Aerial Landscape Image Classification Using CNN and Transfer Learning

WANG SHUCHEN, GAYAKPA AFI SABINE ZITA, ZHOU XIUKANG, ZANNATUL SANZIDA, AHMED ADEL AL-HAIDARY, SHUBAASRI SACHIDANANTHAM, MUHAMMAD SYAFIQ SHAMSHUDIN, ABIDRAHMAN YUSUF

ABSTRACT

This paper investigates the classification of aerial images depicting transmission towers, forests, farmland, and mountains. The classification task is accomplished using a Convolutional Neural Network (CNN) architecture to extract features from input images, followed by Softmax for image classification. The dataset utilized for training and evaluation is a combination of the MLRNet dataset and self-collected images from Google satellite imagery.

Our approach achieves results, with an overall accuracy of 87% on the built CNN model. Furthermore, by utilizing the pre-trained VGG16 and MobileNetV2 models as a starting point for transfer learning, we achieve even higher accuracies. Specifically, VGG16 achieves an accuracy of 90%, while MobileNetV2 outperforms both models with an accuracy of 96%.

The results demonstrate the effectiveness of employing transfer learning with MobileNetV2 for classifying transmission towers, forests, farmland, and mountains.

Keywords: landscape type, Convolutional neural network, Transfer learning, VGG16, MobileNetV2.

1. INTRODUCTION

CNN and Transfer Learning technology has developed rapidly in recent years. Convolutional neural network is the most commonly used model in deep learning, which has strong self-learning ability, adaptability and generalization ability. Image satellite search has applications in military reconnaissance, cruising, map remote sensing, traffic control, agricultural farming, and other fields, and with the development of external neural network (CNN) in deep learning, it has been used in research on image classification problems.

In this paper, we propose a method for classifying aerial landscape images into four distinct categories: transmission tower, forest, farmland, and mountains. To achieve accurate and efficient classification, we employ a Convolutional Neural Network (CNN) architecture and leverage the power of transfer learning by utilizing pre-trained models, specifically VGG16 and MobileNetV2. Our CNN architecture consists of multiple convolutional layers, pooling operations, and fully connected layers with ReLU activation.

The objectives of this project are to develop a high-performance classification system capable of accurately identifying transmission towers, forests, farmland, and mountains. Additionally, we aim to explore the performance capabilities

of Convolutional Neural Networks (CNNs) for object detection in images and demonstrate the efficacy of machine learning approaches, particularly CNNs, in building landscape detection systems. Furthermore, we strive to showcase the improvement in detection quality achieved by employing deep convolutional neural networks.

2. RELATED WORK

Convolutional Neural Networks (CNNs) have recently seen developments in both quality and speed, which has spawned several inventive uses across a variety of fields, including satellite images [15]. One of the most notable instances of this is a recent study into the capacity of daylight satellite images to predict variables that were previously only gathered using on-the-ground surveys. The computer vision and remote sensing (RS) communities have been working together to overcome a few obstacles as they discover the benefits and limitations of satellite sources. Many different types of literature have offered suggestions for efficient technological methods to get around these disparities.

Road detection, baseline mining, visualizing road safety, and automated road fracture detection have all been studied specifically with regard to roads. The literature on remote sensing road detection dates to the 1980s and earlier, when manual digitization and coarse resolution imagery were used [16]. Road detection from remotely sensed images is challenging because, like other image analyses, it depends on a variety of factors, including sensor type, spectral and spatial resolution, weather, light change, and ground characteristics. In addition, it is challenging to model each of these circumstances and involve them into a single module because RS images of roads frequently include discontinuities, occlusion or shadows, nearly parallel boundaries with constant width, and sharp bends. A network of roads is too complex to be modelled using a general structural approach.

The majority of approaches for detecting roads that have been proposed in the literature use one or more of the following types of algorithms: classification-based (NNs and SVMs), knowledge-based, mathematical morphology and dynamic programming. Convolutional networks have recently started to be evaluated for their effectiveness in identifying and extracting road networks. In order to extract roads, Zhang et al. combined GF-2 and World View satellite photos as the input for a CNN, and they reached a 99.2% accuracy [17]. In a similar manner, Xu, Mu, Zhao, and Ma used low and high frequency sub-bands that reflect multiscale picture properties that were acquired by a contourlet transform, and they were successful in classifying scenes with an accuracy of more than 90% [18].

Moreover, by using four different forms of texture information supplied to satellite photos as input for a CNN, Xia, Cao, Wang, and Shang [19] were able to extract roads, forest, and field. They did this by combining conditional random field methods with CNN. This research showed how more texture and spectrum data from several sources can increase the precision of retrieving road, forest, fields information from RS photos.

Another study using CNNs to categorize road quality from satellite pictures was done in 2018 by Gabriel Cadamuro et al. [20]. Here, nearly 7,000 km of primarily trunk roads in Kenya were used to collect the International Roughness Index (IRI) using specialized equipment. The pre-trained networks AlexNet, VGG, and SqueezeNet were used to categorize road segments using the IRI data to label 50x50 cm resolution satellite images of the associated roads. To predict the quality of a given road segment, Cadamuro et al. suggested two solutions: (1) treating the problem sequentially by using the data from

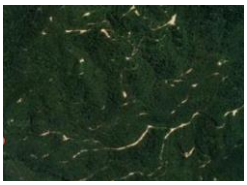
nearby road segments; and (2) better accommodating the continuous nature of road roughness measurements to reduce the detrimental effects of road heterogeneity upon the accuracy of predictions. This study adds to the expanding field of knowledge in a number of areas, including testing using better resolution images, an investigation of the efficiency of cross-continental transfer learning, and the use of a continuous road assessment.

3. PROPOSED SOLUTION

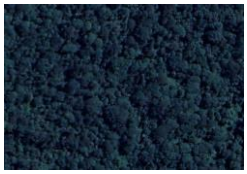
3.1 Dataset and Pre-Processing

Our model's dataset is a valuable combination of two different sources: MLRSNet and self-collected data using maps. This comprehensive dataset comprises a total of 10,400 images and encompasses four distinct landscapes: Transmission towers, Forests, Farmlands, and Mountains. These landscapes are classified into four classes, forming the basis of our classification task. By incorporating both MLRSNet data and our self-collected data, we aim to create a robust and diverse dataset that accurately represents the real-world scenarios our model will encounter. The MLRSNet data provides a foundational set of images with reliable labels, while our self-collected data, obtained through extensive mapping efforts, supplements the dataset with additional samples and enhances its diversity. To ensure effective model training and evaluation, we have divided our dataset into three subsets: training, validation, and test sets. The training set contains 70% of the images, while the validation and test sets contain 15% each. This split enables us to train our model on a significant portion of the data, fine-tune it using the validation set, and finally assess its performance on the independent test set. To facilitate consistent processing and analysis, we have standardized the dimensions of all the images in our dataset by reshaping them to a size of 224x224x3 to ensure that each image is represented by three color channels (red, green, and blue). The samples for each class in the dataset are shown:

Transmission tower



Mountains



Forest

Farmland

3.2 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a deep learning model specifically designed for processing structured grid-like data, such as images. It consists of multiple layers, including convolutional, pooling, and fully connected layers. CNNs leverage the concept of convolution, where small filters are applied to local regions of the input data to extract meaningful features. These features are then progressively learned and aggregated through pooling operations, reducing the spatial dimensions. The learned features are then flattened and fed into fully connected layers for classification or regression [7]. CNNs excel at capturing hierarchical patterns in images, as the convolutional layers learn to detect low-level features like edges and gradually learn higher-level features. With their ability to automatically learn relevant features, CNNs have achieved remarkable success in various computer vision

tasks, including image classification, object detection, and image segmentation.

3.3 Proposed Model for Classification

3.3.1 Architecture

Our proposed CNN architecture employs multiple convolutional layers with different filter sizes, stride values, and pooling operations to extract hierarchical features from input images (image size 224x224x3) and effectively classify images. The model begins with a convolutional layer that applies 64 filters with a size of 7x7. This layer utilizes a stride of 2 to down sample the feature maps and a same padding to maintain the spatial dimensions. The subsequent max pooling layer with a pool size of 3x3 and a stride of 2 further reduces the spatial dimensions while retaining essential information. After a second convolutional layer is introduced with 128 filters of size 3x3, utilizing the same padding to preserve spatial information. Another max pooling layer is then applied to down sample the feature maps. The subsequent convolutional layers, consisting of 256 filters of size 3x3, aim to capture and extract more complex features from the images.

After the fourth convolutional layer, another max pooling layer is employed, further reducing the spatial dimensions, and allowing the model to focus on the most important features. To increase the model's capacity, following feature maps are flattened to transform the multi-dimensional representations into a one-dimensional vector. This vector is then connected to two fully connected layers, each comprising 512 neurons with Rectified Linear Unit (ReLU) activation. The ReLU activation function helps introduce non-linearity, allowing the model to learn complex relationships between features.

To prevent overfitting and improve generalization, dropout layers are incorporated after each fully connected layer with a dropout rate of 0.5. These layers randomly deactivate a fraction of the neurons during training, reducing the model's reliance on specific neurons and enhancing its ability to generalize to unseen data.

Finally, the output layer consists of four neurons, corresponding to the number of classes in the classification task. The SoftMax activation function is applied to produce probability scores for each class, representing the likelihood of the input image belonging to a specific class.

This model offers a balance between complexity and model size. Its capability to learn complex features is enhanced by the inclusion of dropout layers after the fully connected layers, which helps reduce overfitting that may arise due to complexity. Dropout randomly deactivates neurons during training.

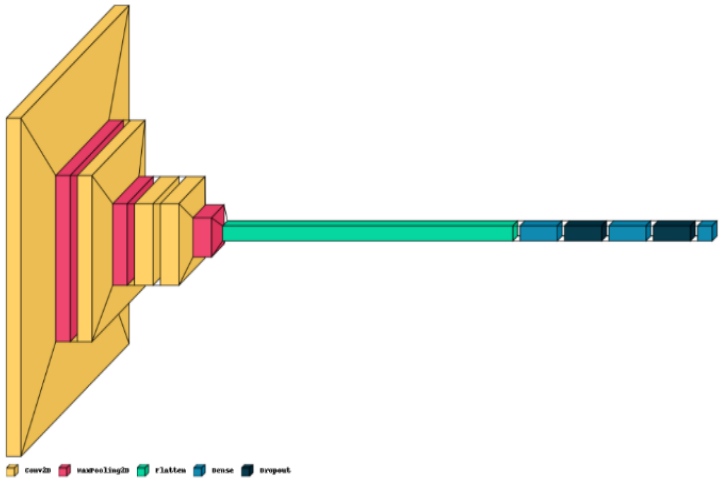


Figure 1 Convolutional neural networks architecture

Layer (type)	Output Shape	Param #
conv2d_44 (Conv2D)	(None, 112, 112, 64)	9472
max_pooling2d_27 (MaxPooling2D)	(None, 55, 55, 64)	0
conv2d_45 (Conv2D)	(None, 55, 55, 128)	73856
max_pooling2d_28 (MaxPooling2D)	(None, 27, 27, 128)	0
conv2d_46 (Conv2D)	(None, 27, 27, 256)	295168
conv2d_47 (Conv2D)	(None, 27, 27, 256)	590080
max_pooling2d_29 (MaxPooling2D)	(None, 13, 13, 256)	0
flatten_9 (Flatten)	(None, 43264)	0
dense_29 (Dense)	(None, 512)	22151680
dropout_18 (Dropout)	(None, 512)	0
dense_30 (Dense)	(None, 512)	262656
dropout_19 (Dropout)	(None, 512)	0
dense_31 (Dense)	(None, 4)	2052

Total params: 23,384,964
Trainable params: 23,384,964
Non-trainable params: 0

Figure 2 Convolutional neural networks diagram

3.3.2 Optimizer and Learning rate

In our model, we used the Adam optimizer with different parameters to train our network. The Adam optimizer is a popular choice for deep learning tasks due to its adaptive nature and efficient gradient descent algorithm [8]. It combines the advantages of two other optimization methods, namely AdaGrad [9] and RMSProp, to provide effective weight updates during training [10].

The learning rate parameter determines the step size at which the optimizer adjusts the model's weights during training. It plays a crucial role in controlling the convergence and performance of the model. By setting the learning rate to 0.001, we aim to strike a balance between stability and agility in the optimization process. A smaller learning rate can provide more stable convergence but might result in slower training, while a larger learning rate can lead to faster training but might risk overshooting the optimal solution.

During the training process, we employ a categorical cross-entropy loss function, which is well-suited for multi-class classification tasks. This loss function measures the dissimilarity between the predicted class probabilities and the true class labels. By minimizing the categorical cross-entropy loss, our model learns to assign higher probabilities to the correct classes and lower probabilities to the incorrect classes.

We train our model using a batch size of 90, which determines the number of samples processed before the optimizer performs a weight update. This batch-wise training allows for efficient memory utilization and computational speed. Furthermore, we train the model for 10 epochs, indicating the number of complete passes through the entire training dataset.

To assess the model's performance during training, we evaluate the accuracy metric, which measures the proportion of correctly classified samples. Additionally, we validate the model's performance on a separate validation dataset (x_val, y_val) to monitor any potential overfitting or generalization issues. We use those parameters to optimize the model's performance and achieve accurate classification.

3.4. Classification using transfer learning

In transfer learning, a pre-trained model is used as a base. However, instead of using the entire model as is, only the lower layers are typically frozen or kept fixed, while the upper layers are modified or replaced with new classifier layers. This approach of transfer learning allows us to benefit from the general knowledge and patterns learned by the base model, which can be applicable to various related tasks. By reusing and adapting the learned features, the model can achieve improved performance and efficiency in the new application, even with a smaller dataset. Here we are using vgg16 [11] and MobileNetV2 [12] transfer learning models

to compare the accuracy with our CNN model developed.

In our usage of the transfer learning model, we decided to freeze the layers and add a new classifier at the end. This decision was based on the computation needs required to train additional layers of the transfer learning model, as well as considerations related to our dataset size and the model's choices.

3.4.1 VGG16

The VGG16 architecture consists of 16 layers, including 13 convolutional layers and 3 fully connected layers. VGG16 uses small 3x3 filters throughout the network, which allows it to learn more detailed and localized features. The architecture also includes max pooling layers to reduce spatial dimensions and increase the model's ability to handle variations in object position.

The convolutional layer is stacked sequentially, with the number of filters gradually increasing from 64 in the first layer to 512 in subsequent layers. This stacking of convolutional layers enables VGG16 to learn complex hierarchical representations of input images.

After the convolutional layers, the feature maps are flattened and passed through three fully connected layers with 4096 neurons each. These layers act as the classifier and provide the final predictions for the image classes. The rectified linear unit (ReLU) activation function is used throughout the network to introduce non-linearity.

VGG16 is well-known for its simplicity and effectiveness, making it a popular choice for many computer vision applications. However, due to the large number of parameters, training VGG16 can be computationally expensive and memory intensive. As a result, it is often used as a feature extractor or fine-tuned for specific tasks using transfer learning.

3.4.2 MobileNetV2

MobileNetV2 is a convolutional neural network (CNN) architecture designed specifically for mobile and resource-constrained devices. It is an evolution of the original MobileNet architecture, developed by Google. MobileNetV2 aims to achieve high performance while maintaining efficiency and low computational cost.

The key feature of MobileNetV2 is the use of depth wise separable convolutions. Inverted residuals use a combination of 1x1 and 3x3 convolutions to efficiently capture both low-level and high-level features. Linear bottlenecks help preserve the information flow and reduce the model's memory footprint.

An important aspect of MobileNetV2 is the use of a technique called width multiplier. This parameter controls the number of filters in each layer, allowing the model's width to be adjusted based on the available resources. By varying the width multiplier, MobileNetV2 can trade-off between model size and performance, making it adaptable to different devices and applications.

4. RESULTS AND DISCUSSIONS

We conducted our experiment using our landscape dataset. Initially, we divided the dataset into training (70%), validation (15%), and testing (15%) subsets. The training and validation are done in parallel. During training, we observed the effect of different parameters and tuned those parameters to obtain an accurate model. Python library “Keras” is used to implement this model using Jupiter Notebook on a laptop with AMD RADEON RYZEN 5, 16GB RAM, and 400GB of disk space.

4.1. Effects of Hyper-Parameters

4.1.1 Effect of Batch Size

The batch size is a hyperparameter that determines the number of samples processed by the model in each training iteration. It impacts both the training time and the model's ability to generalize to unseen data [13]. Training with larger batches generally results in faster training times since the model processes more samples in each iteration. However, this may require more memory to store the activations and gradients, which can be a limitation for resource-constrained systems [14].

To show the effect of the batch size on our model we chose different values of batch size 90, 50, 15 conserving the same learning rate and same epoch on first model that we built from scratch. The results are shown in the table below:

Table 1 : Batch effect on the model accuracy

Batch size	Final Training accuracy	Final Validation accuracy
90	0.966187345	0.84108628749
50	0.911654365	0.82792333841
15	0.916187345	0.82108628749

We can observe that modifying the batch size affects the accuracy of the model, depending on the dataset and the specific model used. In our case, a batch size of 90 resulted in the highest accuracy.

4.1.2 Effect of Number of epochs

Epochs refer to the number of times the entire dataset is passed forward and backward through a machine learning model during the training process. Each epoch consists of multiple iterations, with each iteration processing a batch of data. The number of epochs is a hyperparameter that determines how many times the model will update its weights based on the training data [13]. Training for more epochs allows the model to potentially learn more complex patterns and improve its performance, but there is a risk of overfitting if the model becomes too specialized to the training data. Conversely, training for too few epochs may result in underfitting, where the model fails to capture important patterns in the data. The optimal number of epochs depends on the dataset size, model complexity, and available computational resources [14]. We used three different epoch value (10, 4, 2) conserving the same batch size and same learning rate on the transfer learning model VGG16 to show his effects. The results are shown in the table below:

Table 2 : epochs number effect on the model accuracy

epoch number	Final Training accuracy	Final Validation accuracy
10	0.91399891	0.876038312
4	0.90043549	0.892012774
2	0.90851819	0.882428109

Changing the number of epochs resulted in varying levels of accuracy. In this particular case, the model with an epoch size of 10 achieved the highest accuracy.

4.1.3 Effect of learning rates

The learning rate is a hyperparameter that determines the step size at which the model's parameters are updated during training. It plays a crucial role in the training process and can significantly impact the model's convergence and performance. A higher learning rate allows for larger parameter updates, which can lead to faster convergence but risks overshooting the optimal solution. Conversely, a lower learning rate results in smaller updates, which can lead to slower convergence but may help the model to

converge to a more precise solution. Selecting an appropriate learning rate involves finding a balance between convergence speed and accuracy. To display the effect of learning rate we have selected 3 different values of learning 0.01,0.001,0.0001 and we applied it one our last model MobileNetV2 conserving the same batch size and same epoch. The results are shown in the below table:

Table 3 : learning rate effect on the model accuracy

Learning rate	Final Training accuracy	Final Validation accuracy
0.01	0.977814316749	0.95143771
0.001	0.990413606166	0.95654952
0.0001	0.995343744754	0.95591056

According to Table 3, the best accuracy for this model was achieved by applying a learning rate of 0.0001.

4.2 Models Validation Accuracy and Loss Comparison

Using the fixed value of 90 batch size , 10 epochs and learning rate of 0.001 we can compare the tree models validation accuracy and loss.

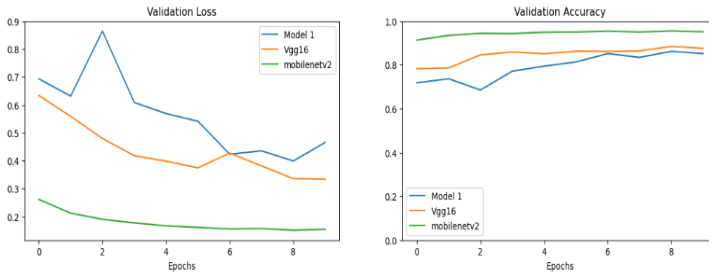


Figure 3 : Validation accuracy and loss comparison

In figure 3, we can see that our proposed model performs less effectively compared to the two-transfer learning models. Among the models evaluated, MobileNetV2 exhibited the highest performance.

4.3. Comparison of Classification Accuracies Between the Proposed and Transfer Learning Models

For the testing step, we utilized a batch size of 90, ran the model for 10 epochs, used the Adam optimizer, and set the learning rate to 0.001. Referring to Table 4, we can observe that the proposed model achieved a test accuracy of 87.2% and a test loss of 0.364. Although the accuracy is satisfactory, there is room for improvement, particularly when comparing it to MobileNetV2 and VGG16.

MobileNetv2 achieved a significantly higher test accuracy of 96.2% and a lower test loss of 0.119. This indicates that MobileNetv2 performs better than the proposed model in terms of accuracy and generalization. With its efficient architecture, MobileNetv2 is suitable for our specific task.

VGG16 achieved a test accuracy of 90.6% and a test loss of 0.298. While VGG16 performs reasonably well, it falls short in terms of accuracy compared to MobileNetV2. However, it is worth considering that the VGG16 architecture may be too complex for our specific task.

Overall, MobileNetV2 outperforms both the proposed model and VGG16 in terms of test accuracy and loss. It strikes a good balance between accuracy and efficiency, making it a suitable choice. The proposed model demonstrates reasonable performance but has room for improvement. While VGG16 may not be as accurate as MobileNetV2, it can still be

valuable in scenarios where more complex feature extraction is required.

Table 4 : comparison of different models' accuracy

Model	Test accuracy	Test loss
Proposed model	0.872204482	0.363988041
MobileNetv2	0.961661338	0.119421191
Vgg16	0.906070291	0.297783434

CONCLUSION

In conclusion, this paper focuses on the usage of deep learning techniques to classify aerial landscape images by combining MLRSNet data with self-collected data. Our proposed Convolutional Neural Network (CNN) architecture, along with transfer learning models VGG16 and MobileNetV2, was evaluated for landscape classification. Transfer learning involves utilizing a pre-trained model and adapting it for a specific task by modifying or replacing the upper layers. VGG16, known for its simplicity and effectiveness, performed reasonably well, while MobileNetV2, designed for efficiency on resource-constrained devices, outperformed both the proposed model and VGG16 in terms of accuracy and loss.

Overall, our findings highlight the effectiveness of transfer learning models, particularly MobileNetV2, for landscape classification. These models provide a balance between accuracy and efficiency, making them valuable choices for real-world applications. The proposed model demonstrates reasonable performance and can be further optimized to enhance its accuracy. Further research and experimentation can focus on improving the proposed model or exploring other transfer learning architectures tailored to specific landscape classification tasks.

REFERENCE

[1] Xu Xiaoping, Yu Xiangjia, Liu Guangjun, et al. Graphite Classification Using Transfer Learning and Focal Loss Convolutional Neural Networks. Application of Computer Systems, 2022, 31(3):248–254. [doi: 10.15888/j.cnki.csa.008378]

[2] Zhang Ke, Feng Xiaohan, Guo Yurong, et al. A review of deep convolutional neural network models for image classification. Chinese Journal of Image and Graphics, 2021, 26(10): 2305–2325.[doi: 10.11834/jig.200302]

[3] Zhang ZH, Wang YY, Lu T, et al. Enhanced lightweight neural networks for plant disease classification. Proceedings of 2021 4th International Conference on Robotics, Control and Automation Engineering (RCAE). Wuhan: IEEE, 2021.55–59. [doi: 10.1109/RCAE53607.2021.9638870]

[4]Chen Zhengtao, Huang Can, Yang Bo, et al., Parallel Convolutional Neural Network Yak Face Recognition Algorithm Based on Transfer Learning (1/OL7. Computer Application: 1-7.12020-10-221

[5] Sandler M, Howard A, Zhu ML, et al. MobileNetv2: Inverted residuals and linear bottlenecks. Proceedings of 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Salt Lake City: IEEE, 2018. 4510 – 4520. [doi:10.1109/CVPR.2018.00474]

[6] Howard AG, Zhu ML, Chen B, et al. MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861, 2017.

[7] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444. doi:10.1038/nature14539

[8] Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In

International Conference on Learning Representations (ICLR).

[9] Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research, 12(7), 2121-2159.

[10] Tieleman, T., & Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 4(2), 26-31.

[11] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[12] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4510-4520.

[13] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

[14] Bengio, Y., Léonard, N., & Courville, A. (2012). Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1305.2982.

[15] Song J, Gao S, Zhu Y, Ma C. A survey of remote sensing image classification based on CNNs. Big Earth Data. 2019;3(3):232–254.

[16] Burke M, Driscoll A, Lobell DB, Ermon S. Using satellite imagery to understand and promote sustainable development. Science. 2021;371(6535): eabe8628. pmid:33737462

[17] Suhui X, Xiaodong M, Peng Z, Ji M. Scene classification of remote sensing image based on multi-scale feature and deep neural network. Acta Geodaetica et Cartographica Sinica. 2016;45(7):834.

[18] Zhang Y, Xia W, Zhang YZ, Sun SK, Sang LZ. Road extraction from multi-source high-resolution remote sensing image using convolutional neural network. In: 2018 International Conference on Audio, Language and Image Processing (ICALIP). IEEE; 2018. p. 201–204.

[19] Cadamuro G, Muhebwa A, Taneja J. Assigning a Grade: Accurate Measurement of Road Quality Using Satellite Imagery. CoRR. 2018;abs/1812.01699

[20] Xia M, Cao G, Wang G, Shang Y. Remote sensing image classification based on deep learning and conditional random fields. Journal of Image and Graphics. 2017;22(9):1289–1301.

APPENDIX

Link to python code source file :

https://drive.google.com/file/d/1QsMqjIrshJYG3QKuPMi4IxzJNXMnfpd6/view?usp=sharing

GROUP 3 TASK DISTRIBUTION

Task	Name	Matric Number	E-mail
Title & Abstract	ZHOU XIUKANG	P-COM0028/22	zhouxiukang@student.usm.my
Introduction	WANG SHUCHEN	P-COM0104/22	wangshuchen@student.usm.my
Related work	SHUBAASRI SACHIDANANTHAM	P-COM0140/22	shubaasri@student.usm.my
Material, method, Results and discussion	GAYAKPA ZITA AHMED AL-HAIDARY Abidrahman Yusuf Muhammad Syafiq bin Shamshudin	P-COM0064/22 P-COM0113/22 P-COM0062/22 P-COM0057/23	gayakpazita@student.usm.my ahmedadel@student.usm.my abdirahman1998@student.sum.my syafiqshamshudin@student.usm.my
Conclusion	Zannatul Sanzida	P-COM0145/22	zannatul.sanzida@student.usm.my