Deep Learning for Terrain Classification and Segmentation

ABSTRACT

This project explores the use of Convolutional Neural Networks (CNNs) for classifying and segmenting high-resolution images from the DeepGlobe 2018 dataset. The study investigates three main tasks: binary classification of land types, estimation of land cover fractions, and pixel-wise segmentation. Introducing F1 optimised thresholding and data augmentation improved the prediction balance, with transfer learning with MobilNetV2 further enhancing performance and efficiency. For pixel-wise classification, U-Net architecture was implemented with a weighted sparse categorical cross-entropy loss function being defined to ensure the model trains over all classes. Training with 200 epochs achieved a mean Intersection over Union score of 0.30, with class specific scores as high as 0.69. These results demonstrate the value of deep learning in terrain classification but also highlight challenges in handling class imbalance and large datasets.

I. INTRODUCTION

Satellite imagery has revolutionised our ability to observe and analyse Earth's surface, providing crucial data for many applications such as monitoring tectonic activity, tracking pollution, and urban planning [1]. However, the sheer volume and complexity of this data leads to a laborious manual interpretation, which is becoming increasingly impractical given the rapid growth of high-resolution image availability. Machine learning techniques, particularly deep learning models like Convolutional Neural Networks (CNNs) have emerged as powerful tools for automating effective data analysis,

facilitating the development of automated systems capable of processing large datasets with great precision. For example, studies have shown that CNNs can accurately segment and map vegetation species with high precision, even using Google Earth satellite imagery [2], highlighting their utility in diverse applications, even with readily available public datasets.

In this study, we explore the application of CNN-based models to a series of classification and segmentation tasks involving satellite imagery. Specifically, we implement and evaluate three models: CNN binary classification for detecting specific land types, fractional estimation of land cover classes within each image, and pixel-level segmentation using a U-Net architecture. By leveraging pretrained models, the project aims to assess the effectiveness and limitations of deep learning models in high-resolution satellite image processing.

II. BACKGROUND

The DeepGlobe Dataset

The training dataset consists of 1146 high-resolution satellite images taken from the DeepGlobe 2018 Land Cover Classification dataset [3]. This dataset provided a standardised platform for the development and evaluation of the machine learning models created in this study. The images are focused on regions with a diverse mix of land cover, particularly in developing countries, where automated land cover

mapping is most useful [3]. The training dataset is a subset of 804 images, with 2448 x 2448 pixels at 50cm per pixel. Each image is paired with a RGB segmentation masks, that labels pixels into one of seven classes:

- Urban
- Agriculture
- Rangeland
- Forest
- Water
- Barren
- Unknown

The high number of pixels per image posed challenges for direct processing with CNNs due to memory constraints in Google Colab and high computational loads. Most standard CNN architectures are optimised for smaller inputs such as 256x256 [4], so this study uses both resizing and tiling (splitting images into smaller overlapping or non-overlapping windows) to preprocess the satellite images. Resizing reduces the resolution but retains global structure, while tiling preserves fine details at the expense of spatial context. This study investigates the performance trade-offs between the two approaches.

The Convolutional Neural Network

CNNs (Figure 1) offer several advantages that allows them to learn spatial features, such as edges, shapes, and textures which makes them particularly suited for image classification. Through operations like pooling, CNNs gain translational invariance, meaning they can recognise terrain classes regardless of their position in the image [5]. This is crucial for large image datasets where the same class will appear in all locations across many images, making the model more robust to translation, scale and distortion.

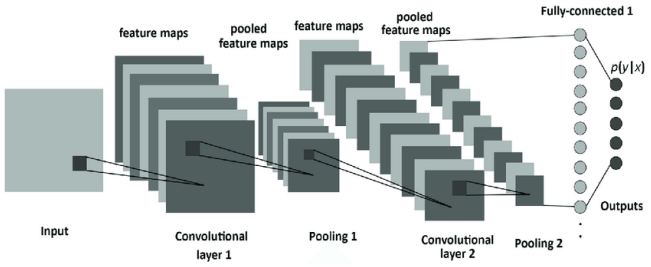

Figure 1: The structure of a CNN, consisting of convolutional, pooling, and fully connected layers Source: [6].

CNN Model Architecture

In our CNN, the first layer was a 2D convolutional layer, using 32 filters with a size of 3x3. This size was chosen as it is small enough to capture fine terrain features such as edges and textures, but without incurring slow computational speeds.

The Rectified Linear Unit (ReLU) activation function, shown in equation (1), was applied to introduce non-linearity, further enabling the network to model complex patterns. This is important because real world training data, such as satellite images, contains lots of variations that a linear model would not effectively handle [7].

$$\mathrm{ReLU}(x) = \max(0,x) \qquad (1)$$

In the hidden layers of the network, the ReLU takes a single input, x, and outputs x if the input is positive, or zero if the input is zero or negative, hence deciding if a neuron should be activated or not based on the positivity of the input. This means that at a given time only a subset of neurons is activated, making the network less dense and better focused on the more relevant features. This was a crucial advantage in choosing ReLU when working with large datasets of

high-resolution satellite images, where it is beneficial for processing speeds and efficient Random Access Memory (RAM) usage to ignore irrelevant input features. ReLU networks can be built with a greater number of layers without occurring problems such as vanishing gradient which can occur with other activations such as sigmoid, allowing for more effective backpropagation, and a faster learning network that is more resilient to changes in terrain, weather conditions, and image noise.

After the convolution, a max pooling layer was used to reduce the size of the feature maps. By taking the maximum value out of a pooling window, this layer keeps just the strongest terrain features, reducing the amount of computation required in following layers, increasing the speed and efficiency of the model. By reducing the spatial dimensions of the input for the next convolutional layer, it carries forward only the most significant elements, reducing computational complexity and overfitting to test images. This is because in satellite imagery, features such as buildings or trees appear in different locations, so by focusing on the strongest signals the exact location of individual features becomes less important. Therefore, this acts to supress any noise and hence the network becomes more resilient to translational variations.

Subsequent convolutional layers with increasing numbers of filters (64 and 128) allowed the model to learn more complex patterns in the train images. In a convolutional neural network, each layer learns a set of features based off the previous layer. The first layers detect simple features such as edges and textures due to their smaller receptive fields, then the following layers combine these features to detect more complex patterns such as trees, rivers and roads. As more layers were added, the receptive field was increased, allowing the network to identify features over larger areas of the training image, which is crucial in the model learning the broader contexts in which terrain features are likely to be found. Overall, the deeper the layer, the more abstract the features it can learn.

The final layer was a dense layer with 6 outputs (one for each class), in which all the neurons are connected to all the neurons in the previous layer, with each neuron calculating a weighted sum of its inputs (with a bias term) then applying the activation function. The role of the dense layer in this network was to interpret the features from the convolutional layers to classify the type of terrain present.

The Adam optimiser was chosen for its adaptive learning rate, and good performance. The Area Under the Curve (AUC) metric was used as an evaluation metric to reflect the model's ability to distinguish between classes.

A sigmoid activation function, shown equation (2), was used for the binary classification of terrain types, as it outputs a value between 0 and 1, allowing the model to predict the probability of each class being present in the image [8]. These outputs are mutually exclusive, ideal for binary classification where each image contains more than one class. The probability was then thresholded at 0.5 to decide whether the class is present or not.

$$f(x) = \frac{1}{1+e^{-x}} \qquad (2)$$

The U-Net Model

The U-Net is a type of CNN that follows a U-shaped structure, where the images flow through an encoder, then a bottleneck, followed by a decoder, with skip

connections linking the encoder and decoder layers. The encoder consisted of repeated 3x3 convolutional layers followed by a ReLU activation function and a 2x2 max pooling operation, which gradually reduces the spatial dimension of the image while learning to detect terrain features. The bottleneck sat at the deepest part of the network and captures the most abstract features of the image. The decoder then upsampled these features using transposed convolutions, to reconstruct the original spatial dimensions while predicting a class for each pixel. Skip connections passed information from early layers in the encoder directly to corresponding layers in the decoder, helping to make the predictions more accurate by combining local and global information.

The final layer of the U-net model used a softmax activation function, equation (3), to produce pixel-wise probabilities. This layer applied a 1x1 convolution to reduce the number of output channels to match the number of terrain classes, followed by a softmax activation to convert the logits into probabilities [9].

$$f_n(x) = \frac{e^{x_n}}{\sum_{m=1}^{M} e^{x_m}} \quad for\ n = 1,2 \dots, M \qquad (3)$$

Softmax function, where x_n is the input (logit) for class n, and M is the total number of classes.

This structure made U-Net especially effective for segmentation tasks, where each pixel needs to be classified separately and efficiently.

MobileNetV2 Pretrained Model

To enhance performance, while reducing training time, this project leveraged a pretrained MobileNetV2 base model, allowing our model to use previously learned low-level features such as edges, textures, and shapes. The project used the version of MobileNetV2 trained on ImageNet, a large dataset of over a million natural images, allowing our project to best utilise the networks learned natural features for satellite images. This model is one of the most efficient pretrained deep learning models for use in classifying satellite images, due to it being developed for devices with limited resources, such as this projects limitations using Google Colab [10].

Loss Functions

Binary cross-entropy was used when the model outputted a probability, and the task is to classify the data into two classes - in this case presence or absence of a terrain feature. In a multi-label classification setting such as this, the binary cross-entropy loss function was applied to each class separately, resulting in independent class predictions.

Sparse categorical cross-entropy was used for multi-class classification, where each input only belongs to one class (e.g. pixels). It compares the probability of the correct class to the true label, as shown in equation (4) [11].

$$\mathcal{L} = -\sum_{x_i \in I} \log p(y_i \mid x_i) \qquad (4)$$

III. METHOD & RESULTS

Data Preprocessing

Each image from the DeepGlobe dataset was divided into smaller non-overlapping tiles of 256x256 pixels to allow for input into the CNN. This had the advantage of preserving detail in the images, as the original resolution is kept, but had the disadvantage of the model only seeing a ninth of the image at a time, potentially missing the broader patterns. It also meant much slower computational speeds, as every

pixel was kept. Tiling was initially chosen over reshaping to preserve the details in the image, and train the model better on finer, more local features.

The corresponding masks were also tiled and were converted from RGB values to class index for easier processing. If an image contained a pixel with a recognised RGB colour, it was assigned a label of 1 for that corresponding class, and hence a binary multi-class label was created for each image.

Some validation images were kept separate from the model for testing, to evaluate the models performance on unseen data. Keeping these images separate right up until testing was crucial, as if the model had already seen these images in training, then evaluation of the model would have been an inaccurate representation of its true performance.

Task 1: Binary Terrain Classification

Model 1 was a simple CNN, adapted for use with low RAM. It adopted a sliding window method, where one image was taken at a time, split into tiles, the model trained over the tiles, then the loop moved onto the next image. This prevented all the images from being loaded onto the RAM at one time, which caused runtime crashes. This method was unsuccessful and predicted only class 1 – agriculture – for every image. This was due to the unequal class distribution, shown Table 1, with class 1 being in 81% of the images. As the model was trained using binary cross-entropy, the average loss over all classes was minimised, and since agriculture appears in the majority of training samples the model learned that predicting just this class resulted in a low overall loss, even if it gets the rare classes wrong.

Class	Urban	Agriculture	Range	Forest	Water	Barren	Unknown
Freq.	649	722	522	194	478	428	159

Table 1: Class distribution in the 804 test images.

Model 2 was created to address these key limitations using two strategies: binary threshold adjustment and data augmentation for rare classes.

The binary threshold of 0.5 favoured frequent classes and supresses rare ones. To correct this, optimal thresholds were calculated using the optimal F1 score, shown in Table 2. For each class the precision recall class was calculated using the models predictions on the validation images. For each threshold a F1 score was calculated (the average of precision and recall [12]) and the threshold that maximises F1 score calculated. This allowed the model to make better predictions for rare classes by setting lower thresholds and avoid false positives in common classes with higher thresholds.

Class	Urban	Agriculture	Range	Forest	Water	Barren	Unknown
Thres hold	0.31	0.69	0.29	0.21	0.23	0.25	0.06

Table 2: Optimal binary threshold for each class.

Data augmentation was implemented to duplicate a tile multiple times, as specified by a boost factor, if it contained any of the defined rare classes to artificially increase its presence in the dataset. Although no spatial transformations were applied to the augmented images for this model, the simple duplications allowed for the model to train on a more balanced dataset, improving the models ability to detect classes other than agriculture.

This had a positive effect on the predictions' accuracies with it now predicting Urban and Agriculture for every image. Incrementally adjusting

the thresholds down further allowed for more classes to be predicted, however the predicted labels were still identical for all images. This suggests a fundamental issue with the model or data preparation, leading to a new pretrained model framework to be adopted.

Model 3 builds on this improvement by leveraging transfer learning though a pretrained model, MobileNetV2. The image tiles were precalculated and saved to avoid recalculation when training, improving model efficiency. The base model was used with the original classification layers removed, such that it outputs a spatial feature map, which was then passed through:

- A GlobalAveragePooling2D layer to flatten the spatial features
- A dense layer with ReLU activation to introduce non-linearity and learn satellite image specific features
- A 0.3 rate dropout layer to prevent overfitting to train images
- A final dense layer with sigmoid activation to produce independent probabilities for each terrain class

As the MobileNetV2 model is pretrained, it improved the performance of the model, making up for the imbalanced dataset.

Model 4 improved the efficiency of Model 3 by training in 'chunks' of 3000 tiles at once, which still conserved memory by not loading all images, but decreased time spent training the model. However as this was an efficiency change, the model still predicted the same label for every image, identifying only Agriculture and Unknown, suggesting a change of approach was needed.

Model 5 introduced a shift in strategy by replacing the tile-based approach with reshaping the images. This change was motivated by the need to improve training speed and include the context of a full satellite image rather than isolated tiles. Each image was resized to 256x256 pixels and saved to a NumPy array to avoid recalculation. The images were then normalised and binary labels generated.

To retain the same benefits as Model 4, the images were passed through the MobileNetV2 model. By reshaping instead of tiling, Model 5 significantly reduced training time, and no longer needed to be trained in a loop as fewer images needed to be processed per epoch, hence the whole dataset could be loaded onto the memory at once.

Model 5 again generated the same label for every image, depending on the threshold adjustments, however the F1 scores greatly improved, with urban, rangeland and water all being above 0.8, with barren and unknown also well above the 0.5 mark.

```
Classification Report:
              precision    recall  f1-score   support

       Urban       0.70      1.00      0.82         7
 Agriculture       0.00      0.00      0.00         9
   Rangeland       0.70      1.00      0.82         7
      Forest       0.20      1.00      0.33         2
       Water       0.86      0.75      0.80         8
      Barren       0.60      1.00      0.75         6
     Unknown       0.50      1.00      0.67         5

   micro avg       0.58      0.75      0.65        44
   macro avg       0.51      0.82      0.60        44
weighted avg       0.53      0.75      0.60        44
 samples avg       0.57      0.73      0.64        44

Percentage of each class correctly identified (when present):
Urban               : 100.0%
Agriculture         : 0.0%
Rangeland           : 100.0%
Forest              : 100.0%
Water               : 75.0%
Barren              : 100.0%
Unknown             : 100.0%
Confusion Matrix:
[[ 2 24]
 [11 33]]
Overall label-wise correctness score: 0.500 (50.0%)

Correctness per class label:
Urban               : 0.700 (70.0%)
Agriculture         : 0.100 (10.0%)
Rangeland           : 0.700 (70.0%)
Forest              : 0.200 (20.0%)
Water               : 0.700 (70.0%)
Barren              : 0.600 (60.0%)
Unknown             : 0.500 (50.0%)
```

Figure 2: Classification Report for Model 5, showing the F1 scores and class correctness percentages.

Figure 2 shows that the aims of Task 1 had been partially met, the model trained over common classes well, but failed to train well over the rare classes, leading to incorrect binary class labels.

Task 2: Class Fractions

The goal of Task 2 was to calculate the fraction of each image each class represents. The true class fractions were calculated by summing how many pixels belonged to each class, then dividing by the total number of pixels.

For estimating class fractions, the model used the same structure, but instead of treating the sigmoid outputs as binary, they were used as fractional values. This had the limitation of treating all class outputs independently, so the fractions did not by definition sum to one, however the binary model was accurate in identifying individual class fractions, such that the average images probabilities summed to 1.019 Which was sufficiently close for this use case.

By comparing these predicted fractions to the true ones, we were able to measure how accurately the model estimated the relative coverage of each class. This was more useful than the binary labels, especially in use cases where the distribution of land types is more important than their locations.

Due to the inaccuracies of the binary model, the fractions produced were also the same for every image, with 100% Forest identified.

This task was revisited using the more accurate models defined in Task 3.

Task 3: Pixel identification

The goal of Task 3 was to perform pixel-wise segmentation of the satellite images, assigning a class to every pixel in the image. To achieve this, a U-Net model was used, a CNN architecture that is popular for this task.

Following the effectiveness of Task 1 Model 5, the model was trained using the resized image dataset. U-Net model 1 has 14 layers, consisting of 6 encoder layers, 2 bottleneck, and 6 decoder layers. When trained on 10 epochs the accuracy was high at 0.6774, however the loss was also high at 0.9845. U-Net Model 2 was created with more layers (12 encoder, 2 bottleneck, 12 decoder) and had similar results of 0.6416 and 1.0049 respectively, as shown in Figures 3 and 4. This suggested the image segmentation was poor as the two models were just learning the most common terrain types, as shown in Figure 5, and predicting them for every image.

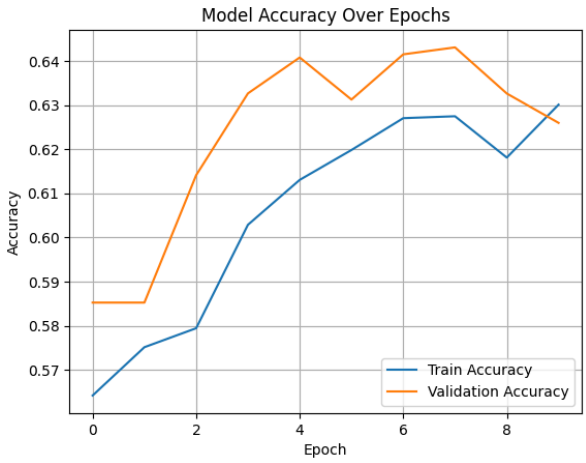

Figure 3: Model 2 accuracy over epochs.

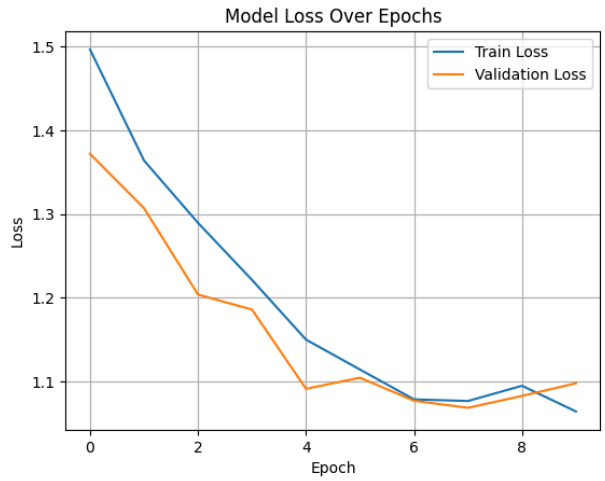

Figure 4: Model 2 loss over epochs.

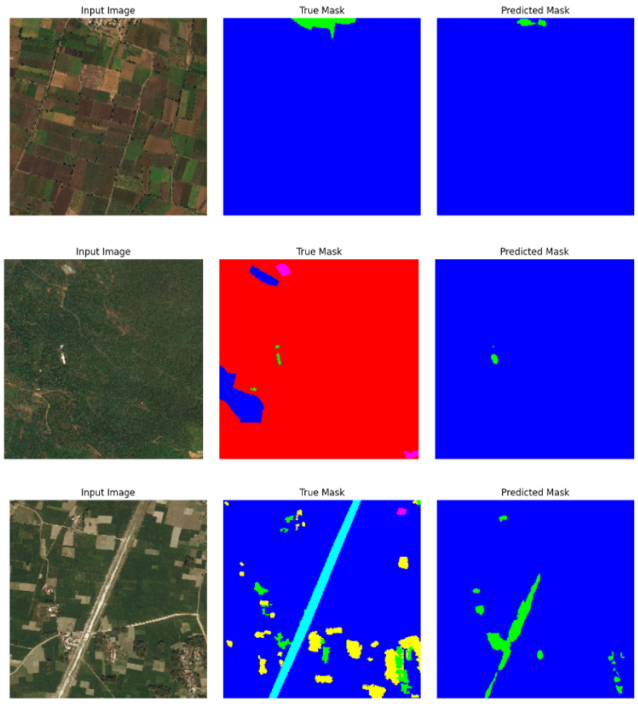

Figure 5: Model 2 predicted masks, showing the overprediction of class 1 – agriculture.

To fix this, in U-Net Model 3 a custom loss function was implemented. This function defined 'weighted sparse categorical cross-entropy' was a modified version of the standard sparse categorical cross-entropy (SCCE) loss with weights added to each pixel's loss contribution [13]. The weights were applied to each class proportionally to the pixel distribution, with the more underrepresented the class the greater weight assigned to it.

The unweighted SCCE for one pixel is shown by equation (5) [14].

$$\mathcal{L} = -\log\left(\hat{y}_{i,y_i}\right)$$

Where $\hat{y}$ is the predicted probability for the true class, y. To address the class imbalance, a weight, w, is introduced in equation (6) [14].

$$\mathcal{L}_{weighted} = -w_i \log\left(\hat{y}_{i,y_i}\right) \quad (6)$$

Where the weight for each class, c, for C total classes is defined as:

$$w_c = \frac{1/f_c}{\frac{1}{C}\sum_{i=0}^{C} 1/f_i} \quad (7)$$

For a dataset of M samples, the total loss is the average loss over all pixels, as shown in equation (8) [14].

$$\mathcal{L}_{weighted} = \frac{1}{M}\sum_{i=1}^{M} - w_i \log\left(\hat{y}_{i,y_i}\right) \quad (8)$$

This penalised the model more for incorrect rare classes, meaning the model could no longer just predict the dominant class to get the best result, and was instead learning on all classes including the rare ones.

The data was augmented using spatial transformations, flipping the image randomly across either axis, giving the model a larger pool of test images to learn from.

The overall accuracy of the model was lower than before at 0.2457, however this is a misleading statistic, as the model can no longer just predict dominant classes (Figure 6). The overall loss of the model is much improved at 0.0326, highlighting the

increased performance due to the class weights (Figure 7).

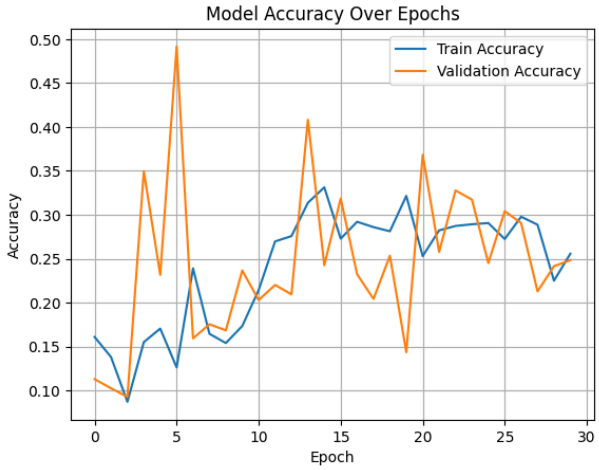

Figure 6: Model 3 accuracy over epochs.1

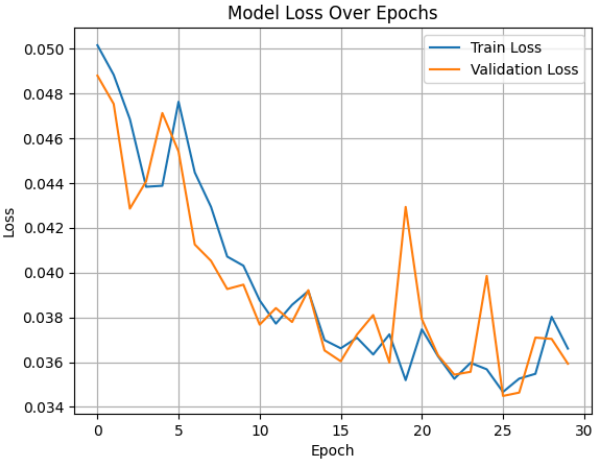

Figure 7: Model 3 loss over epochs.

The model recognised the shapes of the classes within the images well, however, struggled to classify the pixels into the correct class. The true masks only had a few blocks of colour per image; however the predicted masks were speckled with lots of classes. This suggests the model was too confident and is overfitting to noise such as variations in colours and textures, shown in the different types of fields in Figure 8, image 2.

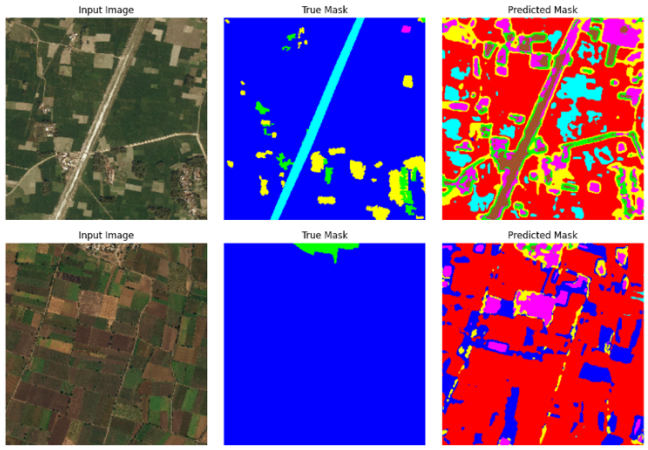

Figure 8: Model 3 predicted masks, finding the correct shaped but mislabelling the classes.

The final U-Net Model 4 added a dropout in the decoder before the output layer to make the predictions less irregular. The augmentation of rare classes was also slightly reduced to prevent overfitting. The effect increasing epochs had on accuracy was also investigated, with iteration one training with 60 epochs increasing the accuracy significantly to 0.4559 (Figure 9).

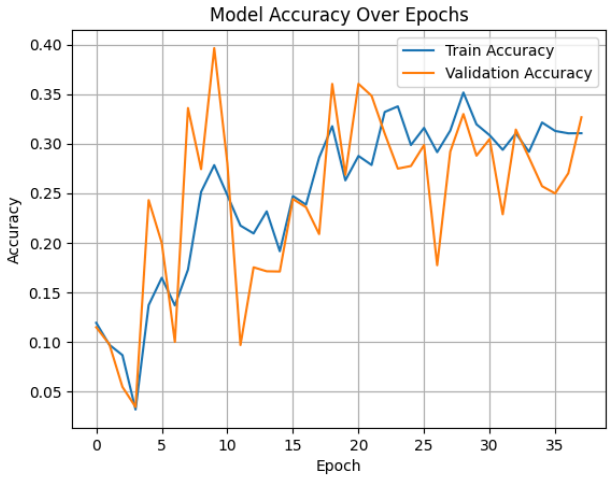

Figure 9: U-Net Model 4 accuracy over 60 epochs, showing a continuous gradual increase, suggesting further training is possible.

However the predicted labels produced by U-Net Model 4 experienced even more noise. The test image shown in Figure 10 shows the correct identification

of urban land highlighted in green, however with lots of noise around it.

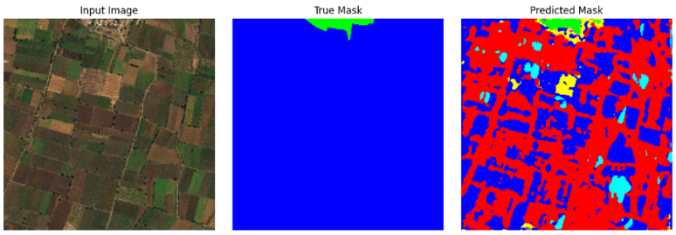

Figure 10: Model 4 predicted labels after training on 60 epochs.

For the investigation into increasing epochs, a new performance metric was defined: the intersection over union (IoU). The IoU (equation 9) measures the overlap between the predicted segmentation, P, and the true mask, T, providing a better assessment of how well the model performs pixel-wise [15].

$$IoU = \left|\frac{T \cap P}{T \cup P}\right| \quad (9)$$

A score above 0.5 is considered good for complex images such as these, with scores closer to zero meaning no overlap between the masks at all.

Class	IoU
0	0.4521
1	0.3426
2	0.1055
3	0.3916
4	0.1928
5	0.1072
6	0.0043

Table 3: U-Net Model 4 IoU scores after training on 60 epochs. The mean IoU is 0.2280.

The mean IoU of 0.2280 suggested the model had captured basic class boundaries but had lots of room to improve. The accuracy over epoch, shown in Figure 8, was still slowly climbing after 60 epochs, suggesting the model could still be trained further. The number of epochs was therefore increased to 200 to ensure the model was fully trained.

Training on 200 epochs increased the accuracy to however the accuracy plateaued at epoch 175, shown in Figure 11, suggesting the model is fully trained.

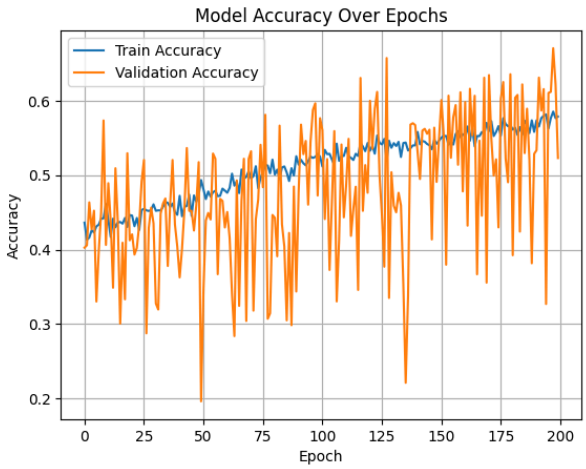

Figure 11: U-Net Model 4 accuracy over 200 epochs.

Class	IoU
0	0.4935
1	0.4117
2	0.1552
3	0.6861
4	0.1585
5	0.2129
6	0.0048

Table 4: U-Net model 4 IoU scores after training on 200 epochs.

This greatly improved the IoU for every class, with more classes being correctly predicted, reflected in the increased mean IoU of 0.3032. The model still did not learn the unknown class, however this is to be expected. The unknown class is made of ambiguous regions such as clouds, mixed terrain, terrain outside of classes, meaning the class as a whole lacks consistent features and patterns. Because these

regions do not follow a predictable visual pattern, the model receives ambiguous or conflicting training each time it encounters the class. To improve overall accuracy the unknown class could be removed from the model entirely, as the unknown class is rare.

TASK 2 – Revisited

With the more accurate U-Net model 4, class fractions were calculated by summing how many pixels in the mask belong to each class then dividing by the total number of pixels.

Due to the extra class noise on the images, the fractions calculated do not completely represent those of the true mask, shown Figure 12, but have increased accuracy compared to the CNN models.

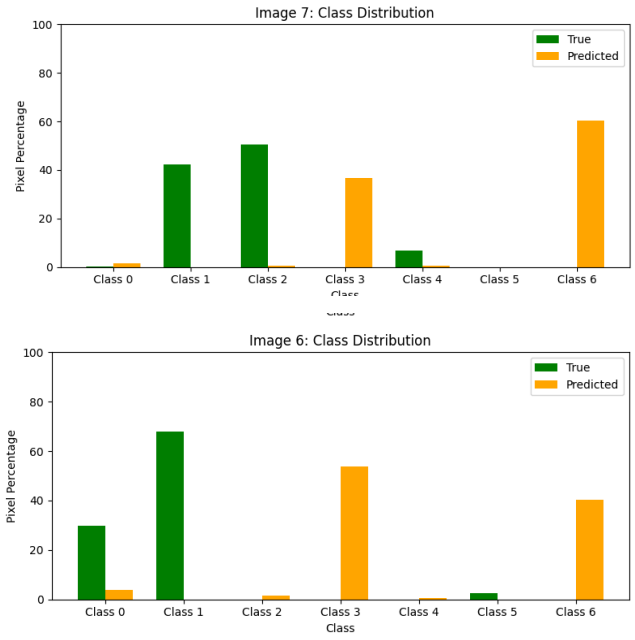

Figure 12: Two example plots for class fractions, comparing true fractions to predicted.

Figure 12 reflects the overall trend of the U-Net Model 5, that the model can predict class shapes present in the image, in these cases two dominant classes in the image, but fails to correctly determine the class.

IV. CONCLCUSION

In Task 1, the goal was to identify the presence of terrain classes in satellite images using binary classification labels. Initial CNN models struggled to label the unseen test images correctly due to class imbalance, consistently predicting the dominant class (agriculture). By adjusting the classification thresholds using F1 optimisation and applying data augmentation to rare classes, Model 2 improved the prediction balance, however consistent predictions across images revealed underlying issues in the model. Introducing MobileNetV2 and reshaping the images helped address these challenges, improving classification for rarer classes, however an overall successful model was not created.

Predictions for Task 2 were initially derived directly from the output probabilities of the binary classification models. While this model tended to overestimate dominant classes, using sigmoid outputs as fractional indicators proved to be useful and a potential route for investigation with a working CNN model. Later results showed that the improved U-Net model was more effective at generating the fractions in this study, however it was not with its own limitations.

In Task 3, U-Net architectures were used for pixel-wise segmentation. Early models also struggled with overfitting to dominant classes in the train images, resulting in visually inaccurate predictions. Implementing a weighted sparse categorical cross-entropy loss function assigned greater importance to the underrepresented classes, significantly improving the models ability to learn from all classes. Although not completely successful, I believe this method, with improvement, could be very effective at classifying large datasets with a varied class distribution.

Overall, the project demonstrates that while pretrained models and class balancing techniques can significantly improve terrain classification, challenges remain due to class imbalance and the complexity of high-resolution data. While the aims of these three tasks were not fully met by the models used in this project, they demonstrated the potential of deep learning models in classifying satellite imagery, with encouraging initial results. These models have many applications in mapping land coverage from urban sprawl in infrastructure planning to crop identification and monitoring in agriculture. By addressing the structural issues in the model, more advanced data augmentation, and the noise caused by overfitting, future work could achieve greater accuracy in terrain classification.

V. REFERENCES

[1]
P. Upadhyay and S. Gupta, "Introduction To Satellite Imaging Technology And Creating Images Using Raw Data Obtained From Landsat Satellite," vol. 1, no. 2, pp. 41–45, Oct. 2012, Available: https://www.researchgate.net/publication/306509020

[2]
S. Watanabe, K. Sumi, and T. Ise, "Identifying the vegetation type in Google Earth images using a convolutional neural network: a case study for Japanese bamboo forests," *BMC Ecology*, vol. 20, no. 1, Nov. 2020, doi: https://doi.org/10.1186/s12898-020-00331-5.

[3]
I. Demir *et al.*, "DeepGlobe 2018: A Challenge to Parse the Earth through Satellite Images," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 172–17209, Jun. 2018, doi: https://doi.org/10.1109/CVPRW.2018.00031.

[4]
A. Naik, K. Parihar, K. Lartius, and G. Waghale, "Comparative Analysis of VGG16, RESNET50, AND CNN Models for Lung Disease Prediction: A Deep Learning Approach," *International Journal for Research in Applied Science & Engineering Technology*, vol. 12, no. 5, Apr. 2024, Available: https://www.ijraset.com/research-paper/comparative-analysis-of-vgg16-resnet50-and-cnn-models

[5]
Myburgh, Johannes C, C. Mouton, and Davel, Marelie H, "Tracking translation invariance in CNNs," *arXiv.org*, 2021. https://arxiv.org/abs/2104.05997 (accessed Apr. 03, 2025).

[6]
S. Albelwi and A. Mahmood, "A Framework for Designing the Architectures of Deep Convolutional Neural Networks," *Entropy*, vol. 19, no. 6, p. 242, May 2017, doi: https://doi.org/10.3390/e19060242.

[7]
N. Kulathunga, N. R. Ranasinghe, D. Vrinceanu, Z. Kinsman, L. Huang, and Y. Wang, "Effects of Nonlinearity and Network Architecture on the Performance of Supervised Neural Networks," *Algorithms*, vol. 14, no. 2, p. 51, Feb. 2021, doi: https://doi.org/10.3390/a14020051.

[8]
S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," *Neurocomputing*, vol. 503, pp. 92–108, Sep. 2022, doi: https://doi.org/10.1016/j.neucom.2022.06.111.

[9]
R. Pathak, "Classification of Fruits using Convolutional Neural Networks and Transfer Learning Models," *Journal of Management Information and Decision Sciences*, vol. 24, no. 3, Jan. 2021.

[10]
B. Wijaya, P. Jaya Gea, A. Delano Gea, A. Sembiring, and C. Hutagalung, "SatelliteImages Classification using MobileNet V-2 Algorithm," *Jurnal dan Penelitian Teknik Informatika*, vol. 7, no. 4, Oct. 2023.

[11]
A. Panteli, J. Teuwen, H. Horlings, and E. Gavves, "Sparse-shot Learning with Exclusive Cross-Entropy for Extremely Many Localisations," 2021. Accessed: Apr. 03, 2025. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2021/papers/Panteli_Sparse-Shot_Learning_With_Exclusive_Cross-Entropy_for_Extremely_Many_Localisations_ICCV_2021_paper.pdf

[12]
Z. C. Lipton, C. Elkan, and B. Narayanaswamy, “Thresholding Classifiers to Maximize F1 Score,” *Cornell University*, May 13, 2014. https://arxiv.org/abs/1402.1892
[13]
Y. Ho and S. Wookey, “The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling,” *IEEE Access*, vol. 8, pp. 4806–4813, 2020, doi: https://doi.org/10.1109/access.2019.2962617.
[14]
J. Terven, D. Cordova-Esparza, A. Ramirez-Pedraza, and E. Chavez-Urbiola, “Loss Functions and Metrics In Deep Learning,” 2023. Available: https://arxiv.org/pdf/2307.02694
[15]
F. van Beers, A. Lindström, E. Okafor, and M. Wiering, “Deep Neural Networks with Intersection over Union Loss for Binary Image Segmentation,” *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods*, vol. 1, 2019, doi: https://doi.org/10.5220/0007347504380445.