Comparative Analysis Of CNN Architectures Using Amazon Deforestation Dataset

Gonzalo Barrientos (16104515)   Nick Thomson (16121669)   Siddharth Chatterjee (17043669)
Zobair Hasan (12012441)

Abstract

Through this assignment we aim to explore and compare the performance of different Convolutional Neural Network (CNN) architectures. We draw our motivation from the multi-class labelling problem posed by the Amazon rainforest dataset on Kaggle. The architectures considered are: 4 and 5 layer CNN models, Deep CNN and VGG models, and ensemble models. We also discuss in detail the issues faced when scaling from binary classification tasks to multi-label classification tasks.

1. Introduction

Since 1970, nearly 20 percent of the Amazon rainforest has been lost to deforestation due to human activity (Rhett, 2017). This has led to significant environmental implications in the form of reduced biodiversity, habitat loss, and climate change. During the bulk of this 40-year period, the Brazilian government have depended on detection algorithms using coarse-resolution images to track the terrain changes in the Amazon forest. (Popkin, 2016) However, the lack of detail in these images and the drawbacks of existing algorithms mean it can take weeks or months to detect changes in land conditions.(Allen Zhao & Yu, 2017) Deforestation therefore cannot be stopped in its early stages. Therefore novel datasets and approaches which can identify changes in land conditions more efficiently are highly valuable. Data gathered on human activities in the Amazon from these images revealed that the primary reason for deforestation is illegal agriculture, logging, and mining. (Rhett, 2017)

In 2017, Kaggle together with a geo-spatial solutions company (SCCON) challenged Kagglers to label satellite images of the Amazon rainforest with atmospheric conditions and various classes of land cover/land use. To aid with this challenge they released a dataset of labelled satellite images of the Amazon rainforest collected by Planet, a satellite imaging company.(Kaggle, 2017). The dataset is novel in that it takes advantage of very high resolution satellite imagery. It was hoped that this new dataset could be used to draw further insight on how satellite images can capture deforestation driven by human activities.

From a machine learning perspective, the challenge is a multi-label image classification problem. The current state-of-the-art approach for tackling such problems is to use Convolutional Neural Networks (CNNs). CNNs from their inception have been designed to tackle vision related problems unlike general purpose classifiers such as multi-layer perceptrons (MLPs). CNNs are designed to specifically expect images as inputs. Based on this assumption, convolutional layers work by convolving filters across images to detect features. CNNs are therefore able to capture location invariant features unlike MLPs which are only able to detect features specific to particular regions in an image. (CS231) Features identified in CNNs are therefore more robust and transferable. This opens up the possibility of transfer learning, where we can use weights trained for a particular CNN, for similar image classification problems. Due to the aforementioned factors, we focus our exploration on CNNs.

Our project aims to explore the performance of different CNN architectures based on AlexNet. We train a simple baseline CNN architecture and examine how performance changes as we make incremental changes to the architecture. In particular, we investigate the effect of varying the number of convolutional layers and max pooling layers with a view to confirming that deeper architectures perform better. We also investigate the effectiveness of transfer learning and perform fine tuning using VGG-16. This is with a view to assessing how well a network trained on images of objects can be transferred to satellite images. Finally, we take a novel approach inspired by ensemble learning and an example from the literature(Liang & Tang, 2017). This approach is taken to explore whether specialised networks perform better than a single network. Our approach involves training three networks in parallel. The outputs of each network are then concatenated to produce a final classification vector. We evaluate performance by looking at the F2 score achieved and also by analysing confusion matrices.

2. Related Work

DeepSat A Learning framework for Satellite Imagery (Saikat Basu & Nemani, 2015) looks into the dataset SAT-4 and SAT-6. These datasets are pertinent today and lay the groundwork for satellite image classification. The two datasets, together, contain 900,000 images, which are one-hot encoded with the labels: barren land, trees, grassland, and other for SAT-4 and barren land, trees, grassland, roads, buildings and water for SAT-6. This paper explores several techniques for image classification such as: autoencoders, deep belief networks and basic CNNs.

While serving as a good motivation for further study in the field of satellite imagery, this paper provides little help in the Deep Learning portion of this project. This is mainly due to the difference in classification (one-hot as opposed to multi-label binary) and the difference in the distribution of the datasets. SAT-4 contains 4 labels which are evenly distributed, while the Amazon dataset is heavily skewed with 17 total labels, meaning methods may not generalise well.

(Adke & Johnson, 2017) explored the role of transfer learning in pretrained models used in other fields when applied to satellite imagery. This paper derives heavily from the work of (Otavio A. B. Penatti & dos Santos, 2015) who generalised deep networks used in classification on ImageNet to the new domain of aerial scenes/satellite imagery. It achieved near state-of-the-art results using a combination of Overfeat (Pierre Sermanet & LeCun, 2014) and CaffeNet(a modified version of AlexNet (Alex Krizhevsky & Hinton, 2010) pretrained on the ImageNet (**?**) dataset) to help classify images in the Brazilian Coffee Dataset (Coffee, 2014) and the UCMerced (UCMerced, 2010) Dataset. We take follow (Adke & Johnson, 2017) and also explore transfer learning in our project.

(Adke & Johnson, 2017) also discuss ensemble classification methods. These have been known to achieve great results on the Kaggle leaderboards, however they are very computationally demanding and time intensive. Furthermore, they do not provide any additional benefit than heavily optimising the accuracy for competitions. We briefly investigated them for this project, however we decided against having ensembling as a focus due its increased computational demands.

(Allen Zhao & Yu, 2017) built on the ensemble methods used in (Adke & Johnson, 2017) to complete the Kaggle competition. Their paper down scaled the images to 64x64 and augmented them in order to achieve a higher score on the leaderboards. This approach is of interest, as our project required us to downscale the images in order to fit the memory constraints we faced. However, we recognise there is an important trade-off between down scaling and the ability to use data augmentation when trying to optimise for accuracy. The decision to down scale in order to allow for data augmentation may not be optimal given the loss of information from down scaling.

(Allen Zhao & Yu, 2017) enumerated certain unsuccessful approaches to our image classification task. This shed important light on the methods that may fail to improve our F2 accuracy score. One particular approach was the restriction of the weather labels to one particular label per image. This caused a lower F2-score as F2 penalises False Positives more than it rewards False Negatives. In this project we have aimed to bridge the gap between the conventional approach and this approach by categorising the labels and assigning a label from each category to each image.

(Liang & Tang, 2017) uses the same (Kaggle, 2017) dataset and builds its baseline model with a simple CNN. This approach has been replicated in our project in order to create a baseline model for classification, and comparison with more complex models. The baseline CNN is excellent benchmark; both in terms of low compute time and high accuracy. (Liang & Tang, 2017) then used a modified pretrained VGG-16 (Simonyan & Zisserman, 2015) model on the RGB layers of the dataset to classify the images. It also mentions that infrared channels (the fourth channel in all images) was not used as this was not believed to increase the F2-score and increased the GPU compute time of the model. We have also chosen not to use for infrared channels for the same reasons.

The merits of data augmentation and class balancing are discussed in (Liang & Tang, 2017). This paper found that the use of both methods were able to stop overfitting and increase the validation accuracy. Class balancing reduces overfitting as it stop the model from learning the features of the major classes, which artificially inflates the F2-score.

(Adke & Johnson, 2017), (Allen Zhao & Yu, 2017) and (Liang & Tang, 2017) take slightly different approaches to the Kaggle competition. All of them build a baseline CNN before studying a deeper network like VGG or ResNet (Kaiming He & Research, 2010). Throughout these papers, the underlying theme seems to imply that baseline CNNs, even when shallow (3-4 layers), are able to perform just below the state-of-the-art implementation. The deeper models introduced in these papers only produced marginal gains on the baseline. (Adke & Johnson, 2017) deemed a basic 8 layer CNN to be the best performing model.

3. Dataset

The full dataset consists of 40,478 satellite images. The images are 256x256 pixels with 4 channels (RGB and infrared) and each image covers 90 hectares of land. There are 17 labels that each image can be classified by. These fall into 3 broad categories: Weather conditions, common cover/activity and rare cover/land use. Weather conditions include: clear, partly cloudy, cloudy, and haze. The common cover/land use labels are: primary rainforest, Water (rivers and lakes), Habitation, agriculture, road, cultivation and bare ground. The rare cover/land use labels include slash and burn agriculture, selective logging, blooming, conventional mining, and artisanal mining.

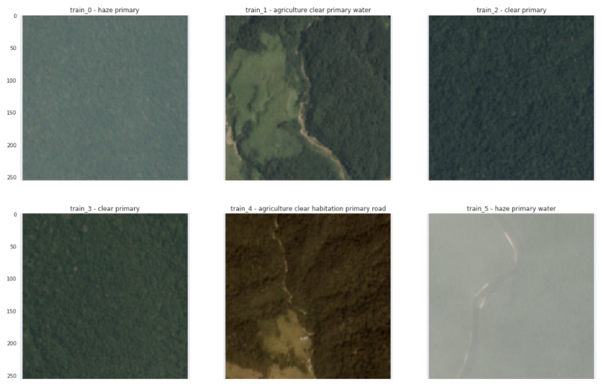

train_0 - haze primary
train_1 - agriculture clear primary water
train_2 - clear primary
train_3 - clear primary
train_4 - agriculture clear habitation primary road
train_5 - haze primary water
0
50
100
150
200
250

Figure 1. Examples of images from the dataset and their associated labels. Top left to right: Haze-primary, agriculture-clear-primary-water, clear-primary. Bottom left to right: clear-primary, agriculture-clear-habitation-primary-road, haze-primary-water

An image can be associated with multiple labels, but certain combinations of labels are not possible. For example, an image should not have both the clear and cloudy label. Unfortunately, the dataset is not free from labelling errors. The frequency of the 17 labels is also quite varied; with the primary label denoting forest forming 90% of images, whereas mining appears in less than 1% of images.

4. Methodology

4.1. Preprocessing

We split the provided dataset into training, validation, and test set. The training set consists of 80% of the dataset (32,382 images), while the validation and test sets have

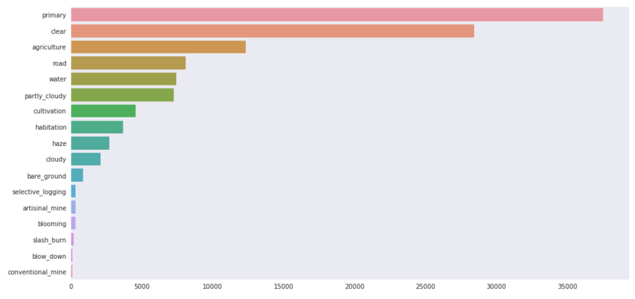


Figure 2. Distribution of class labels

10% of the dataset (4,048 images) each. We took care to ensure the three datasets (training, validation, test) were balanced. Each of the three datasets used have the same distribution of labels.

Of the 4 colour channels available in the data set, we only make use of the RGB channels and do not use the infrared channel as its inclusion only results in very marginal gains (Liang & Tang, 2017), while requiring significantly more memory.

Due to our computational resource constraints, we made the decision to the downscale the images from 256x256 pixels to 128x128 pixels using interpolation.

We also considered using data augmentation, like (Allen Zhao & Yu, 2017) who incorporate vertical/horizontal flipping, shifting, horizontal shearing and random rotation. However, we decided against doing this due to the increased training time and memory requirements.

4.2. Network Architectures and Training

4.2.1. Training

We developed and trained our models in python using the Keras deep learning library. All of our models are trained for 25 epochs using a batch size of 256. We use Adam as our optimiser and binary cross-entropy for our loss. The choice of optimiser is not one emphasised in the literature tackling our problem area. Hence, we used Adam (Kingma & Ba, 2015) as it has more general-purpose applications. With regards to our choice of loss function, we chose binary cross-entropy loss as our outputs are not mutually exclusive. Since our image can have multiple labels, the output vector will not be one-hot encoded. Therefore, a categorical cross-entropy loss is not suitable. Our single output vector is instead set to 1 for the labels where the predicted probability is above a configurable threshold. As we have a single output, we use the binary cross-entropy loss.

The probability threshold for which to assign a 1 to a label in the output vector was not a focus during our investigations. We tried a small number of thresholds, looking at 0.1 increments starting from 0.1 up to 0.5. We did not look beyond this as we noticed model performance began to deteriorate as we got more conservative with our threshold. The final threshold we settled upon was 0.2. Therefore, any label assigned a probability of above 0.2 was set to 1 in our final output vector.

We use the callback functionality on Keras along with an early stopping mechanism with a patience of 3. This allows us to check for overfitting. We also implement the reduction of learning rate on plateaus, which helps our model find a more precise minimum point. Finally, we implement model checkpoints where we save the best performing weights in case model starts to overfit in later epochs.

4.2.2. Simple Convolutional Neural Network

Our input layers take an 128x128x3 input. As mentioned previously, this is because we are using 128x128 pixels images and considering the RGB channels. The inputs are batch normalised for all of our models.

Model 0
As our starting point, we train a very simple CNN with 3 convolutional layers each followed by a max pooling layer. The first convolutional layer is given 16 filters with the number of filters doubling in each subsequent layer. Each max pooling layer has a pool size of 2. Finally, before the output layer we have a global average pooling layer. With regards to activation functions and weight initialisation, we use RELUs for all layers bar the output layer. The output layer uses a sigmoid activation function. We choose the sigmoid as opposed to softmax as using a softmax assumes the probabilities of each of the classes are dependent, that means that the classes should mutually exclusive if we consider softmax as the activation function of the last fully connected layer. This is not an assumption we wish to make. The design of the aforementioned architecture is inspired by VGG and AlexNet which follow a similar structure but with many more layers. We incrementally improve upon this model by training CNNs with additional convolutional layers and max pooling layers.

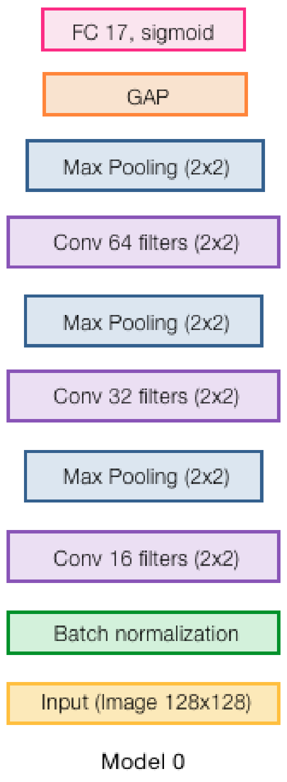


Figure 3. Baseline CNN Model

Model 1 and 2
Models 1 and 2 add additional convolutional and pooling layers. Model 1 adds 1 additional convolutional layer and 1 additional pooling layer immediately after the final pooling layer of model 0. Model 1 therefore has 4 pairs of convolutional and pooling layers. Model 2 repeats the additions of a convolutional and pooling layer so that we have 5 pairs of convolutional and pooling layers. Our motivation behind increasing the number of convolutional layers is that we expect it allow the model to capture more complex features and therefore better distinguish different labels. The resolution of our images is fairly high, so we expect a shallow network will not adequately capture all possible features.

Model 3
The final CNN that we train has an additional convolutional layer after 4 of the 5 convolutional layers trained in the penultimate baseline model. The additional convolutional layer is given the same number of filters as the layer immediately preceding it. We therefore have 2 convolutional layers followed by a max pooling layer repeated 4 times. The final convolutional layer does not have an additional layer following it. This is once more taking inspiration from VGG and drawing our model architecture closer to the VGG architecture.

4.2.3. Transfer learning: Fine Tuning VGG-16

The performance of deep learning models can be highly dependent on the size of the dataset they are trained on. Not all problem domains have access to large datasets for training purposes. Transfer learning is the practice of using pretrained models that have been trained on existing very large datasets. It is thought that as CNNs identify features that are location invariant, the trained features can translate over to other similar types of problems. The VGG model is trained on a large training set of images and is a model that could be applicable to other image classification tasks such as ours.

Transfer learning by fine tuning involves using another model's architecture and pretrained weights as the starting point. The input and output layers of the pretrained model is removed and replaced by one suited to the new classification problem that the model is being applied to. Furthermore, with fine tuning, we do not simply use the pretrained weights as received but instead conduct further training after initialising from the pretrained weights. This can be done for a subset of the layers in the original model or for all layers. We retrain the weights for all layers.

We are not the first to try fine tuning VGG-16 for the Amazon rainforest dataset. (Adke & Johnson, 2017) used VGG-16 due to its prior success for land classification problems and because of the similar dimensions of the original VGG-16 input size in comparison to our satellite image data set.

use labels. They postulate that this will allow each network to be more focused and therefore better identify features specific to each respective grouping of labels.

We propose the use of three networks. One for weather conditions, one for common cover/land use and one for rare cover/land use. Unlike (Liang & Tang, 2017), we do not have just the one network for cover/land use. Our motivation is that common and rare cover/land use may have different features, so having separate networks may allow them to better identify features specific to their respective labels.

We base the architecture of the three networks on one of our previous CNNs. We choose model 2, the version with 4 max pooling layers and 8 convolutional layers (in pairs preceding each max pooling layer). This is selected as it has the best trade off between performance and training times. Like the simple CNN, each network outputs a vector of 1s and 0s where labels with a value above a specified threshold are assigned - 1, and those below - 0. The vectors from each network are then concatenated and a final vector is produced which matches that of our normal CNN.

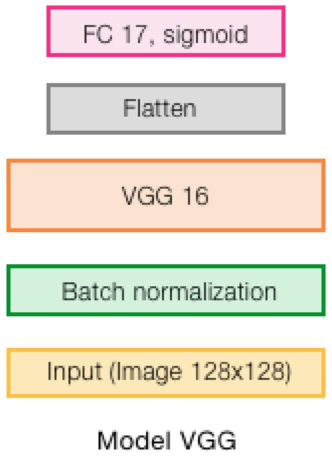


Figure 4. VGG-16 Architecture

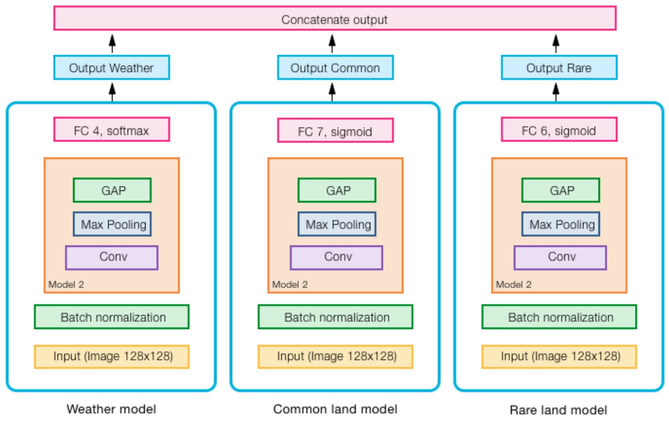


Figure 5. Ensemble Model Architecture

4.2.4. 3 Network model

(Liang & Tang, 2017) take advantage of the fact that the labels related to weather conditions are independent of the other labels. Instead of training a single network, they train two networks. One network classifies the weather condition label for an input image and the other network classifies the common cover/land use and the rare cover/land

According to the architecture presented on Figure 5, we notice that each fully connected layer has the number of units related to its number of classes. For instance, the common land model has 7 units in the last layer. It is important to mention that we are considering a softmax layer for the weather model since only for this case the classes are mutually exclusive. That means that the weather model was trained using a categorical cross-entropy loss.

4.2.5. Evaluation

Our primary metric for evaluating the performance of our model is the F2 score. This is used in majority of the papers looking at the Amazon rainforest dataset as well as the Kaggle competition. The F2 score is a metric which combines the precision and recall measures.

Precision is defined as follows:

$$\frac{True\ Positives}{True\ Positives + False\ Positives}$$

Intuitively, this measures how accurate a model is when it classifies a particular image with a certain label. High precision may mean when a model classifies an image with a label, it is highly likely to be correct. This however does not mean the model is necessarily well performing as there may be a lot of false negatives. Recall is a metric which takes into account false negatives.

Recall is defined as follows:

$$\frac{True\ Positives}{True\ Positives + False\ Negatives}$$

The generic F-score has the following formula and is a ratio between precision and recall:

$$F_2 = \frac{(1+\beta^2)\ precision}{\beta^2\ precision + recall}$$

The commonly used F1 score follows the same formula with β = 2. We follow the literature and use a β value of 2 (hence F2 score), thus giving preference to recall over precision. The primary reason for using the F2 score over the F1 score is the imbalanced nature of our dataset. Certain labels are prevalent in the training set and our model will perform well in classifying images with these labels. On the other hand, the model is likely to perform poorly for the less prevalent labels and there is likely to be many false negatives for those labels. However, due to the imbalance in the dataset, there will be many more true positives than false negatives. We therefore use the F2 score to provide greater emphasis on recall and occurrences of false negatives.

The F2 score is a good indicator of the overall performance of our models, but we also examine confusion matrices to better appreciate how well our models are performing for each particular label. The confusion matrix captures precision and recall on a per label basis. We present visualisations of our confusion matrices to better demonstrate how well our models perform.

5. Results Discussion

Figure 6 shows the train and validation loss of 2 of our models throughout the training. This is important to study due to the nature of the line rather than the final loss, this can show the effects of overfitting and the issues when working with non-convex operations.

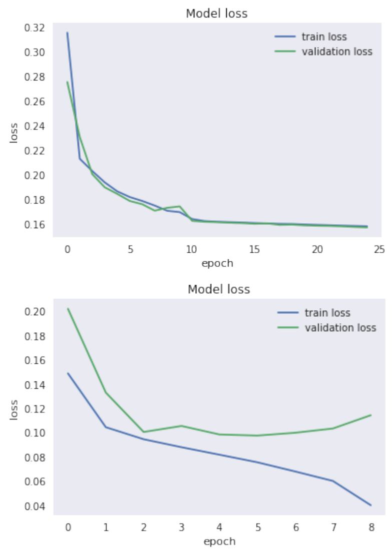


Figure 6. The plots of the losses of two of our models (best performing CNN and VGG respectively)

Figure 8 shows the evaluation tables of the models tests. These tables show the 17 labels of our data, the number of predictions of that label and the number of images with that label. It provides great insight into the way in which the models classify the separate labels, this breakdown is important when judging the effectiveness of our model.

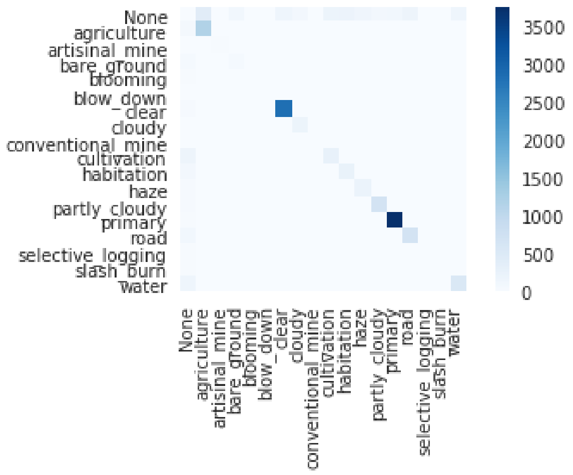


Figure 7. The confusion matrix of our VGG, this can help to give an indication of where the model may have gone wrong

The confusion matrix in Figure 7 has been created from the results of the VGG model, between this and the tables in Figure 8 the results are far easier to evaluate than just a number. They also provide other insights into the nature of the dataset which just having the accuracy cannot.

Model	Architecture	F2-Score	Accuracy
Model 0	3 Layer CNN	0.79	0.91
Model 2	5 Layer CNN	0.82	0.93
Model 3	8 Layer CNN	0.85	0.94
Ensemble	3 CNNs	0.84	0.99
VGG	VGG	0.92	0.96

Table 1. A final table of the results achieved by the models in this project

Finally, Table 1 shows an overview of the models we used in this project and their respective scores and accuracies.

Figure 6 showed the training and validation loss of the best performing CNN and VGG. This plot shows that despite the better performance, VGG performed best on Epoch 4. This is due to overfitting of the model, we used callbacks to save the weights of this Epoch, allowing us to use the most efficient model for our final F2-score. This early overfitting of the model could be due to the lack of dropout used when training the model, however, since we used fine-tuning and not a complete training of the model, dropout likely would not have fixed this issue. Meanwhile the CNN did not have the same issue of overfitting.

Upon completion of the final model, it is very clear to see some of the issues that working with a dataset with such an imbalance will provide. In particular, the inability to learn any of the features that make up some of the important labels in this dataset. When studying Figure 8, it is clear to see the issue at hand. The best performing CNN appeared to do very well in terms of both accuracy and F2-score however, it failed to classify 6 out of 17 of the labels correctly. This could become an issue when using a model to detect something in this dataset with few labels assigned to it, for example, conventional mining.

In the related works section we studied (Adke & Johnson, 2017), (Allen Zhao & Yu, 2017) and (Liang & Tang, 2017)'s papers on the same dataset. During most of the results sections of these papers they discussed how well the CNNs performed and when looking at the accuracy it is easy to see why. However, studying Figure 8 shows that CNNs fail to classify the labels less frequent in the training set very well. This is expected given the reliance of deep learning models on having a sizeable training set.

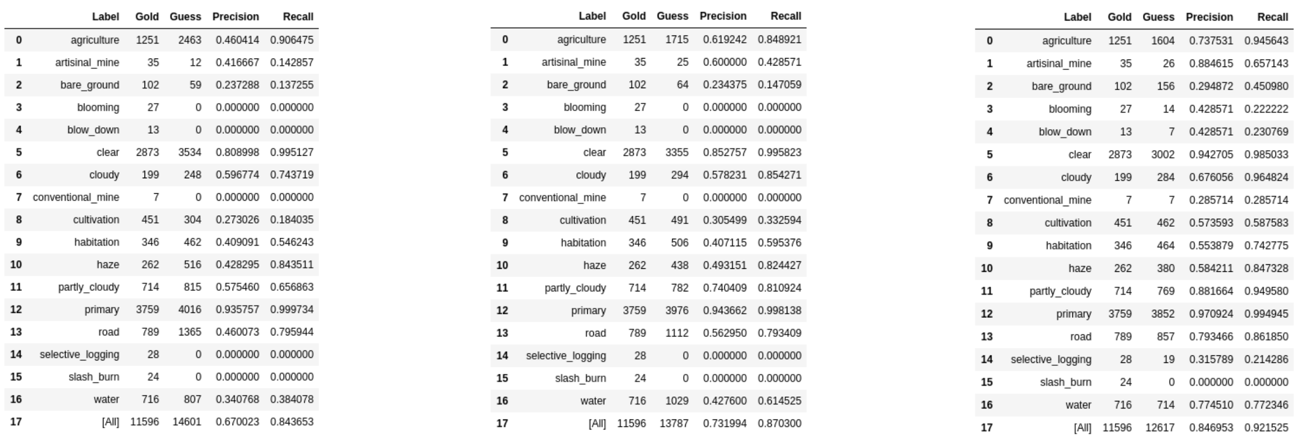
VGG was able to classify these 6 labels mentioned earlier. However it still performed poorly on these 6 labels. The performance on the 3 most commonly occurring labels in the dataset were able to artificially improve the results of this test.

Finally we tested an ensemble model of 3 CNNs where each of the individual CNNs is based off model 2. We hoped that having 3 separate models would improve our results and our test results confirmed this hypothesis. The original model 2 achieved an F2-Score of 83 and our combined model achieves an F2 score of 84. However, interestingly we also observe the highest accuracy of all of our models, but not the highest F2 score. It achieved a validation accuracy of 0.99, this is highest of all of our models. However, it only achieved an F2-score of 0.84, this is the biggest difference between all of the observed accuracy and F2-scores. This provides confirmation that accuracy is a poor indicator of model performance. When inspecting Figure 9 we can see that it failed to classify the less well represented classes at all, this is in line with the results from our CNNs. Furthermore, the model struggled to classify anything other than the primary labels, this could be why the validation accuracy is so high compared to the F2-score.

6. Conclusion and Future Work

We began our project with the aim of exploring the performance of different CNN architectures based on AlexNet. Our investigation of the effect of varying the number of convolutional layers and max pooling layers confirmed our expectation that additional convolutional and pooling layers lead to better results. Our investigation of the effectiveness of transfer learning and perform fine tuning using VGG-16 was also informative and allowed us to confirm that networks trained on a dataset based on images of objects can translate to satellite images. Finally, we were successfully able to implement our proposed alternative approach and it was interesting to observe that it did indeed result in an improved F2 score. Alongside the positive story with regards to performance improvements from exploring various architectures, the interesting tangential discovery we made was the poor performance of our models in classifying labels with few examples in our training set. This highlighted to us a limit of deep learning approaches.

(Otavio A. B. Penatti & dos Santos, 2015) has stated that the deep neural networks that are able to classify the ImageNet dataset well are able to classify satellite imagery. Since the paper was published there have been breakthroughs in classifying the ImageNet data for example, VGG-19, which builds upon the VGG-16 model we have used in this project. In ImageNet based competitions, VGG-19 has performed better than VGG-16, however due to memory and compute constraints this project used

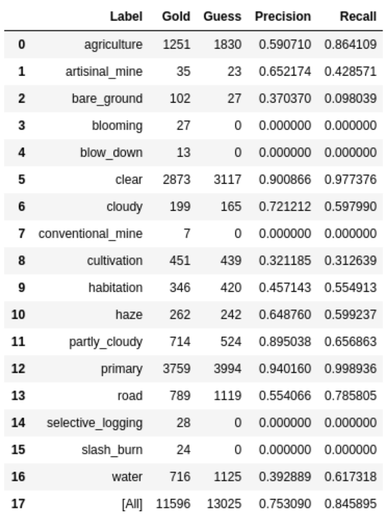


	Label	Gold	Guess	Precision	Recall
0	agriculture	1251	2463	0.460414	0.906475
1	artisinal_mine	35	12	0.416667	0.142857
2	bare_ground	102	59	0.237288	0.137255
3	blooming	27	0	0.000000	0.000000
4	blow_down	13	0	0.000000	0.000000
5	clear	2873	3534	0.808998	0.995127
6	cloudy	199	248	0.596774	0.743719
7	conventional_mine	7	0	0.000000	0.000000
8	cultivation	451	304	0.273026	0.184035
9	habitation	346	462	0.409091	0.546243
10	haze	262	516	0.428295	0.843511
11	partly_cloudy	714	815	0.575460	0.656863
12	primary	3759	4016	0.935757	0.999734
13	road	789	1365	0.460073	0.795944
14	selective_logging	28	0	0.000000	0.000000
15	slash_burn	24	0	0.000000	0.000000
16	water	716	807	0.340768	0.384078
17	[All]	11596	14601	0.670023	0.843653

	Label	Gold	Guess	Precision	Recall
0	agriculture	1251	1715	0.619242	0.848921
1	artisinal_mine	35	25	0.600000	0.428571
2	bare_ground	102	64	0.234375	0.147059
3	blooming	27	0	0.000000	0.000000
4	blow_down	13	0	0.000000	0.000000
5	clear	2873	3355	0.852757	0.995823
6	cloudy	199	294	0.578231	0.854271
7	conventional_mine	7	0	0.000000	0.000000
8	cultivation	451	491	0.305499	0.332594
9	habitation	346	506	0.407115	0.595376
10	haze	262	438	0.493151	0.824427
11	partly_cloudy	714	782	0.740409	0.810924
12	primary	3759	3976	0.943662	0.998138
13	road	789	1112	0.562950	0.793409
14	selective_logging	28	0	0.000000	0.000000
15	slash_burn	24	0	0.000000	0.000000
16	water	716	1029	0.427600	0.614525
17	[All]	11596	13787	0.731994	0.870300

	Label	Gold	Guess	Precision	Recall
0	agriculture	1251	1604	0.737531	0.945643
1	artisinal_mine	35	26	0.884615	0.657143
2	bare_ground	102	156	0.294872	0.450980
3	blooming	27	14	0.428571	0.222222
4	blow_down	13	7	0.428571	0.230769
5	clear	2873	3002	0.942705	0.985033
6	cloudy	199	284	0.676056	0.964824
7	conventional_mine	7	7	0.285714	0.285714
8	cultivation	451	462	0.573593	0.587583
9	habitation	346	464	0.553879	0.742775
10	haze	262	380	0.584211	0.847328
11	partly_cloudy	714	769	0.881664	0.949580
12	primary	3759	3852	0.970924	0.994945
13	road	789	857	0.793466	0.861850
14	selective_logging	28	19	0.315789	0.214286
15	slash_burn	24	0	0.000000	0.000000
16	water	716	714	0.774510	0.772346
17	[All]	11596	12617	0.846953	0.921525

Figure 8. The evaluation tables of the simple baseline CNN(left), best performing CNN(middle) and VGG(right).



	Label	Gold	Guess	Precision	Recall
0	agriculture	1251	1830	0.590710	0.864109
1	artisinal_mine	35	23	0.652174	0.428571
2	bare_ground	102	27	0.370370	0.098039
3	blooming	27	0	0.000000	0.000000
4	blow_down	13	0	0.000000	0.000000
5	clear	2873	3117	0.900866	0.977376
6	cloudy	199	165	0.721212	0.597990
7	conventional_mine	7	0	0.000000	0.000000
8	cultivation	451	439	0.321185	0.312639
9	habitation	346	420	0.457143	0.554913
10	haze	262	242	0.648760	0.599237
11	partly_cloudy	714	524	0.895038	0.656863
12	primary	3759	3994	0.940160	0.998936
13	road	789	1119	0.554066	0.785805
14	selective_logging	28	0	0.000000	0.000000
15	slash_burn	24	0	0.000000	0.000000
16	water	716	1125	0.392889	0.617318
17	[All]	11596	13025	0.753090	0.845895

Figure 9. An evaluation table of the Ensemble model used in our tests

only VGG-16. ResNet is another architecture that has provided breakthroughs in many fields of machine vision. Despite being discounted in (Adke & Johnson, 2017), this is currently state-of-the-art of for image classification and there is no reason not to believe that when correctly utilised this architecture can improve our results from VGG-16.

Another method for further investigation would be to remove the primary labels from this dataset, this would probably require a bigger dataset but could remove the issue that the skew in the data provided. Removing 'agriculture', 'clear' 'primary' and 'road' from this dataset could have allowed us greater insight into the strengths and weaknesses of the model. We have tried to combat this with the final ensemble model, and further investigation in this area could find interesting results.

Other than deeper networks, Capsule Networks (Sara Sabour & Hinton, 2017) have recently appeared as a way to overcome the issues that CNNs have with images that are augmented too much for the network to be able to make sense of the image. CapsNets introduce dynamic routing and routing by agreement, methods which aim to maintain all of the features from lower layers rather than just the strongest features in each layer. This new architecture has performed exceptionally well on the MNist dataset, scaling the model up to be used on bigger and more complex datasets may have proven difficult but seeing how the results translated would have been of interest.

As mentioned in the related works section, if we were looking to optimise the accuracy of the model on the dataset, then ensemble models may have been the optimal model for the purpose. With no constraints ensemble models, infrared channels being included, full data augmentation and full scale images could have been used to try and full optimise the model. This may have lead to far better results but for an overview and investigation of machine vision with satellite images, this was not needed in this project.

References

Adke, Rohisha and Johnson, Joe. Predicting land use and atmospheric conditions from amazon rainforest satellite imagery, 2017. URL http://cs231n.stanford.edu/reports/2017/pdfs/551.pdf.

Alex Krizhevsky, Ilya Sutskever and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks, 2010. URL https://papers.nips.cc/paper/4824-imagenet-classification-\with-deep-convolutional-neural-networks.pdf.

Allen Zhao, Shawn Hu and Yu, Chenyao. Classifying weather, terrain, and deforestation of the amazon using deep multi-task convolutional neural nets, 2017. URL http://cs231n.stanford.edu/reports/2017/pdfs/918.pdf.

Coffee. Coffee, 2014. URL http://www.patreo.dcc.ufmg.br/downloads/brazilian-coffee-dataset/.

CS231, Stanford. Convolutional neural networks. URL https://cs231n.github.io/convolutional-networks/.

Kaggle. Planet: Understanding the amazon from space, 2017. URL https://www.kaggle.com/c/planet-understanding-the-amazon\protect\discretionary{\char\hyphenchar\font}{}{}from-space.

Kaiming He, Xiangyu Zhang, Shaoqing Ren Jian Sun and Research, Microsoft. Deep residual learning for image recognition, 2010. URL https://arxiv.org/pdf/1512.03385.pdf.

Kingma, Diederik and Ba, Jimmy Lei. Adam: A method for stochastic optimization, 2015. URL https://arxiv.org/pdf/1412.6980.pdf.

Liang, Chao and Tang, Meng. Understand amazon deforestation using neural network, 2017. URL http://cs231n.stanford.edu/reports/2017/pdfs/904.pdf.

Otavio A. B. Penatti, Keiller Nogueira and dos Santos, Jefersson A. Do deep features generalize from everyday objects to remote sensing and aerial scenes domains?, 2015. URL https://www.cv-foundation.org/openaccess/content_cvpr_workshops_2015/W13/papers/Penatti_Do_Deep_Features_2015_CVPR_paper.pdf.

Pierre Sermanet, David Eigen, Xiang Zhang Michael Mathieu Rob Fergus and LeCun, Yann. Overfeat: Integrated recognition, localization and detection using convolutional networks, 2014. URL https://arxiv.org/pdf/1312.6229.pdf.

Popkin, Gabriel. Satellite alerts track deforestation in real time, 2016. URL http://www.nature.com/news/satellite-alerts-track-deforestation\protect\discretionary{\char\hyphenchar\font}{}{}in-real-time-1.19427.

Rhett, Butler. Calculating deforestation figures for the amazon, 2017. URL https://rainforests.mongabay.com/amazon/deforestation_calculations.html.

Saikat Basu, Sangram Ganguly, Supratik Mukhopadhyay Robert DiBiano Manohar Karki and Nemani, Ramakrishna. Deepsat a learning framework for satellite imagery, 2015. URL https://arxiv.org/pdf/1509.03602.pdf.

Sara Sabour, Nicholas Frosst and Hinton, Geoffrey E. Dynamic routing between capsules, 2017. URL https://arxiv.org/pdf/1710.09829.pdf.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition, 2015. URL https://arxiv.org/pdf/1409.1556.pdf.

UCMerced. Ucmerced, 2010. URL http://weegee.vision.ucmerced.edu/datasets/landuse.html.