Satellite Image Classification

Aakash Rathee
AI and Cybersecurity
University of Klagenfurt
aakashra@edu.aau.at

***Abstract*—Satellite images has become a key in collecting geographical data and has helped in many sectors of the economy by contributing to a faster & better planning. A satellite image classification system can speed up the inspection of images and help take faster decisions. The task of satellite image classification falls under the umbrella of computer vision and is addressed as a multi class single label supervised learning task. Four different CNN architectures were tested based on the results of previous work. The viability of a feature extractor was also investigated, which did not provide any significant increase in the model's performance. The model was trained on a dataset containing 57810 annotated images distributed into 33 different classes. The model achieved high accuracy, precision, recall and f1 score on the test set. Analysis of the results showed that single label classification is not the best approach for a dataset that contains high correlation between its classes.**

I. Introduction

Satellite images provide valuable data about a geographical location which acts as an indispensable resource for decision-makers and professionals. It has applications in many fields and industries such as meteorology, oceanography, fisheries, agriculture, biodiversity, geology, cartography, land use planning, and military, etc.[1] With the introduction of satellite imagery the time required to collect data has reduced drastically. But due to the complexity of data, humans were still required to manually inspect and label each single image. A system which is able to automate such a tedious task would drastically reduce the time required to process such vast amount of data.

Deep learning algorithms such as Convolutional Neural Networks (CNNs) are great at extracting and detecting features from images and have shown remarkable results in many different applications. It has also been the dominant approach for many computer vision tasks such as image/ video recognition, detection and classification[2]. The task of labelling satellite image is ideal for deep learning, which belongs to image classification in computer vision and multi-class single-label in supervised learning.

Use of deep learning for satellite image classification is already a well known research topic, this problem lies at crossroads of remote sensing, computer vision, and machine learning. The objective of this paper is create a model which will be able to identify what the land is being used for. To train such a model a large dataset of labelled images of different land uses is required. A CNN architecture must be selected which is able to obtain sufficiently good results overall and class-wise accuracy, precision, recall and f1-score. In addition testing must be done to check the viability of the model in real world scenario.

II. Datasets

A. Aerial Imagery Dataset (AID)

It consists of 9,780 images in total distributed into 30 different classes: *airport, bare land, baseball field, beach, bridge, center, church, commercial, dense residential, desert, farmland, forest, industrial, meadow, medium residential, mountain, park, parking, playground, pond, port, railway station, resort, river, school, sparse residential, square, stadium, storage tanks and viaduct.* With each class containing 220 - 420 images. Each image is of the dimension 600x600 pixel with a spatial resolution ranging from 0.5m - 8m[3].

The images were extracted from Google Earth from different countries and regions around the world, mainly China, the United States, England, France, Italy, Japan, Germany[3].

B. PatternNet

It consist of 30,400 images in total uniformly distributed into 38 different classes: *airplane, baseball field, basketball court, beach, bridge, cemetery, chaparral, Christmas tree farm, closed road, coastal mansion, crosswalk, dense residential, ferry terminal, football field, forest, freeway, golf course, harbor, intersection, mobile home park, nursing home, oil gas field, oil well, overpass, parking lot, parking space, railway, river, runway, runway marking, shipping yard, solar panels, sparse residential, storage tank, swimming pool, tennis court, transformer station and wastewater treatment plant.* Each image is of the dimension 256x256 pixel with a spatial resolution ranging from 0.062m - 4.693m. The images were extracted from Google Earth imagery and Google Map API for US cities[4].

C. RESICS45

It consists of 31,500 images total uniformly distributed into 45 different classes: *airplane, airport, baseball diamond, basketball court, beach, bridge, chaparral, church, circular farmland, cloud, commercial area, dense residential, desert, forest, free-way, golf course, ground track field, harbor, industrial area, intersection, island, lake, meadow, medium residential, mobile home park, mountain, overpass, palace, parking lot, railway, railway station, rectangular farmland, river, roundabout, runway, sea ice, ship, snowberg, sparse residential, stadium, storage tank, tennis court, terrace, thermal power station, and wetland.* Each image is of the dimension 256x256 pixel with a spatial resolution ranging from 0.2m - 30m. The images were extracted from Google Earth[5].

D. UC Merced Land Use

It consist of 2100 images in total uniformly distributed into 21 different classes: *agricultural, airplane, baseball diamond, beach, buildings, chaparral, dense residential, forest, freeway, golf course, harbor, intersection, medium density residential, mobile home park, overpass, parking lot, river, runway, sparse residential, storage tanks, and tennis courts.* Each image is of the dimension 256x256 pixel with a spatial resolution of 1 foot (0.3m)[6].

The images were extracted from large images from the United States Geological Survey National Map Urban Area Imagery collection of the following cities of United State: Birmingham, Boston, Buffalo, Columbus, Dallas, Harrisburg, Houston, Jacksonville, Las Vegas, Los Angeles, Miami, Napa, New York, Reno, San Diego, Santa Barbara, Seattle, Tampa, Tucson, and Ventura[6].

E. Combined Dataset

A new dataset was created by combining the previous 4 datasets (AID, PatternNet, RESICS45 and UC Merced Land Use). This dataset contains 33 different classes. Certain classes such as Agricultural in UC Merced Land Use, Circular Farmland and Rectangular Farmland in RESISC45, and Farmland in AID were combined into one single class. Table I shows the components of each class of the combined dataset and the classes which were used

I. COMPONENTS OF COMBINED DATASET'S CLASSES

Class	AID	PatternNet	RESISC45	UC Merced	# Images
Agricultural	Farmland		Circular Farmland, Retangular Farmland	Agricultural	1870
Airport	Airport	Airplane	Airplane	Airplane	2660
Bareland	Bareland				310
Beach	Beach	Beach	Beach	Beach	2000
Bridge	Bridge	Bridge	Bridge		1860
Buildings	Commercial, School		Commercial Area	Buildings	1450
Cemetery		Cemetery			800
Center	Center				260
Chaparral		Chaparral	Chaparral	Chaparral	1600
Coastal Mansion		Coastal Mansion			800
Desert	Desert		Desert		1000
Forest	Forest	Forest	Forest	Forest	1850
Freeway	Viaduct	Freeway, Overpass	Freeway, Overpass	Freeway, Overpass	3620
Golf Course		Golf Course	Golf Course	Golf Course	1600
Harbor	Port	Ferry Terminal, Harbor	Harbor, Ship		3480
Industrial Area	Industrial		Industrial Area		1090
Intersection		Crosswalk, Intersection	Intersection, Roundabout	Intersection	3100
Meadow	Meadow		Meadow		980
Mobile Home Park		Mobile Home Park	Mobile Home Park	Mobile Home Park	1600
Oil Gas Field		Oil Gas Field, Oil Well			1600
Park	Park				350
Parking	Parking	Parking Lot, Parking Space	Parking Lot	Parking Lot	2790
Railway	Railway Station	Railway	Railway, Railway Station		2460
Residential	Dense Residential, Medium Residential	Dense Residential	Dense Residential, Medium Resident	Dense Residential, Medium Residential	3100
Runway		Runway	Runway	Runway	1600
Shipping Yard		Shipping Yard			800
Sparse Residential	Sparse Residential	Sparse Residential	Sparse Residential	Sparse Residential	1900
Square	Square				330
Stadium	Stadium		Stadium		990
Storage Tank	Storage Tanks	Storage Tank	Storage Tank	Storage Tanks	1960
Thermal Power Station		Transformer Station	Thermal Power Station		1500
Wastewater Treatment Park		Wastewater Treatment Park			800
Wetland			Wetland		700

There are 4 classes (Bareland, Center, Park and Square) which have less than 500 images. In order to make more uniform distribution, image augmentation was done on these 4 classes. This will help in reducing the bias of the model. 4 different transformations were applied to all the images along with a random rotation. Fig. 1 shows how the augmentation changed the images.

- Increase sharpness
- Decrease sharpness
- Change Contrast
- Gaussian Blur

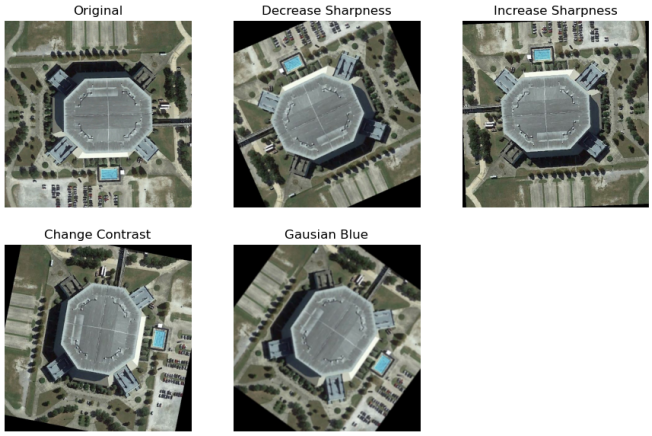


Fig. 1 Example of image augmentation

After the augmentation the total size of the dataset was 57810 images with each class contain images between 700 and 3620. Fig. 2 shows the distribution of images in each class of the combined dataset. In the dataset AID, PatterNet, RESISC45 and UC Merced contributed 12710, 21600, 21700 and 1800 images with a contribution percentage of 21.99%, 37.36%, 37.54% and 3.11% respectively.

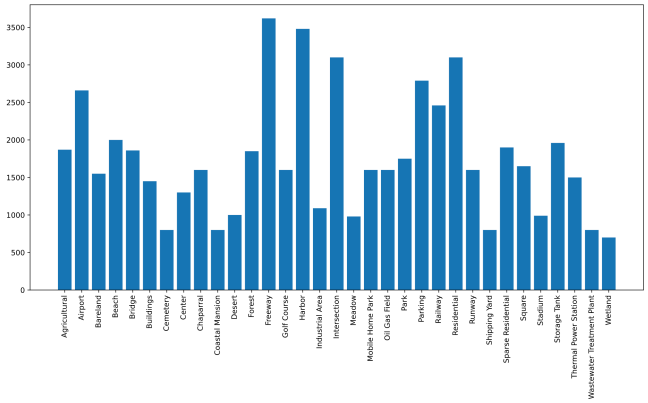


Fig. 2. Distribution of images in combined dataset

The dataset was split into 3 different categories Train, Validation, and Test in the ratio of 75%, 10% and 15% respectively, while maintaining the ratios between the classes. Example images of each class can be seen in the Appendix.

F. RSI-CB256

It consists of 24,747 images in total distributed in 35 different classes: *avenue, highway, bridge, parkinglot, marina, airport runway, crossroads, pipeline, town, airplane, forest, river protection forest, mangrove, shrubwood, artificial grassland, sapling, sparse forest, lakeshore, river, stream, coastline, hirst, dam, sea, sandbeach, snow mountain, mountain, desert, green farmland, bare land, dry farm, storage room, residents, city building, container.* With each class containing 223 - 1307 images. Each image is of the dimensions 256x256 pixels with a spatial resolution ranging from 0.22m - 3m[7].

The images were extracted from Open Street Map (OSM), Google Earth and Bing Maps of the following regions and countries: Beijing, Shanghai, Hong Kong, Guangzhou, Hainan, Fujian, and other provinces in China, New York, Washington, Los Angeles, Chicago, and other cities in the US; Tokyo, Osaka, Kobe and other cities in Japan; Paris, Nice, and other cities in France; Ottawa, Toronto, and other cities in Canada; and Moscow, St. Petersburg, and other cities in Russia[7].

G. WHU-RS19

It consists of 1,005 images in total distributed in 19 different classes: *Pond, Forest, River, Bridge, Park, Football Field, Farmland, Viaduct, Commercial, Industrial, Residential, Airport, Railway Station, Beach, Port, Mountain, Desert, Meadow, Parking.* With each class containing 50 - 60 images. Each image is of the dimensions 600x600 pixels with a spatial resolution upto 0.5m. The images were extracted from Google Earth. Images of different class were extracted from different regions[8].

H. External Dataset

A new dataset was created by combining RSI-CB256 and WHU-RS19 with the same class structure of the combined dataset. The purpose of this dataset is to test the model on data which is not similar to its training data. It will help in testing the viability of the model in a real world scenario.

Fig. 3 shows the distribution of images in the external dataset, there were some classes which had no images in this dataset. This dataset might not be the best indicator for viability of the model in a real world scenario, but can be a good starting point.

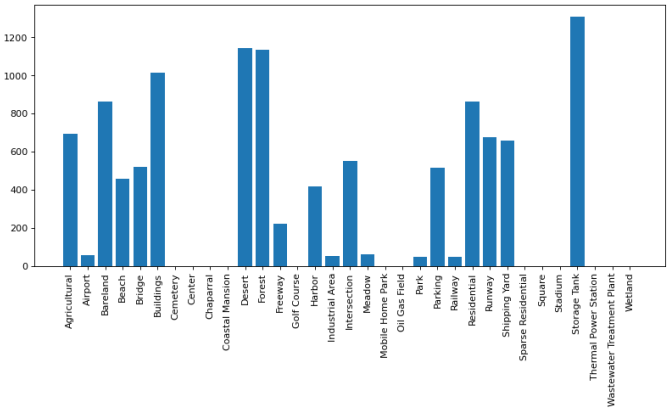


Fig. 3. Distribution of images in external dataset

III. Related Work

The analysis of the related work in the field of satellite image classification was carried to find suitable approaches to the problem, setting the estimate for the performance of the model, model architecture and etc.

A. Current Trends in Deep Learning for Earth Observation: An Open-source Benchmark Arena for Image Classification

This paper provides an analysis of 10 different state-of-the-art architecture and compares them to a variety of multi-class and multi-label classification tasks from 22 datasets. The study helped in deciding which CNN architecture to choose, whether the use of transfer learning helps and settings the base line for the model performs[9].

Fig. 4 (Figure 2 in paper) shows the comparison between training from scratch and using pre-trained models in multi-class and multi-label classification for 10 different CNN architecture. It is clear that using transfer learning has great benefits to the model's performance in both multi-class and multi-label classification.

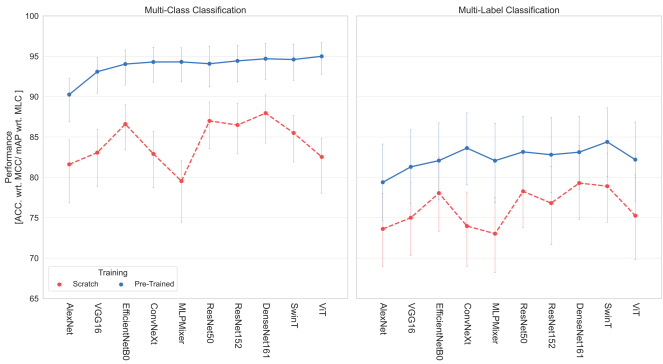


Fig. 4. Comparison of different CNN architecture in multi-class and multi-label classification when trained from scratch and employing pre-trained models

Dataset\Model	AlexNet	VGG16	ResNet50	ResNet152	DenseNet161	EfficientNetB0	ViT	MLPMixer	ConvNeXt	SwinT
WHU-RS19	93.532	99.005	99.502	98.01	**100**	99.502	99.502	98.507	99.005	99.502
Optimal31	80.914	88.71	92.204	92.473	94.355	91.667	**94.624**	92.742	93.011	92.473
UC Merced	92.143	95.476	98.571	**98.810**	98.333	98.571	98.333	98.333	97.857	98.571
SIRI-WHU	92.292	93.958	95	**96.25**	95.625	95	95.625	95.208	**96.25**	95.625
RSSCN7	91.964	93.929	95	95	94.821	95.536	**95.893**	95.179	94.643	95.179
BCS	89.583	90.972	92.014	92.361	92.708	91.319	92.014	93.056	91.493	**93.403**
AID	92.9	96.1	96.55	97.2	97.25	96.25	**97.750**	96.7	96.95	97.4
CLRS	84.1	89.9	91.567	91.9	92.2	90.5	**93.200**	90.1	91.1	92.533
RSI-CB256	99.354	99.051	99.677	**99.859**	99.737	99.717	99.758	99.657	99.596	99.677
Eurosat	97.574	98.148	98.833	**99**	98.889	98.907	98.722	98.741	98.778	98.944
PatternNet	99.161	99.424	**99.737**	99.49	**99.737**	99.539	99.655	99.704	99.671	99.688
RESISC45	90.492	93.905	96.46	96.54	96.508	94.873	**97.079**	95.952	96.27	96.587
RSD46-WHU	90.646	92.422	94.158	94.404	**94.507**	93.387	94.238	93.673	93.627	93.536
So2Sat	59.203	65.375	61.903	65.169	65.756	65.801	**68.551**	67.066	66.169	65.950
SAT6	99.98	99.993	**100**	**100**	**100**	99.988	99.998	99.995	99.999	99.999
Avg. Rank	9.93	8.67	4.67	3.80	3.13	5.87	**3.07**	5.33	5.47	3.20

Fig. 5. Accuracy of different pre-trained CNN architectures on multi-class classification

Fig. 5 (Table 6 in paper) shows the how different pre-trained CNN architecture performed on different dataset. This helped in the selection of CNN architecture and setting the baseline for the performance. 4 different CNN architectures were selected based on these results

- Vision Transformer (ViT): has the best performance
- DenseNet161: performed the best on PatternNet
- ResNet152: performed the best on UC Merced
- ResNet50: performed the best on PatternNet

Using the accuracy of the selected architecture on AID, PatternNet, RESISC45 and UC Merced a weighted average of the accuracy was calculated with the weighted being the ratio of theses dataset in the combined dataset. This will be used as expected performance from the model. Table II shows the weighted average accuracy of the selected architecture.

II. WEIGHTED AVERAGE OF ACCURACY

Architecture	Weight Average
Vision Transformer	98.228 %
DenseNet161	97.934 %
ResNet152	97.858 %
ResNet50	97.770 %

B. *DeepSat –A Learning framework for Satellite Imagery*

This paper discusses how current classification approaches are not suitable for the classification of satellite images due to the high variation in the data. The paper tries to solve this problem by creating a new classification framework. This framework extracts features from an input image, normalises them and feeds the normalised feature vectors to a Deep Belief Network (DBN) for classification. On SAT-4 dataset the DBN with feature extraction produced a classification accuracy of 97.95% which out performed state-of-the-art algorithms by approximately 11%. On SAT-6, it produces a classification accuracy of 93.9% and outperforms the other algorithms by approximately 15%. The SAT-4 and SAT-6 dataset contains 500,000 and 405,000 multispectral images of the dimensions 28x28 pixels respectively, distributed in 10 different classes. In comparison, the images of the combined dataset only contains Red, Green and Blue channel[10].

The key features that we use for classification are mean, standard deviation, variance, 2nd moment, direct cosine transforms, correlation, co-variance, autocorrelation, energy, entropy, homogeneity, contrast, maximum probability and sum of variance of the hue, saturation, intensity, NIR channels as well as those of the colour co-occurrence matrices, Enhanced Vegetation Index (EVI), Normalised Difference Vegetation Index (NDVI) and Atmospherically Resistant Vegetation Index.

IV. MODEL ARCHITECTURE

The model architecture was inspired from the DeepSat paper. Fig. 6 shows the model architecture. The input to the model is going to be an RGB image, which will then be transformed into a 224x224 pixel image and then into a tensor. This tensor will go to feature extractor and the normalisation block which transform the mean and standard deviation of each channel according to the CNN's requirement. The output of the feature extractor and the CNN are concatenated and sent to a multilayer perceptron which outputs the predictions of the models.

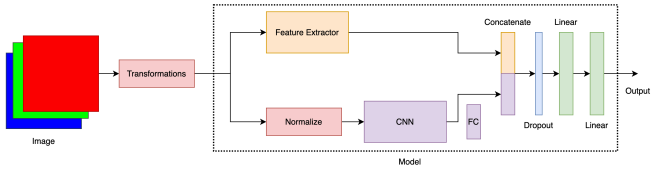


Fig. 6. Model Architecture

Table III, shows a summary of the CNN architecture used in the model.[12]

III. CNN ARCHITECTURE SUMMARY

Architecture	#Layers	#Parameters	Version
Vision Transformer[15]	12	~$86.5 \cdot 10^6$	b_16_224
DenseNet161[14]	161	~$26.4 \cdot 10^6$	
ResNet152[13]	152	~$58.1 \cdot 10^6$	
ResNet50[13]	50	~$23.5 \cdot 10^6$	

Due to technical limitations, features such as EVI and NDVI were not possible to extract as they required the use of the NIR channel. Table IV shows the list of all features that were extracted by the feature extractor. In total there 22 features that were extracted by the feature extractor.

IV. LIST OF EXTRACTED FEATURES

Channel Feature	Textural Features
Red: mean, variance, standard deviation	Energy
Green: mean, variance, standard deviation	Entropy
Blue: mean, variance, standard deviation	Contrast
Hue: mean, variance, standard deviation	Homogeneity
Saturation: mean, variance, standard deviation	
Intensity: mean, variance, standard deviation	

Fig. 6 shows how the feature extractor works, the input to the feature extractor is a image tensor. Which is then converted to a PIL image and sent to the PIL feature extractor. Which use the ImageStat module in the Pillow library to get the required channel features. The PIL image is also converted to a numpy array which is then used to calculate the energy, entropy, contrast and the homogeneity of the images. Both the channel feature and the textural feature are concatenated and normalised.

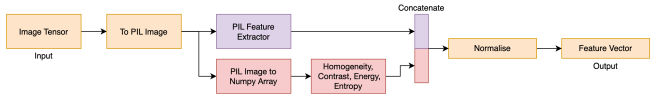


Fig. 6. Feature extractor

Equation (1) to (4) shows how the textural features are calculated, where N_g is the quantised grey levels and $p(i,j)$ be the (i,j)th entry in a normalised GLCM. The implementation for calculating the textural features was taken from "Measuring Texture and Color in Images"[11].

$$\text{Energy} = \sum_i \sum_j p(i,j)^2 \tag{1}$$

$$\text{Entropy} = -\sum_i \sum_j p(i,j)\, log(p(i,j)) \tag{2}$$

$$\text{Contrast} = \sum_{n=0}^{N_g-1} n^2 \left\{ \sum_{i=0}^{N_g-1} \sum_{j=0}^{N_g-1} p(i,j) \,\middle|\, |i-j| = n \right\} \tag{3}$$

$$\text{Homogeneity} = \sum_i \sum_j \frac{1}{1+(i-j)^2} p(i,j) \tag{4}$$

In order to get the min and max of each feature, the images of the training section of the combined dataset were resized to 224x224 pixel, and then the features were extracted and stored in an array. The min and max value were taken from the array of each feature. Table V shows the min and max value used for each feature.

V. FEATURE'S MIN AND MAX FOR NORMALISATION

Feature	Min	Max
Red Mean	0	256
Red Variance	6.789	9762.0.997
Red Standard Deviation	2.5844	99.8033
Green Mean	0	256
Green Variance	5.0467	8210.6875
Green Standard Deviation	2.2465	90.6128
Blue Mean	0	256
Blue Variance	5.8049	9199.0952
Blue Standard Deviation	2.4093	95.9119
Hue Mean	0	256
Hue Variance	0	15013.3399
Hue Standard Deviation	0	122.5289
Saturation Mean	0	256
Saturation Variance	0	107.0989
Saturation Standard Deviation	0	11470.1724
Intensity Mean	0	256
Intensity Variance	53.8551	9113.2301
Intensity Standard Deviation	7.3386	95.4623
Energy	0	0.4828
Entropy	1.6408	10.5852
Contrast	0	3438.2474
Homogeneity	0.0756	0.8689

Fig. 7 shows the final fully connected layer which was used in training. After the CNN's output and extracted features are concatenated they are sent to a dropout layer with a probability of 20%. This makes it hard for the model to overfit. The output of the dropout layer is sent to a linear layer with a ReLU activation function which reduced its size to 1024. The output of the first linear layer is sent to another linear layer with an sigmoid activation function which reduces its dimensions to 33.

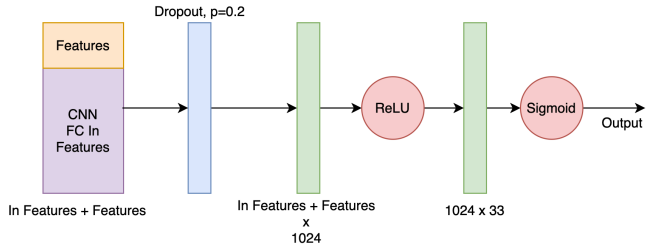


Fig. 7. Final fully connected layer

V. TRAINING PROCESS

The training process was taken from "Current Trends in Deep Learning for Earth Observation: An Open-source Benchmark Arena for Image Classification" paper, with a slight modification. In the paper, the full model was fine tuned during transfer learning, due to hardware limitations only the final fully connected layer will be fine tuned. Table VI shows the training parameters.

VI. TRAINING PARAMETERS

Parameter	Value
Batch Size	128
Max Epoch	100
Learning Rate	0.01, 0.001, 0.0001
Loss Function	Cross Entropy Loss
Optimiser	RAdam weight decay = 0
Schedular	ReduceLROnPlateau patience = 5 factor = 0.1
Early Stopping	Stop training if AbsMaxDifference(validationLoss[-10:]) < 0.00001

Fig. 8 shows the transformation applied to the input image before sending it to the model. These transformations were also taken form "Current Trends in Deep Learning for Earth Observation: An Open-source Benchmark Arena for Image Classification" paper. For the validation and testing test, resize dimensions were changed from 256x256 to 224x224 and the flips were removed.

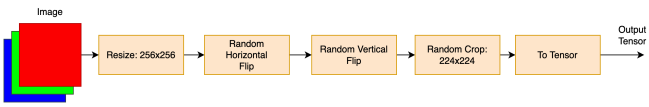


Fig. 8. Input transformations

In every epoch a checkpoint of the model state, optimiser state, scheduler state and loss function state were saved and learning rate, training loss, training accuracy, validation loss and validation accuracy were logged locally in a txt file. The model with the lowest validation loss was selected as the best model for a given configuration and used for comparison with other models of different configurations.

To ensure randomness doesn't influence the results, the seed was set to 42. Every model was trained on the same system. Table VII shows a summary of the system used.

VII. SYSTEM CONFIGURATION

Processor	11th Gen i9-11950H
RAM	32 GB
Operation System	Windows 11 Pro V23H2
GPU	NVIDIA RTX A4000
GPU Dedicated RAM	8 GB
Driver Version	537.99

VI. EVALUATION OF LEARNING RATE

In order to identify which learning rate produced the best results. ResNet50 architecture without any feature extractor was selected and trained on three different learning rate - 0.01, 0.001, 0.0001. ResNet50 was selected because it is the smallest CNN architecture out of all the selected CNN architecture.

Fig. 9 shows the comparison of training loss, training accuracy, validation loss and validation accuracy of the 3 learning rates. In all 4 metrics learning rate of 0.001 outperforms the other two learning rates, i.e having a lower training and validation loss and a higher training and validation accuracy.

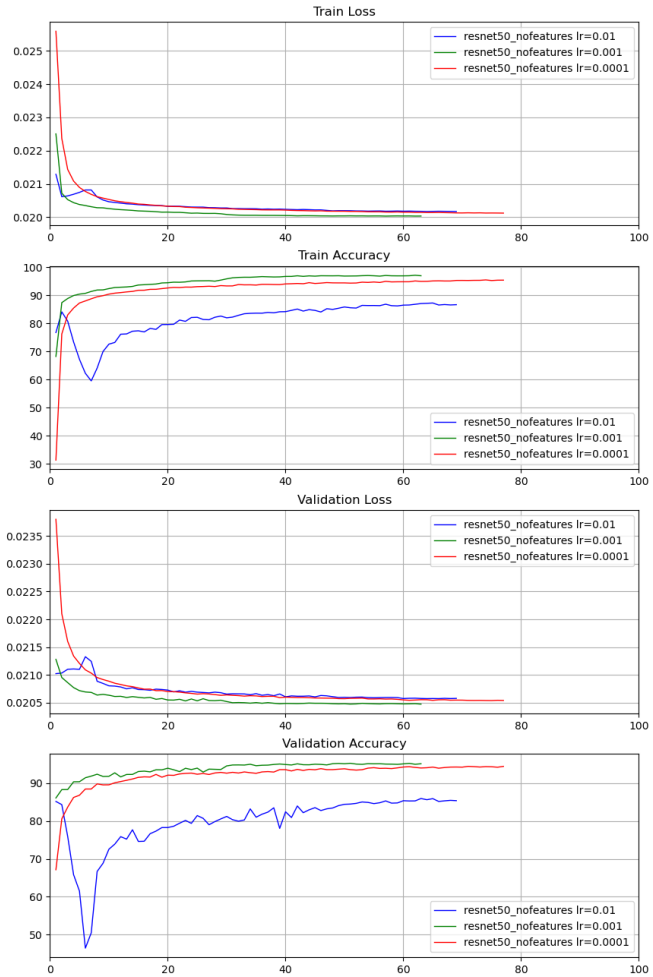


Fig. 9. Training comparison of different learning rate of ResNet50 with no feature extraction

Table VIII shows the summary of the results. In all metric learning rate of 0.001 has outperformed the other learning rate. In addition the learning rate of 0.001 converged the fastest.

VIII. RESULTS OF DIFFERENT LEARNING RATE FOR RESNET50

	0.01	0.001	0.0001
Accuracy	84.693%	**95.332%**	94.006%
Avg Precision	0.8732	**0.9560**	0.9462
Avg Recall	0.8491	**0.9534**	0.9401
Avg F1 Score	0.8493	**0.9543**	0.9426
Total Epochs (Best Epoch)	69 (66)	63 (**63**)	77 (75)

All the other variation and configuration of the model will be trained at a learning rate of 0.001.

VII. EVALUATION OF FEATURE EXTRACTOR

In order to identify which combinations of features help improve the model's performs, compared to no feature extraction. 3 different variation of the feature extractor were trained with the ResNet50 architecture at a learning rate of 0.001. These variations are channel features, textural features and both channel and textural features.

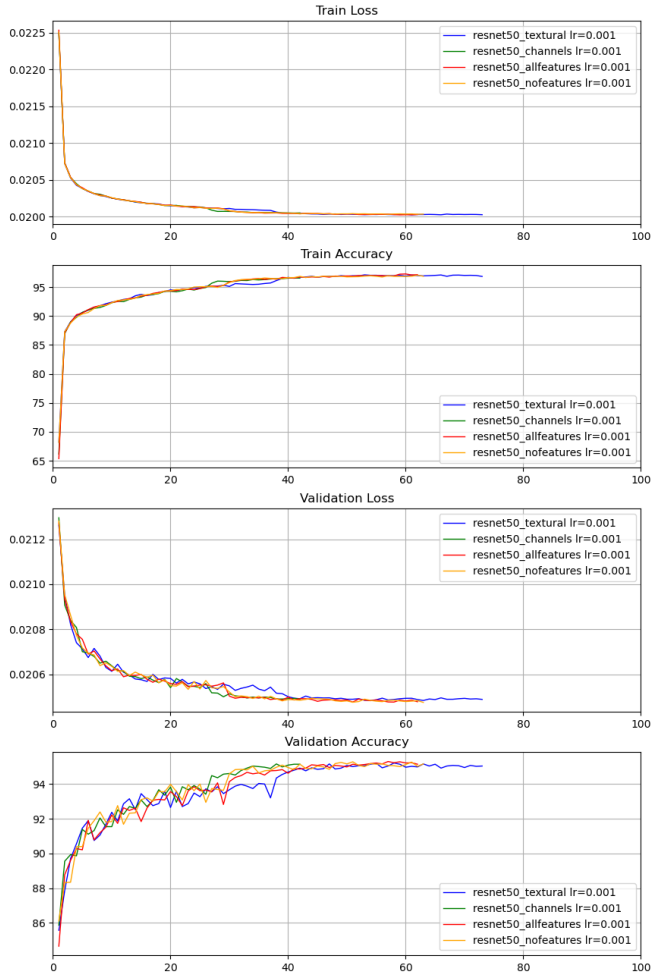


Fig. 10. Training comparison of different feature extractor of ResNet50

Fig. 10 shows the comparison of training loss, training accuracy, validation loss and validation accuracy of different feature extractor. It can be seen that all the metrics have an near identical shape.

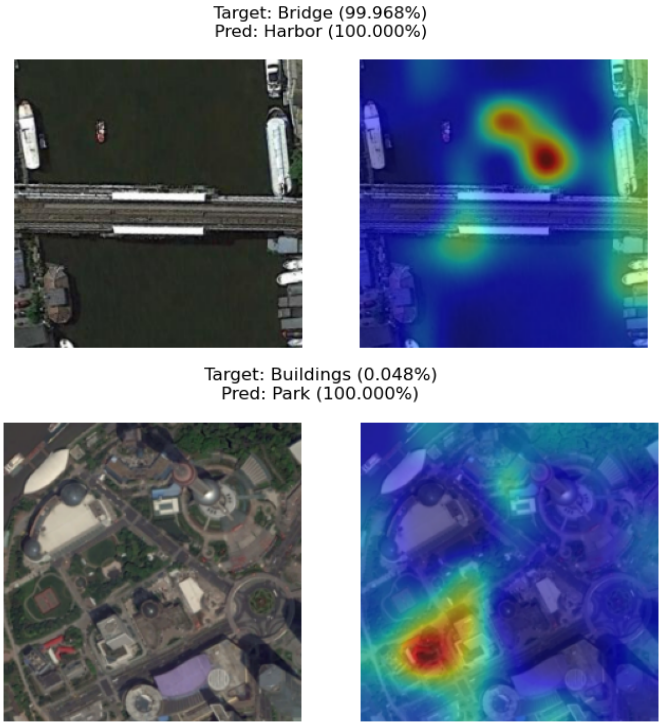
Table IX shows the summary of results. No variation of the feature extractor has significantly improved the model's performance compared to no feature extraction. Where as the DeepSat showed an approximately 10% improvement. This could be because of the additional features which were used, which were not possible due to the image containing only 3 channels.

IX. RESULTS OF DIFFERENT FEATURES FOR RESNET50

	No Features	Channel Features	Textural Features	All Features
Test Set Accuracy	95.332%	94.917%	94.836%	**95.378%**
External Set Accuracy	79.062%	**80.396%**	79.733%	80.078%
Avg Precision	0.9560	0.9517	0.9514	**0.9571**
Avg Recall	**0.9534**	0.9497	0.9494	0.9531
Avg F1 Score	0.9543	0.9503	0.9500	**0.9548**
Total Epochs (Best Epoch)	63 (63)	(38)	(54)	(58)
Duration / Epoch	6.5 min	7.8 min	41.4 min	46.7 min

Other possible reason why the feature extractor did not improve the model's performs could be, the CNN architecture used in the Deep Sat paper was very shallow compared to the ResNet50. In addition, the paper was written in September 2015, the author had no idea ResNet50 or any other architectures used in this paper. Therefore it was decided not to test the feature extractor with any other CNN architecture.

Fig 11 shows some example of misclassified images along with the activation map by the no feature extractor ResNet50 model. The activation maps were generated using the TorchCam[12] library.



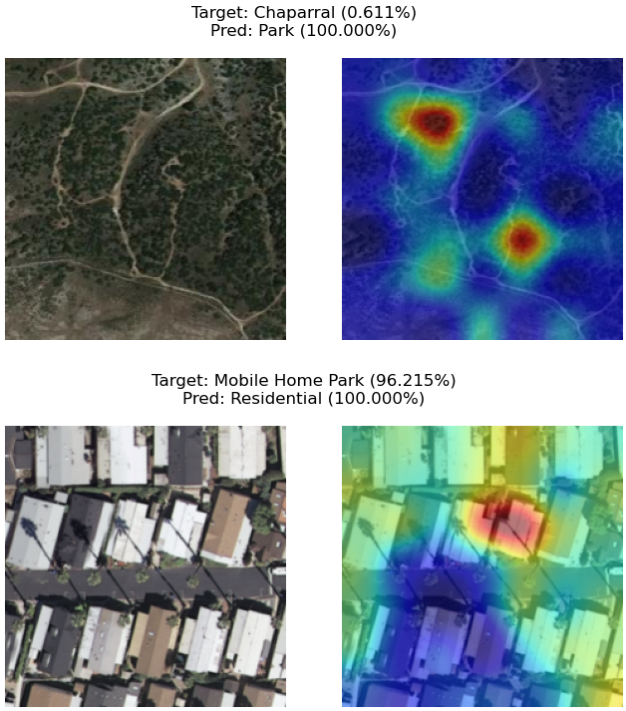


Fig. 11. Activation maps of misclassified images by ResNet50

It can be seen from Fig. 11 that some of the misclassified images have a very high prediction percentage for the target class. This could be because of multiple features being present in a single image. To counter this problem, the accuracy, average precision, average recall and average f1 score was calculated again, with a small change in how a classification is marked as correct. If the image is misclassified, then the target probability is compared with the prediction probability and if target probability is within 2% of the prediction probability then the classification is marked as correct. Table X shows the summary of results with this change.

X. RESULTS OF DIFFERENT FEATURES FOR RESNET50

	No Features	Channel Features	Textural Features	All Features
Test Set Accuracy	**97.556%**	97.291%	97.407%	97.418%
Avg Precision	**0.9762**	0.9734	0.9748	0.9761
Avg Recall	**0.9761**	0.9751	0.9751	0.9750
Avg F1 Score	**0.9760**	0.9741	0.9748	0.9754
% of Misclassified images in the 2% range	47.654%	46.710%	**49.777%**	44.140%

Since feature extraction does not provide any significant improvement to the model's performance in all the metrics, all other variation and configuration of the models will be trained without a feature extractor.

VIII. EVALUATION OF CNN ARCHITECTURES

In order to find out which CNN architecture helps in improving the model's performance compared to ResNet50. 3 different variations of CNN architecture - ResNet152, DenseNet162 and Vision Transformer were trained at a learning rate of 0.001 without any feature extractor.

Fig. 12 shows the comparison of training loss, training accuracy, validation loss and validation accuracy of different

CNN architectures. It can be seen that Vision Transformer outperforms all the other architecture

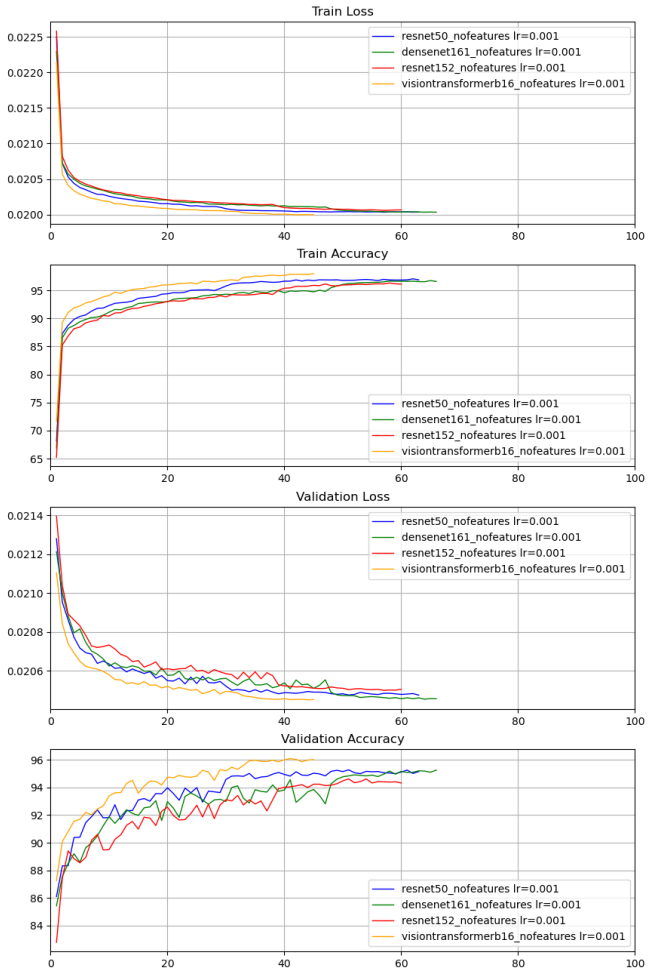


Fig. 12. Training comparison of different CNN architecture

Table XI shows the summary of results of different CNN architecture. Vision transformers has outperformed every other architecture in all the metrics. In addition, it was the fastest model to converge. DenseNet161 outperformed every other architecture when the metrics were calculated with the modified method. The difference between the expected accuracy and the test set accuracy could be because only the last fully connected layer was trained, where as, in the "Current Trends in Deep Learning for Earth Observation: An Open-source Benchmark Arena for Image Classification" paper - the full model was trained.

XI. RESULTS OF DIFFERENT CNN ARCHITECTURES

	ResNet50	ResNet152	DenseNet161	Vit
Expected Accuracy	97.770 %	97.858 %	97.934 %	98.228 %
Test Set Accuracy	95.332%	94.076%	95.021%	**95.989%**
External Set Accuracy	79.062%	78.267%	83.108%	**86.394%**
Avg Precision	0.9560	0.9442	0.9530	**0.9616**
Avg Recall	0.9534	0.9404	0.9509	**0.9600**
Avg F1 Score	0.9543	0.9420	0.9517	**0.9606**
Total Epochs (Best Epoch)	63 (63)	60 (57)	66 (64)	45 (44)
Duration / Epoch	6.5 min	24.5 min	8.7 min	17.1 min
2% Range				
Test Set Accuracy	97.556%	96.784%	**97.937%**	97.775%
Avg Precision	0.9762	0.9694	**0.9808**	0.9790
Avg Recall	0.9761	0.9678	**0.9796**	0.9774
Avg F1 Score	0.9760	0.9684	**0.9801**	0.9780
% of Misclassified images in the 2% range	47.654%	45.720	58.565%	44.540%

IX. FURTHER OPTIMISATION

A. *NLLLoss*

Since CrossEntropyLoss penalises on the non-target probabilities. As it was seen that the image can contain features of multiple classes. Therefore a loss function is required which only penalises on the target probability. Negative log likelihood loss is one such loss. The model was trained using NLLLoss with DenseNet161 and Vision Transformer architecture and keeping all the other training parameters the same.

Table XII shows the results of training with NLLLoss for both architecture. Both architecture performed better and had a lower percentage of misclassification of images within the 2% range when trained using NLLLoss. But with the new method of calculating the performance, architecture trained with CrossEntropyLoss performed better. In addition the models took longer to converge when trained with NLLLoss.

XII. COMPARISON OF RESULTS WITH NLLLOSS AND CROSSENTROPYLOSS

	DenseNet161 NLLLoss	DenseNet161 CELoss	ViT NLLLoss	ViT CELoss
Test Set Accuracy	96.727%	95.021%	**96.934%**	95.989%
External Set Accuracy	81.783%	83.108%	**87.366%**	86.394%
Avg Precision	0.9692	0.9530	**0.9707**	0.9616
Avg Recall	0.9686	0.9509	**0.9691**	0.9600
Avg F1 Score	0.9688	0.9517	**0.9700**	0.9606
Total Epochs (Best Epoch)	100 (81)	66 (64)	59 (48)	45 (44)
2% Range				
Test Set Accuracy	96.761%	**97.937%**	96.992%	97.775%
Avg Precision	0.9696	**0.9808**	0.9712	0.9790
Avg Recall	0.9689	**0.9796**	0.9700	0.9774
Avg F1 Score	0.9692	**0.9801**	0.9704	0.9780
% of Misclassified images in the 2% range	1.056%	58.565%	1.880%	44.540%

B. *Final Layer*

From the previous results, the best learning rate, CNN architecture and loss function were identified. To further improve the model's performance 2 more variations of the last fully connected layer were trained, shown in Fig. 13. The performance of these variation were compared to Vision transformer trained with a learning rate of 0.001, NLLLoss and the last fully connected shown in Fig. 7 with Sigmoid replaced with LogSoftmax.

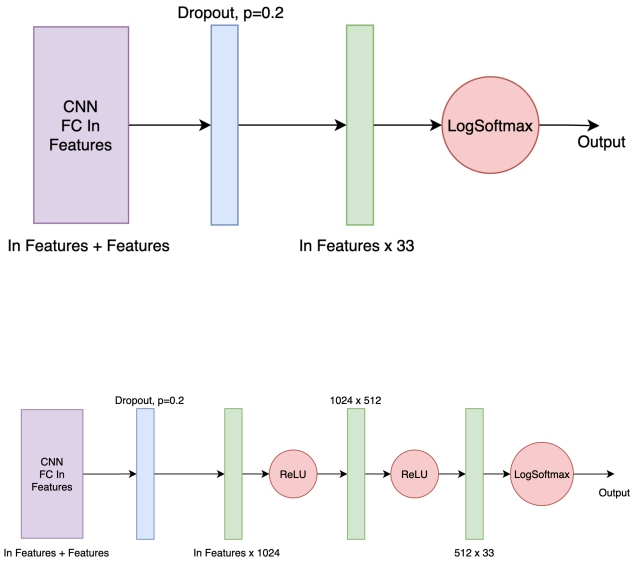


Fig. 13. Variations for the last fully connected layer

Table XIII shows the results of different last fully connected layer. The original last fully connected layer performed better in all performance metrics compared to the variations of last fully connected layer

XIII. RESULTS OF DIFFERENT LAST FULLY CONNECTED LAYER

	Original	Variation 1	Variation 2
Test Set Accuracy	**96.934%**	95.136%	96.427%
External Set Accuracy	87.366%	**87.587%**	84.955%
Avg Precision	**0.9707**	0.9558	0.9671
Avg Recall	**0.9691**	0.9529	0.9651
Avg F1 Score	**0.9700**	0.9542	0.9660
Total Epochs (Best Epoch)	59 (48)	68 (67)	46 (30)
2% Range			
Test Set Accuracy	**96.992%**	95.182%	96.450%
Avg Precision	**0.9712**	0.9562	0.9672
Avg Recall	**0.9700**	0.9533	0.9654
Avg F1 Score	**0.9704**	0.9546	0.9662
% of Misclassified images in the 2% range	1.880%	0.948%	0.645%

X. BEST MODEL RESULTS

The configuration of the model is Vision Transformer as the base CNN architecture and with no feature extractor and the original last fully connected layer shown in Fig. 7 with Sigmoid replaced with LogSoftmax. Trained at a learning rate of 0.001 with a batch size of 128 and a NLLLoss as the loss function. Table XIV shows the performance of this model using different measures.

XIV. PERFORMANCE OF MODEL

Metric	Value	Value in 2% Range
Test Set Accuracy	96.934%	96.992%
Avg Precision	0.9707	0.9712
Avg Recall	0.9691	0.9700
Avg F1 Score	0.9700	0.9704

Fig. 14 shows the confusion matrix. The 3 highest values in the matrix are 10, 9 and 7 of the target class Intersection, Spare Residential and Runway which were misclassified as Residential, Residential and Airport respectively. For a multi-class single label classification these values are in the acceptable range. Even for humans it would be hard to assign a single label to these images. Fig. 15 shows examples of some misclassified images. Additionally, it can also be seen that the model did not create a bias towards a single class even through the dataset was not distributed uniformly.

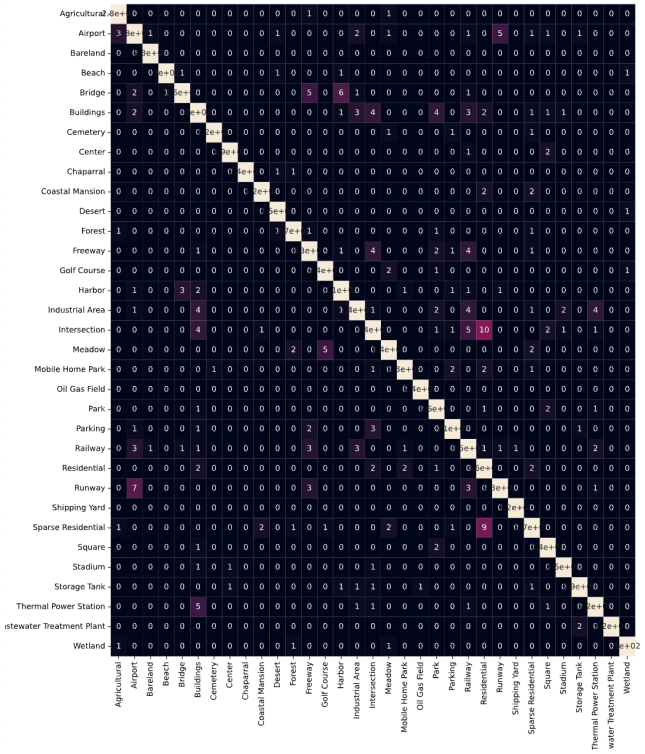


Fig. 14. Confusion matrix

Fig. 15. Example of misclassified images with target and predicted class probability

Table XV shows the precision, recall and f1 score of individual classes. The lowest precision, recall and f1 score of 0.8950, 0.8780 and 0.8070 was achieved by building, industrial area and building respectively. In addition the class with the lowest performance was also building. One of the reason for building performing the worst could be that building refers to a general building where as there are other classes present which describe a more specific type of building such as airport, residential. Furthermore, all of the classes has performed sufficiently well.

XV. PER CLASS PRECISION, RECALL AND F1 SCORE

Class	Precision	Recall	F1score
Agricultural	0,9789	0,9929	0,9859
Airport	0,9574	0,9574	0,9574
Bareland	0,9915	1,0000	0,9957
Beach	0,9966	0,9867	0,9916
Bridge	0,9813	0,9427	0,9616
Buildings	0,8950	0,8991	0,8970
Cemetery	0,9915	0,9750	0,9832
Center	0,9897	0,9846	0,9871
Chaparral	1,0000	0,9917	0,9958
Coastal Mansion	0,9748	0,9667	0,9707
Desert	0,9739	0,9933	0,9835
Forest	0,9820	0,9820	0,9820
Freeway	0,9724	0,9742	0,9733
Golf Course	0,9752	0,9833	0,9793
Harbor	0,9790	0,9808	0,9799
Industrial Area	0,9290	0,8780	0,9028
Intersection	0,9606	0,9441	0,9523

Meadow	0,9452	0,9388	0,9420
Mobile Home Park	0,9831	0,9708	0,9769
Oil Gas Field	0,9959	1,0000	0,9979
Park	0,9485	0,9810	0,9645
Parking	0,9833	0,9809	0,9821
Railway	0,9360	0,9512	0,9435
Residential	0,9441	0,9806	0,9620
Runway	0,9700	0,9417	0,9556
Shipping Yard	0,9917	1,0000	0,9959
Sparse Residential	0,9504	0,9404	0,9453
Square	0,9646	0,9879	0,9761
Stadium	0,9733	0,9799	0,9766
Storage Tank	0,9863	0,9796	0,9829
Thermal Power Station	0,9600	0,9600	0,9600
Wastewater Treatment Plant	1,0000	0,9833	0,9916
Wetland	0,9714	0,9714	0,9714

XI. DEPLOYMENT

The best performing model was deployed using AWS EC2, AWS Lambda and Firebase. EC2 is used to run the model, Lambda connects the EC2 to a https endpoint and Firebase is used to store images. The deployment can be viewed at https://www.aakashrathee.com/satelliteimageclassification.

The first step in deploying the model is to setup the EC2 instance. A new t2.small instance was created running Ubuntu with 20GB of storage. Once the instance was running, pip and Nginx were installed. Nginx will allow us to route a request on the public ip address of the instance to the localhost. To configure this behaviour, a new file was created in the *etc/nginx/sites-enabled* directory. Fig. 16 shows the contents of this file. This tells Nginx to listen on port 80 of the server's ip address and route the request to localhost port 5000. Port 80 uses HTTP.

```
server {
  listen 80;
  server_name 18.158.59.37;
  location / {
    proxy_pass http://127.0.0.1:5000;
  }
}
```

Fig. 16. Nginx configuration

The next step is to download the required packages using pip, these packages are flask, torch and torchvision. Once the packages are installed, the code was cloned from GitHub (https://github.com/aakashr143/SateImageEC2). The repository contains 4 python scripts: Models.py, app.py, download.py and lambda.py. The next step is to download the model's weights by running download.py. The scripts downloads the model.pth file from AWS S3 and saves it locally. Now all the required resources are downloaded in order to run the Flask app. To ensure the script runs even after the ssh session is closed, the app.py was run using the following command *nohup python3 app.py &*.

To check if everything is running as it was intended the public IPv4 address of the EC2 instance was opened in the browser and "s" was removed from https. In this case it was *http://18.158.59.37/*.

In order to access Flask app, the web app has to send a request to a HTTP URL, which is blocked by browsers as it is insecure. To overcome this problem a Lambda function was created with a function URL. The code for the Lambda function can be found in lambda.py.

The lambda function takes a query parameter of the location of the image the user wishes to classify and then it makes a POST request to the EC2 instance using the location of the image, and sends back the response of the EC2 instance back to the client.

Once the user has selected the image, the web app uploads it to Firebase storage and gets the downloadable URL of that image. The request is made to the Lambda function using the downloadable URL.

Fig. 17 shows the results of classification from the web app. The output of the model is passed through a softmax function. This image was taken from Google maps, north of Klagenfurt.

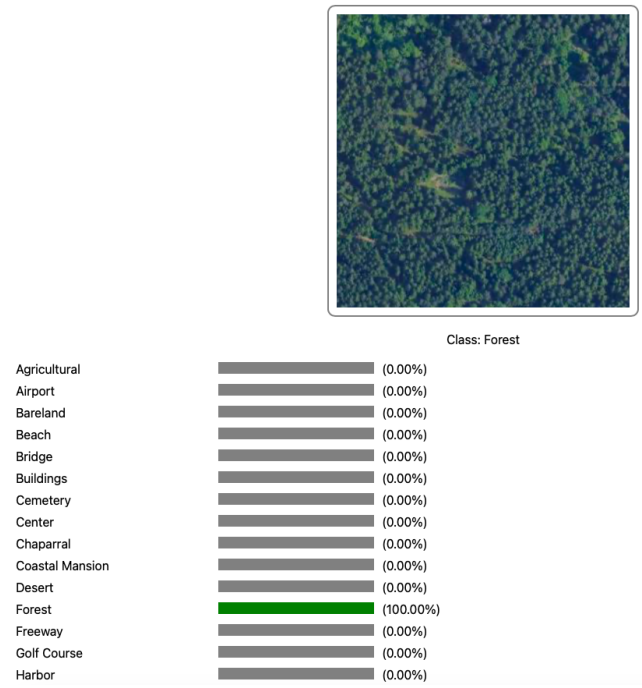


Fig. 17. Results of classification from the web app

XII. CONCLUSION

The aim was to develop an AI model which is able to identify the land use in satellite images. This is considered as an image classification task in computer vision and represented as a multi class single label task in supervised learning. The model has able to achieve an accuracy of 96.934%, with the Vision Transformer architecture. Which is close to the expected accuracy of 98.228%. The expected accuracy was not reached as only the last fully connected layer was trained. In addition, the viability of the model was tested in a real world scenario, the model was tested on a completely different dataset. The model achieved an accuracy of 87.366%, which shows that the model also useable in real world applications.

The approach of multi class single label classification is not ideal for a task like satellite image classification. As a single image can contain several features , multiple different classes makes it hard for the classifier to assign a single label. A much better approach would be to use multi label classification or semantic segmentation. The new method of calculating the performance metric of the model tried to simulate the behaviour of multi label classification, which showed an improvement in the model's performance. These results should not be interpreted as that the model performed much better with multi label classification when trained as single label classification. The improvement indicate that single label classification is not ideal for a dataset which has a high correlation between its classes.

The feature extractor did not provide any significant improvement to the model's performance. i.e the model performed better without one. In addition to that, there was a significant increase in the duration to train the model with the feature extractor present.

There are many possibilities for improving the model's performance. Such as additional data augmentation and different hyperparameter optimisations. Multi-spectral images could be used, which can provide additional information to the model. Additionally the images from the deployment are stored, they can also be added to the dataset.

XIII. Supplement Material

On Mac, if you are unable to uncompress then, change the extension from .zip to .rar. Only experience this with best models.

Demo: https://www.aakashrathee.com/satelliteimageclassification

Dataset(5.1GB): https://satellite-image-classification.s3.eu-central-1.amazonaws.com/LandUseImagesDataset.zip

Best model of each configuration(2.4GB): https://satellite-image-classification.s3.eu-central-1.amazonaws.com/Best+Models.zip

Code: https://github.com/aakashr143/SatelliteImageClassification/tree/main

Deployment Code: https://github.com/aakashr143/SateImageEC2

All Checkpoints and Logs

ResNet50 All Features(6.9GB): https://satellite-image-classification.s3.eu-central-1.amazonaws.com/resnet50_allfeatures.zip

ResNet50 Channel Features(4.7GB): https://satellite-image-classification.s3.eu-central-1.amazonaws.com/resnet50_channels.zip

ResNet50 Textural Features(8.1GB): https://satellite-image-classification.s3.eu-central-1.amazonaws.com/resnet50_eech.zip

ResNet50(23.1GB): https://satellite-image-classification.s3.eu-central-1.amazonaws.com/resnet50_nofeatures.zip

ResNet152(13.9GB): https://satellite-image-classification.s3.eu-central-1.amazonaws.com/resnet152_nofeatures.zip

DenseNet161(21.1GB): https://satellite-image-classification.s3.eu-central-1.amazonaws.com/densenet161_nofeatures.zip

VisionTransformer(66.9GB): https://satellite-image-classification.s3.eu-central-1.amazonaws.com/visiontransformerb16_nofeatures.zip

References

1. H. Ouchra and A. Belangour, "Satellite image classification methods and techniques: A survey," 2021 IEEE International Conference on Imaging Systems and Techniques (IST), Kaohsiung, Taiwan, 2021, pp. 1-6, doi: 10.1109/IST50367.2021.9651454. Available: https://ieeexplore.ieee.org/document/9651454
2. C. Kyrkou and T. Theocharides, "Deep-Learning-Based Aerial Image Classification for Emergency Response Application using Unmanned Aerial Vehicles" Available: https://openaccess.thecvf.com/content_CVPRW_2019/papers/UAVision/Kyrkou_Deep-Learning-Based_Aerial_Image_Classification_for_Emergency_Response_Applications_Using_Unmanned_CVPRW_2019_paper.pdf
3. Gui-Song Xia, Jingwen Hu, Fan Hu, Baoguang Shi, Xiang Bai, Yanfei Zhong, Liangpei Zhang "AID: A Benchmark Dataset for Performance Evaluation of Aerial Scene Classification", 2016. Available: https://arxiv.org/pdf/1608.05167v1.pdf
4. Zhou, W., Newsam, S., Li, C., & Shao, Z. (2018). "PatternNet: A benchmark dataset for performance evaluation of remote sensing image retrieval. ISPRS Journal of Photogrammetry and Remote Sensing", 145, 197-209.
5. Gong Cheng, Junwei Han, Xiaoqiang Lu. "Remote Sensing Image Scene Classification: Benchmark and State of the Art". Proceedings of the IEEE, 105(10): 1865-1883, 2017.
6. Yi Yang and Shawn Newsam, "Bag-Of-Visual-Words and Spatial Extensions for Land-Use Classification," ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS), 2010.
7. Li, Haifeng and Dou, Xin and Tao, Chao and Wu, Zhixiang and Chen, Jie and Peng, "RSI-CB: A Large-Scale Remote Sensing Image Classification Benchmark Using Crowdsourced Data". 2020. Available: doi.org/10.3390/s20061594
8. Gui-Song Xia, Wen Yang, Julie Delon, Yann Gousseau, Hong Sun and Henri Maître, "Structural high-resolution satellite image indexing". 2010
9. Ivica Dimitrovski, Ivan Kitanovskia, Dragi Koceva, Nikola Simidjievski, "Current Trends in Deep Learning for Earth Observation: An Open-source Benchmark Arena for Image Classification", 2022, Available: https://arxiv.org/pdf/2207.07189v2.pdf
10. Saikat Basu, Sangram Ganguly, Supratik Mukhopadhyay, Robert DiBiano, Manohar Karki and Ramakrishna Nemani, "DeepSat – A Learning framework for Satellite Imagery", 2015. Available: https://arxiv.org/pdf/1509.03602v1.pdf
11. Avinash Kak. "Measuring Texture and Colour in Images", 2022. Available: https://engineering.purdue.edu/kak/Tutorials/TextureAndColor.pdf
12. François-Guillaume Fernandez, "TorchCAM". Available: https://frgfm.github.io/torch-cam/#
13. K.He, X.Zhang, S.Ren, J.Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778
14. G.Huang, Z.Liu, L.Van Der Maaten, K.Q.Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708
15. A.Dosovitskiy, L.Beyer, A.Kolesnikov, D.Weissenborn, X.Zhai, T.Unterthiner, M.Dehghani, M.Minderer, G.Heigold, S.Gelly, etal., An image is worth 16x16 words: Transformers for image recognition at scale, arXiv preprint arXiv:2010.11929 (2020).

Appendix

Example of images of each class in the combined dataset

Sparse Residential

Parking

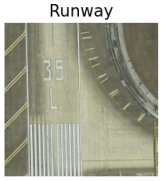
Intersection

Airport

Wastewater Treatment Plant

Shipping Yard

Runway



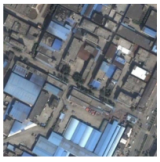
Agricultural

Harbor

Wetland

Forest

Industrial Area

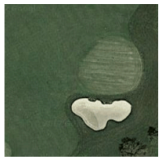


Buildings

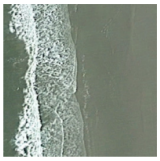
Storage Tank

Park

Golf Course



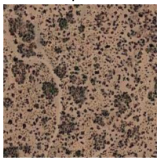
Beach



Freeway

Bareland

Chaparral



Stadium

Thermal Power Station

Cemetery

Residential

Mobile Home Park

Desert

Square

Center

Bridge

Railway

Coastal Mansion

Meadow

Oil Gas Field