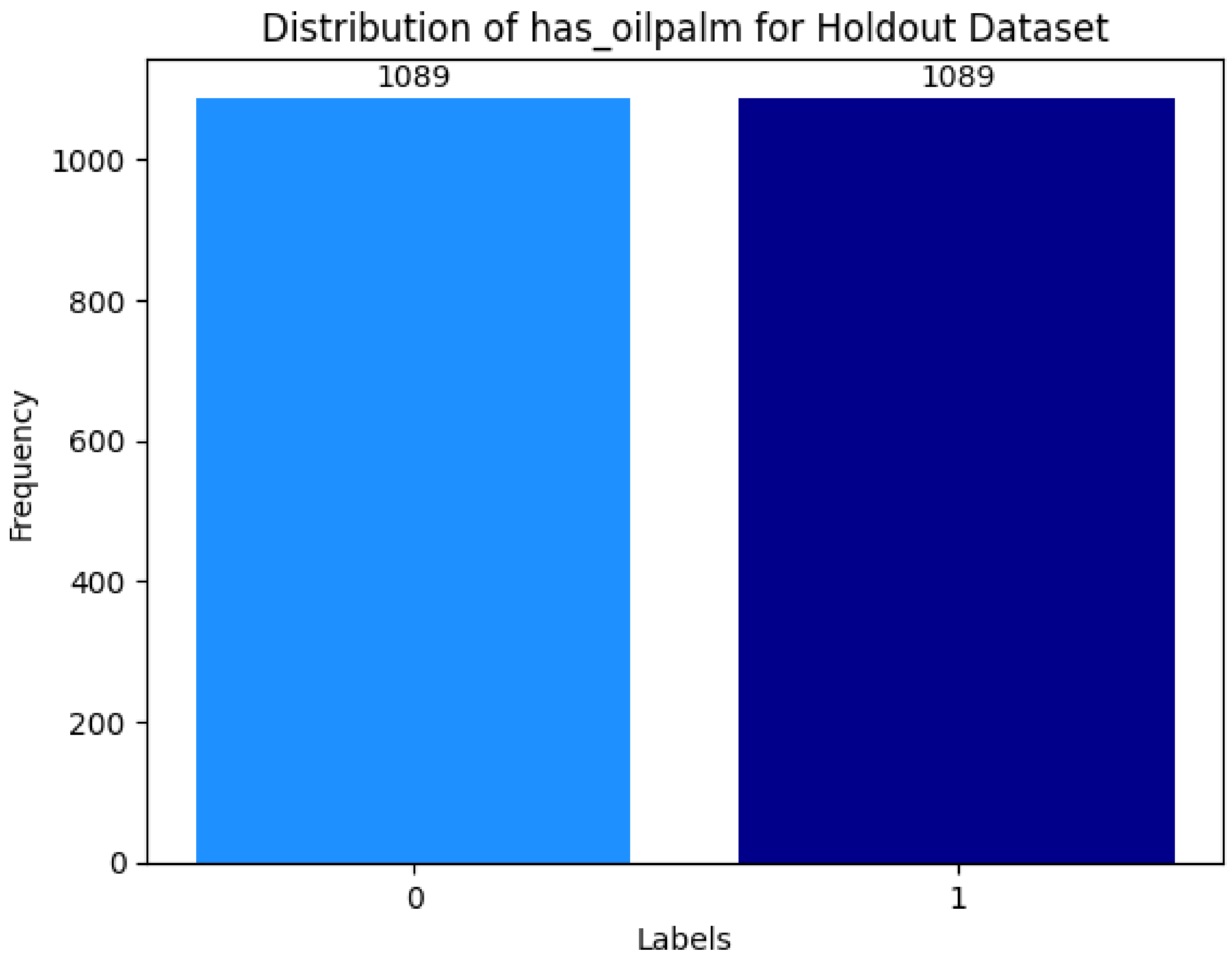
`score` is between 0.5 inclusive and 0.7 non-inclusive



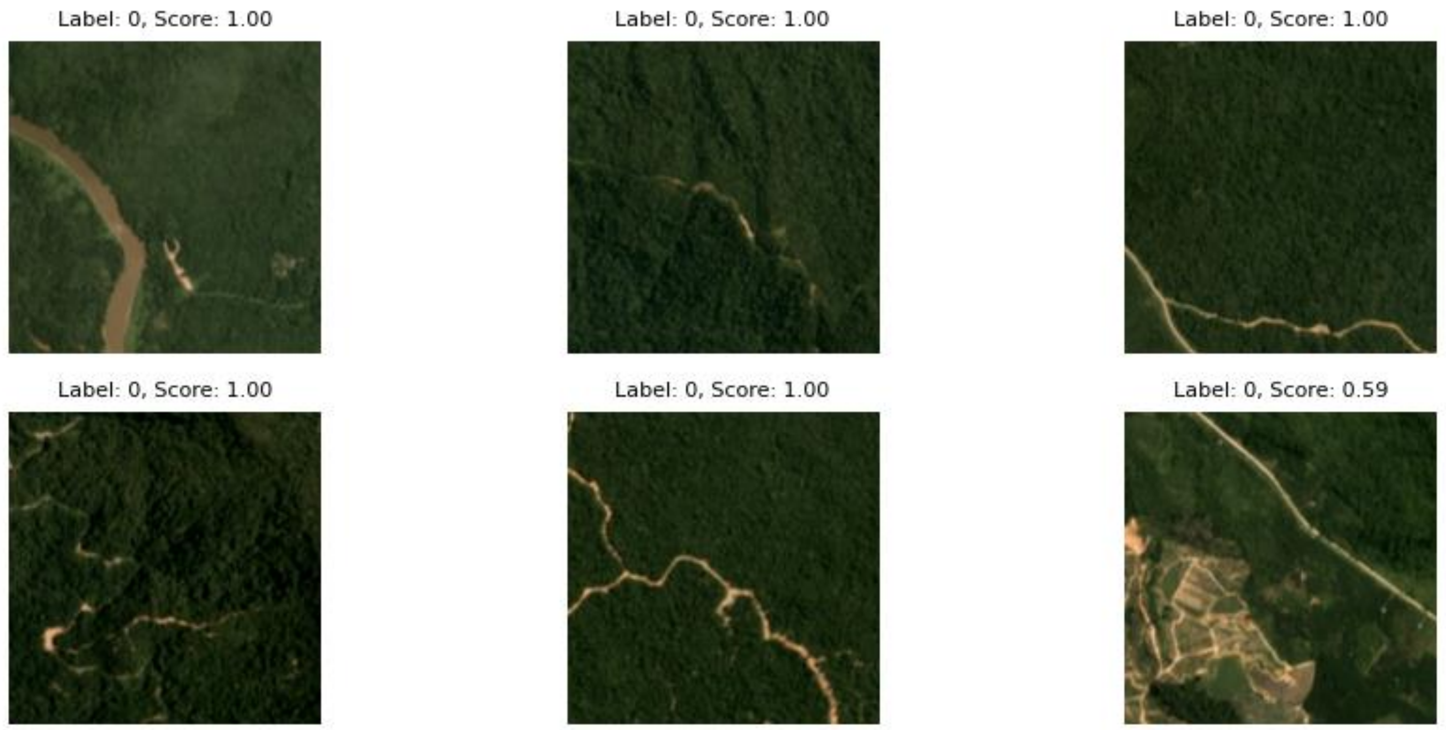
`score` is below 0.5



Distribution of Holdout Labels, has_oilpalm



Negative Class Holdout Images



Positive Class Holdout Images

Data Wrangling

- Since the dataset was highly imbalanced, both underampling of the negative class and oversampling of the positive class were utilized
- Establish two criteria for `score`:
  - For the negative class images, set `score` equal to 1
  - For the positive class images, set `score` greater than or equal to 0.5
  - These two criteria ensure robustness to our dataset

Data Wrangling

- After filtering the dataset with these two criteria, 7671 negative class images and 593 positive class images were obtained
- Finally, the negative class images were downsampled to 5930 and the positive class images were oversampled to 5930 via augmentation

Data Augmentation

Data Augmentation

Data Augmentation

- Retain as much of the original image as possible while adding robustness
  - Variations in color, lighting, and overall image quality
  - Resulting images are rather subtle, especially for the random brightness, random contrast, and random blur transformations
- Since the model uses VGG-19 (a pre-trained neural network), the images needed to be preprocessed in a specific manner, namely via the `preprocess_input` from `tensorflow.keras.applications.vgg19`
  - Each image was resized to (224, 224) with 3 color channels
  - Serialized and zipped using `gzip` and `pickle` libraries

Modeling

- Serialized and zipped images were unloaded for modeling
- Training data was train-test split via 80/20 stratified split, then further split into training and validation using another 80/20 stratified split
- VGG-19 was utilized (i.e., using transfer learning)
  - A pre-trained neural network that consists of 16 convolutional and 3 fully connected layers and trained from over 1 million images from the ImageNet collection
  - Initial layers were frozen to leverage its feature extraction capabilities, to help find certain patterns in the satellite mages
  - We also introduce our own custom layers since this model was not trained on satellite images

Modeling

Layer Type	Details
Pretrained base	VGG-19 (frozen)
Flatten	Converts extracted features (multi-dimensional) to one-dimensional
Dense 1	300 neurons, ReLu, Kaiming normal initialization
Dropout 1	40%
Dense 2	200 neurons, ReLu, Kaiming normal initialization
Dropout 2	30%
Dense 3	100 neurons, ReLu, Kaiming normal initialization
Dropout 3	25%
Dense 4	50 neurons, ReLu, Kaiming normal initialization
Dropout 4	25%
Dense 5	25 neurons, ReLu, Kaiming normal initialization
Dropout 5	20%
Output Layer	1 neuron, Sigmoid, L2 Regularization

Kaiming Normal Initialization

- Kaiming normal initialization was used on dense layers with ReLu activation functions
  - Addresses vanishing and exploding gradient problems → helps model learn and optimize better
  - In this type of weight initialization, Kaiming and his colleagues derived the relationship $W \rightarrow N(0, \frac{2}{n})$. W is a random number that follows a Gaussian distribution with mean 0 and variance of $\frac{2}{n}$, where n is the number of inputs to the node
  - The factor of 2 is specific to the ReLu activation function

L2 Regularization

- Aka Ridge regularization or weight decay
- Unlike L1 regularization which pushes the weights to zero, which in turn promotes sparsity, L2 regularization pushes the weights to be small but not zero
- Beneficial since:
  - Smaller weights reduce model complexity and overfitting
  - Weights are more evenly distributed (i.e., prevents cases where some weights are much larger than others)
  - Better numerical stability, especially when there is multicollinearity or when features are highly correlated with each other
  - Improves interpretability by reducing the influence of less important features
- Only applied to final dense layer to prevent over-regularization, which can hinder model's performance

AdamW Optimizer

- AdamW was used with `learning_rate = 1e-4` and `weight_decay = 5e-5`
- In Adam, the weight decay is applied indirectly when updating gradients, which can unintentionally modify the model's adaptive learning capabilities and interfere with the optimization process
- AdamW separates the weight decay from the gradient step, which ensures regularization impacts the parameters without altering the model's adaptive learning capabilities
  - In turn, this helps regularize models more precisely and helps models generalize better

Custom Callback

- A custom callback was used to keep record of several metrics evaluated on the holdout dataset, namely:
  - Precision scores of both classes
  - Recall scores of both classes
  - F1 scores of both classes
  - ROC-AUC score across epochs
  - Average precision (AP) score
- Optimize for the highest ROC-AUC score and AP score (since dataset was originally imbalanced)

Grid Search with Number of Epochs and L2 Regularization Values

- A grid search was done to determine the best-performing pair of hyperparameters
  - In this case, the number of epochs and an L2 regularization value
  - `epoch_values = [3, 4]`
  - `l2_values = [0.03851, 0.03852, 0.03853, 0.03854, 0.03855]`
- In earlier iterations, the model would begin overfitting after about 6 epochs, and L2 regularization values between 0.038 and 0.039 yielded the best results

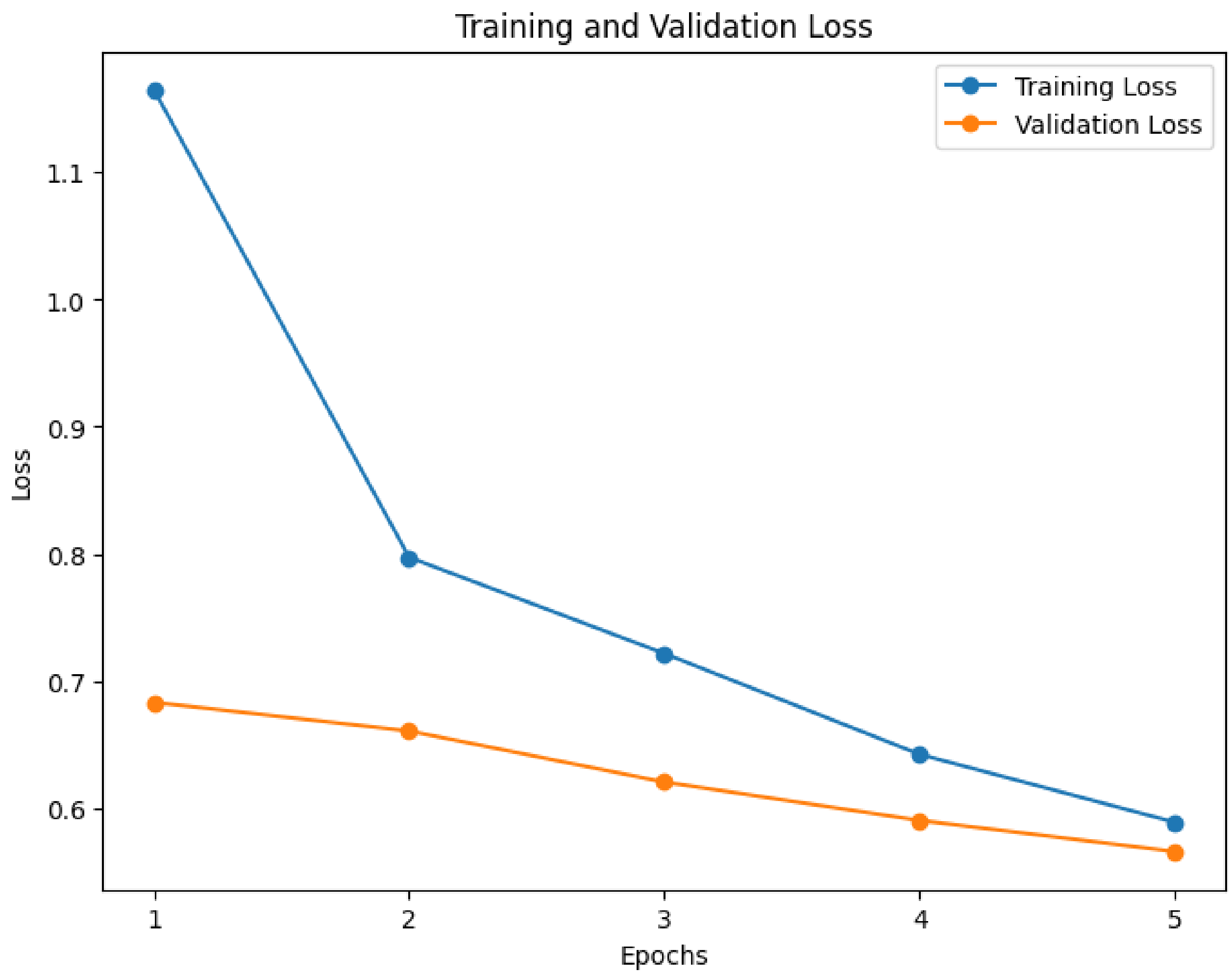
Grid Search with Number of Epochs and L2 Regularization Values

- While it was important to optimize for the highest ROC-AUC score and AP score, the condition used to determine the best-performing model was the average F1 score
  - `avg_f1_score = (f1_class_0 + f1_class_1) / 2`
  - We want to minimize both type I (predicting the presence of a palm oil plantation when there is none) and type II (predicting no presence of a palm oil plantation when there is) errors
- The best-yielding hyperparameters were `epoch = 3` and `l2_value = 0.03851`

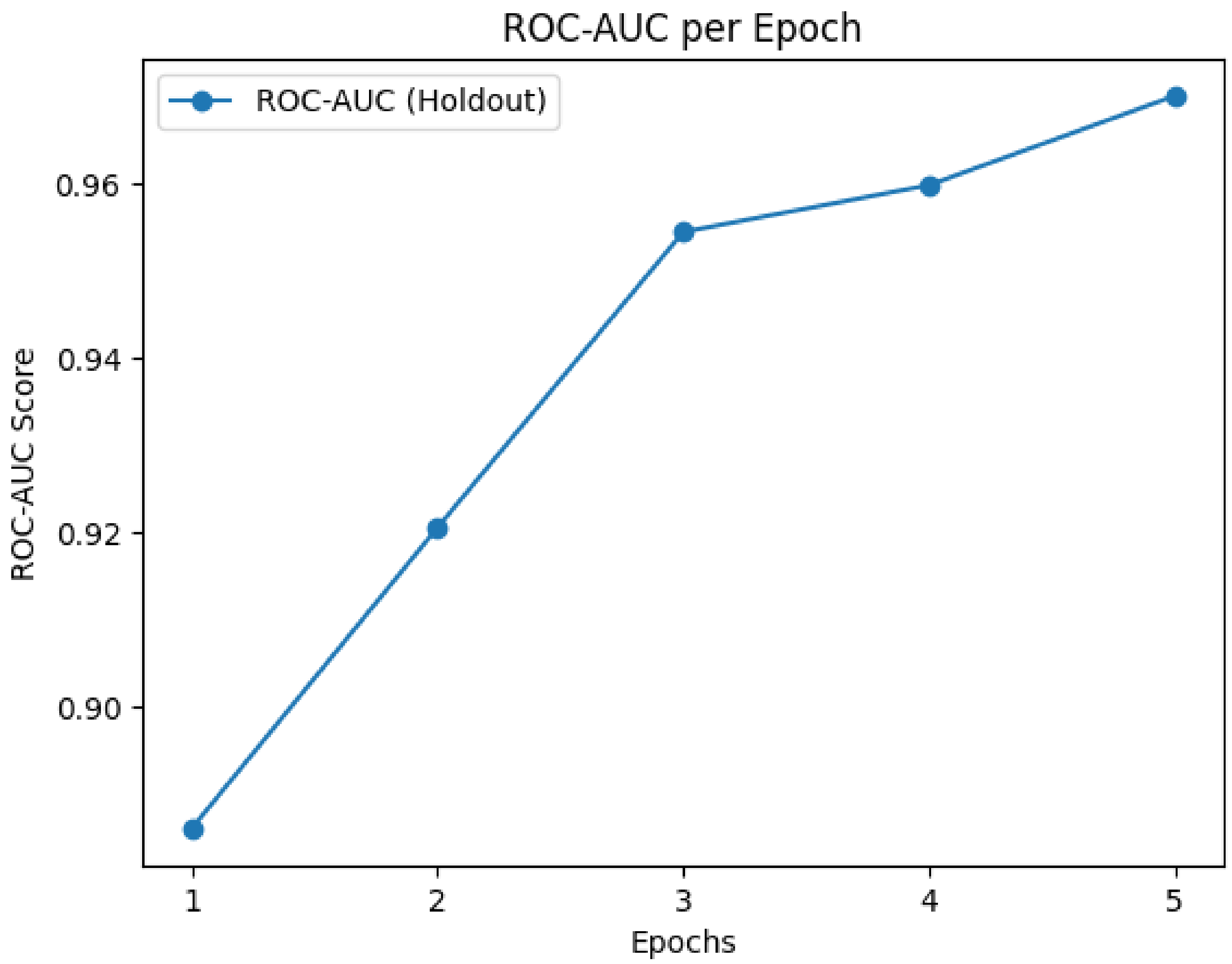
Grid Search with Number of Epochs and L2 Regularization Values

- The best model yielded from the grid search produced a holdout ROC-AUC score of 0.9545 and a holdout AP score of 0.9588
    - This model was then saved and subsequently trained (fine-tuned) for 2 additional epochs and the learning rate for the AdamW optimizer was lowered, where `learning_rate = 1e-5`
- The final model yielded a holdout ROC-AUC score of 0.9701 and a holdout AP score of 0.9721.

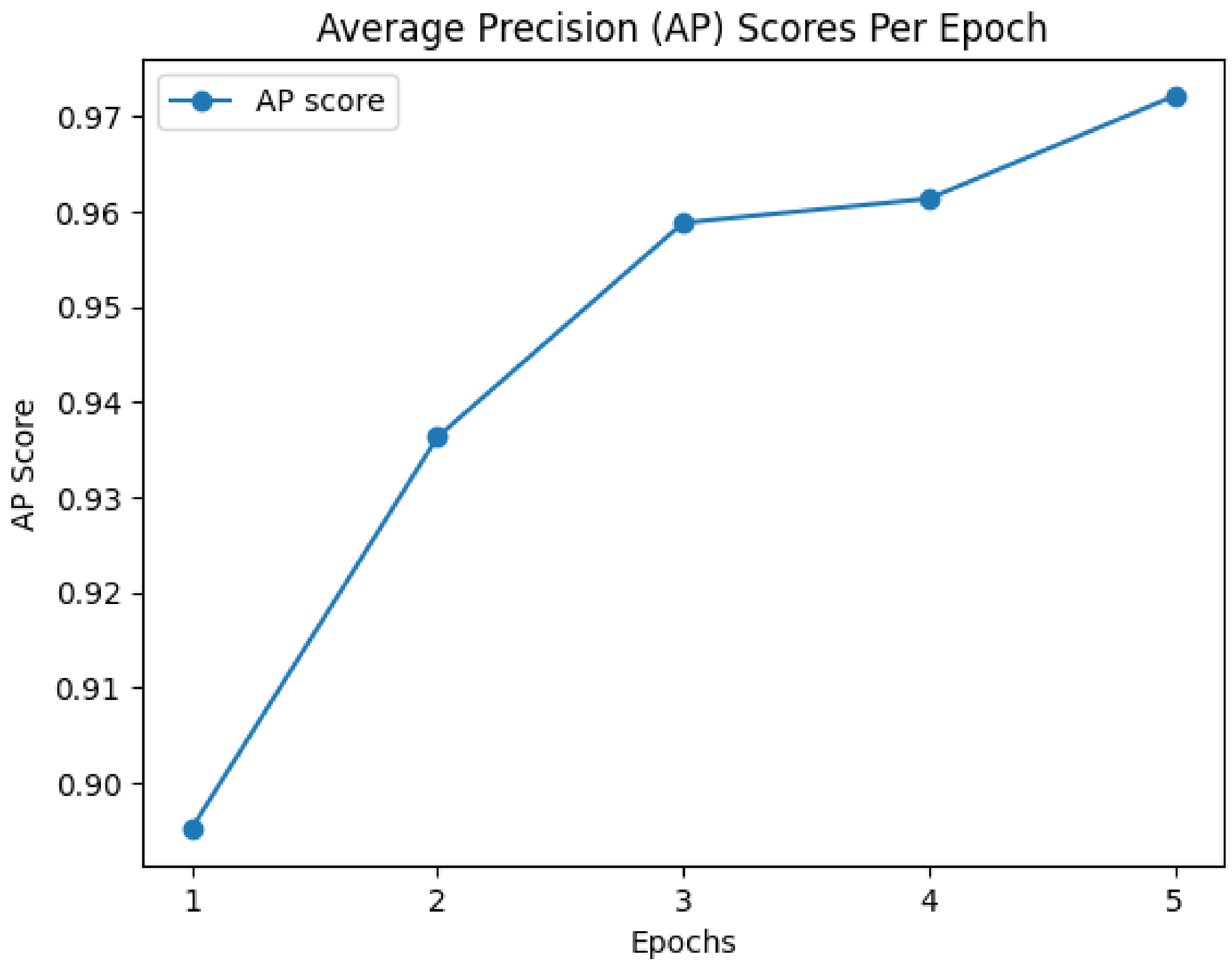
Training and Validation Losses



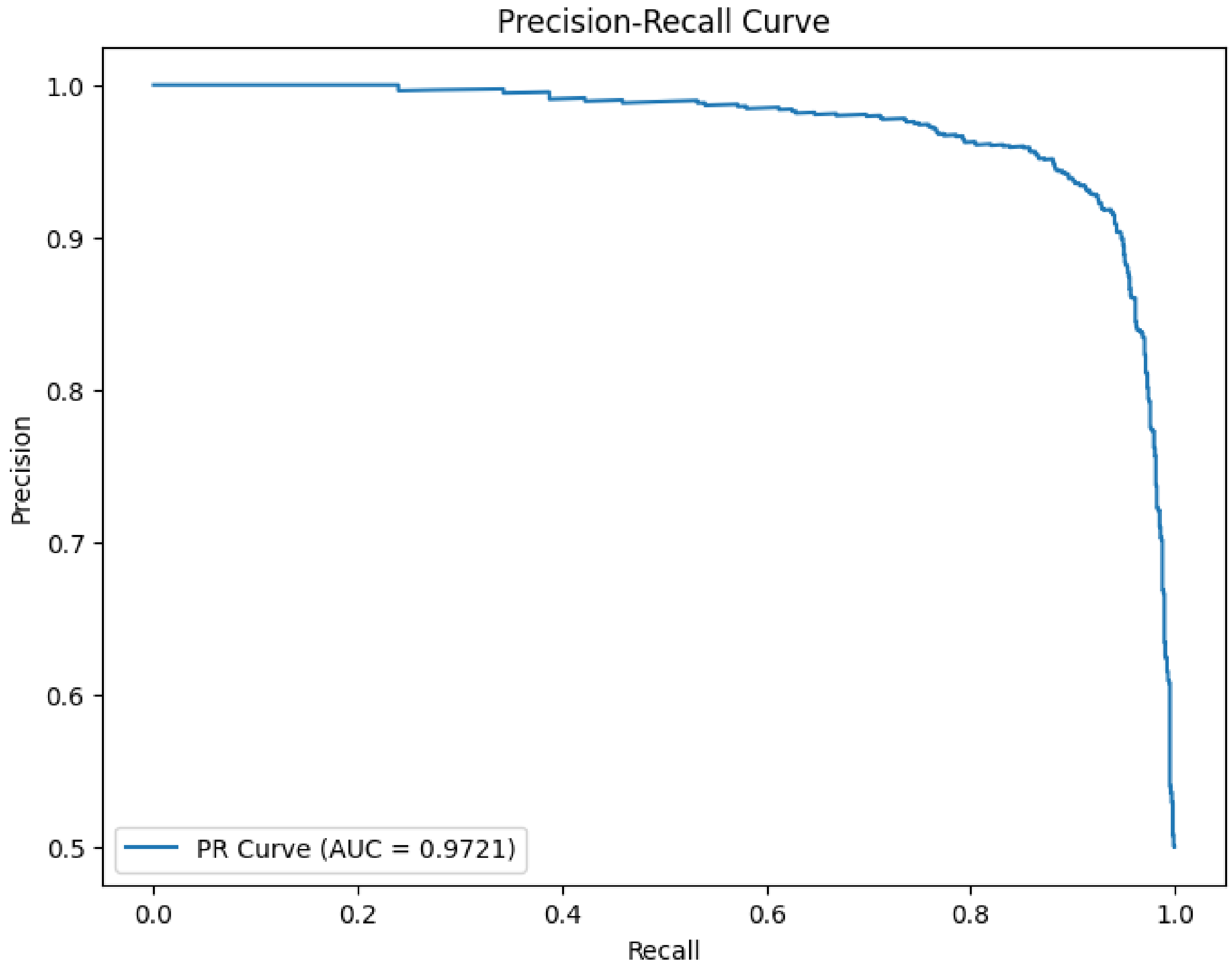
ROC-AUC Score per Epoch



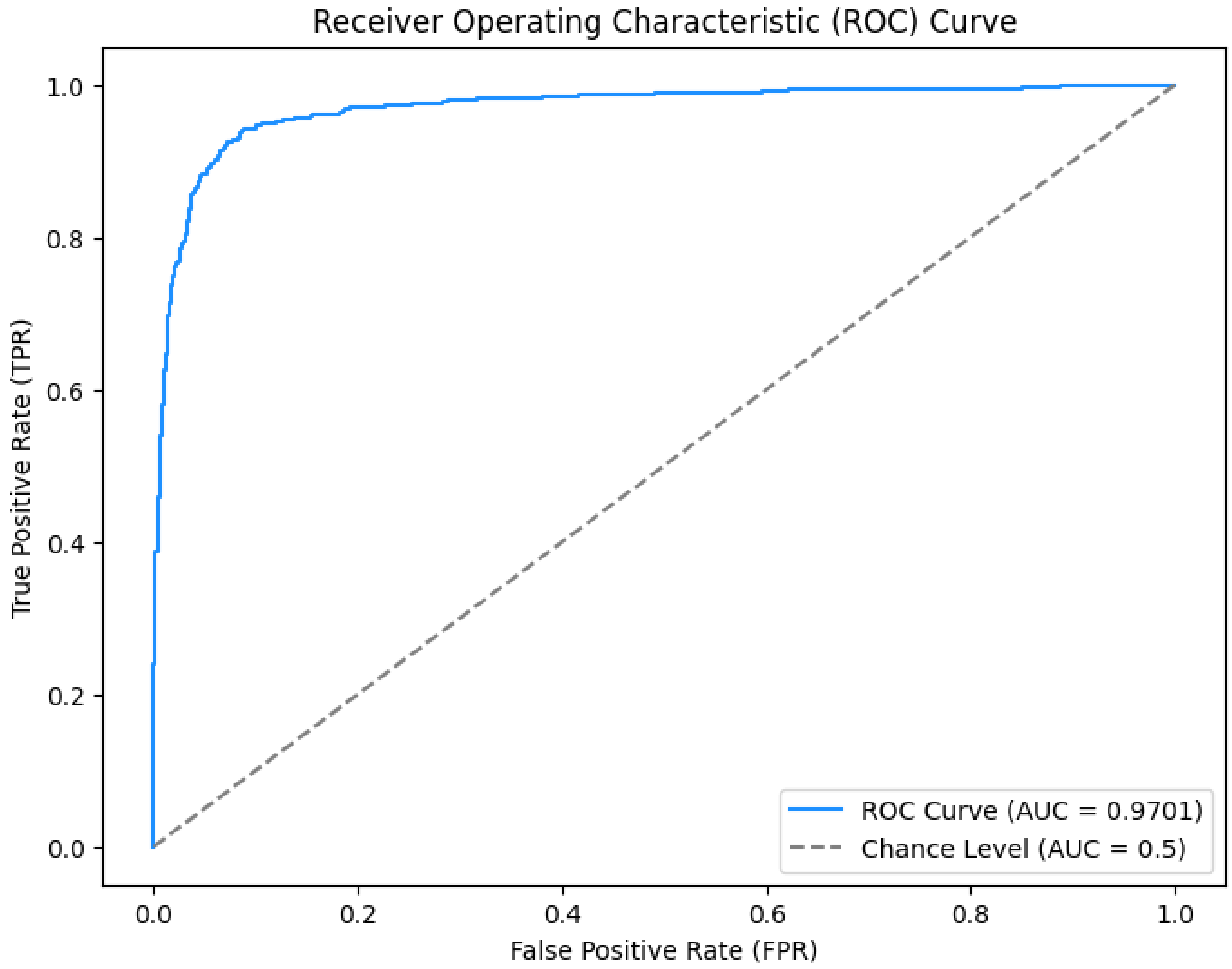
AP Score per Epoch



Precision-Recall Curve



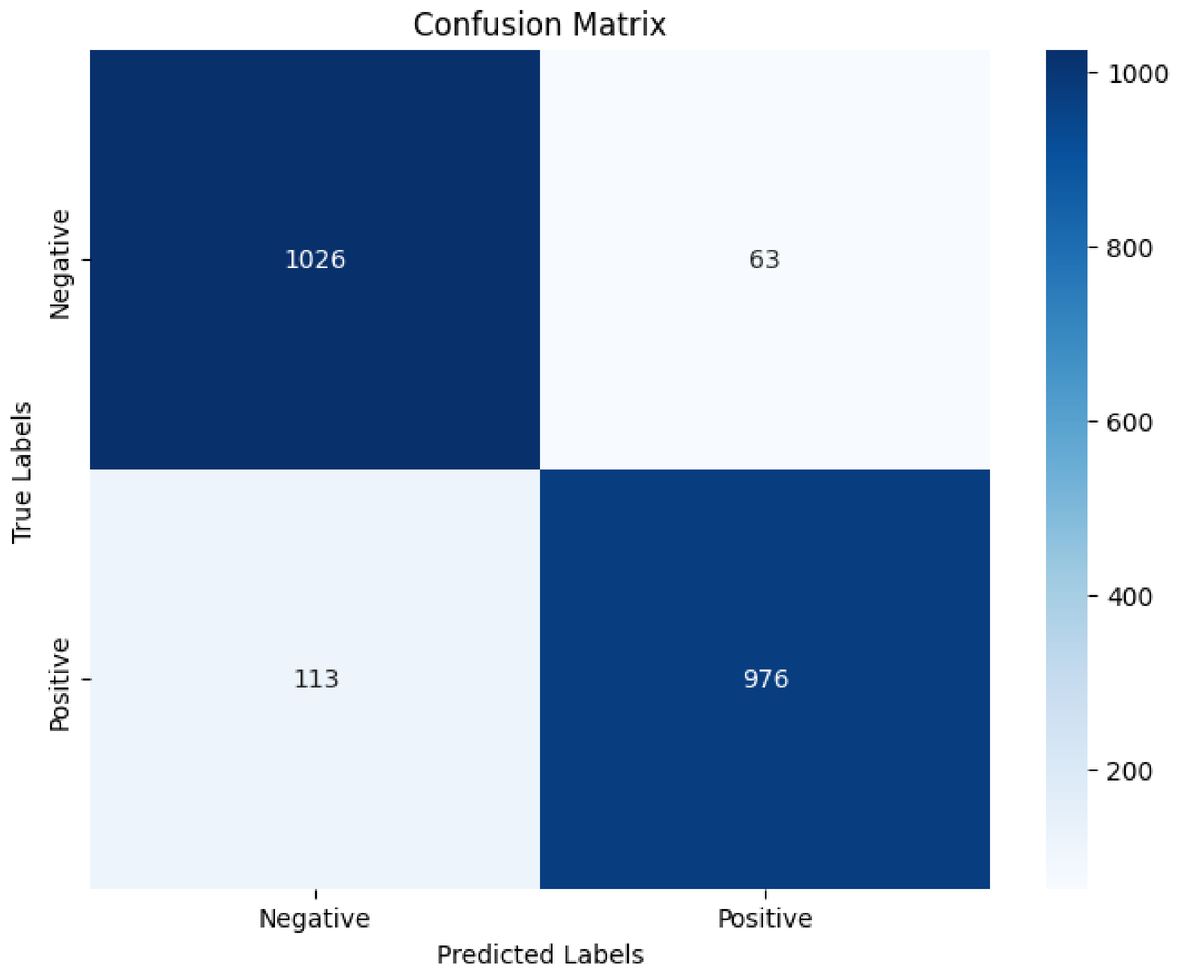
ROC Curve


Receiver Operating Characteristic (ROC) Curve
True Positive Rate (TPR)
False Positive Rate (FPR)
0.0
0.2
0.4
0.6
0.8
1.0
ROC Curve (AUC = 0.9701)
Chance Level (AUC = 0.5)

Classification Report

	precision	recall	f1-score	support
0	0.90	0.94	0.92	1089
1	0.94	0.90	0.92	1089
accuracy			0.92	2178
macro avg	0.92	0.92	0.92	2178
weighted avg	0.92	0.92	0.92	2178

Confusion Matrix



Results

- Both training and validation losses decreased together and start to converge to similar levels after the fifth epoch
- Both ROC-AUC and AP scores increased after each epoch → model progressively improved
  - Both metrics were considerably high since both were close to 1
- For the confusion matrix, the model predicted 94% of the negative class and 90% of the positive class correctly, also achieving an F1 score of 0.92 for both classes

Conclusion

- Model yielded remarkable results due to data augmentation, hyperparameter tuning, and fine-tuning
  - Effectively addressed class imbalance and prevent overfitting
  - Adapt VGG-19 to our unique problem
- Future considerations include:
  - Try another pretrained model like ResNet-50
  - Readjust augmentation for the positive-class images
  - Setting different threshold for `score` for the positive class images at greater than or equal to 0.6
  - Random grid search to save on time and computational costs

References

- WWF. “8 Things to Know about Palm Oil.” WWF, WWF, 12 Nov. 2018, www.wwf.org.uk/updates/8-things-know-about-palm-oil.
- “Palm Oil | USDA Foreign Agricultural Service.” Usda.gov, 2024, www.fas.usda.gov/data/production/commodity/4243000.
- Pykes, Kurtis. “AdamW Optimizer in PyTorch Tutorial.” Datacamp.com, DataCamp, 21 Oct. 2024, www.datacamp.com/tutorial/adamw-optimizer-in-pytorch.
- MathWorks. “Mel Frequency Cepstral Coefficients.” Mathworks.com, 2020, www.mathworks.com/help/audio/ref/cepstralcoefficients.html, https://doi.org/1061245463.woff2.
- “Global Deforestation Slowing but Tropical Rainforests Remain under Threat, Key FAO Report Shows.” Newsroom, 5 Mar. 2022, www.fao.org/newsroom/detail/global-deforestation-slowing-but-rainforests-under-threat-fao-report-shows-030522/en.

References

- Vaj, Tiya. "Why We Freeze Some Layers for Transfer Learning - Tiya Vaj - Medium." Medium, 24 Jan. 2024, vtiya.medium.com/why-we-freeze-some-layers-for-transfer-learning-f35d9f67f99c.
- "VGG-Net Architecture Explained." GeeksforGeeks, 7 June 2024, www.geeksforgeeks.org/vgg-net-architecture-explained/.
- "Kaiming Initialization in Deep Learning." GeeksforGeeks, 27 Dec. 2023, www.geeksforgeeks.org/kaiming-initialization-in-deep-learning/.
- GeeksforGeeks. "L1/L2 Regularization in PyTorch." GeeksforGeeks, 31 July 2024, www.geeksforgeeks.org/l1l2-regularization-in-pytorch/. Accessed 7 Feb. 2025.
- Brownlee, Jason. "A Gentle Introduction to Dropout for Regularizing Deep Neural Networks." Machine Learning Mastery, 3 Dec. 2018, machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/.