

Key Pointers for Developing Pre-Trained Convolutional Neural Networks for Remote Sensing Image Classification

Nisha Gupta¹, Ajay Mittal², and Satvir Singh³

Submitted: 05/01/2024

Revised: 20/02/2024

Accepted: 12/03/2024

Abstract: The process of correctly identifying objects in an image is known as image classification. Effectively classifying high-resolution spatial images for huge remote sensing archives is known as remote sensing image classification. Better classification performance is directly correlated with effective feature extraction from images. The majority of feature extraction steps employed manually created low-level features that concentrated on basic components like color, form, and texture before deep learning was widely used in remote sensing image classification. However, because of their poor performance, these conventional handmade methods were swiftly superseded by Convolution Neural Networks (CNN's), that successfully recovered abstract information. However, while creating deep Convolution Neural Networks, it is important to carefully consider the significant training constraints of CNNs. This research aims to investigate the primary training challenges encountered while training deep learning pre-trained models utilizing a transfer learning approach.

Keywords: Image Classification, Convolution Neural Networks, Pre-Trained Models

1 Introduction

The extraction of salient features is crucial for image categorization and retrieval (Gu, 2019). Sharif et al. (2019) use machine learning and deep learning to extract crucial information from photos. The literature consistently demonstrates that deep learning methods outperform in both image classification and retrieval. Unlike machine learning, deep learning automatically performs feature extraction and classification in a single

phase, as illustrated in figure 1.

In multiclass classification, if a dataset has C classes, the query picture sample corresponds to one of them. CNN's output neurons are equal to the dataset's total number of classes, C . These output neurons generate vectors. The target vector (t) is a single ground truth vector with one positive and $C-1$ negative classes [1].

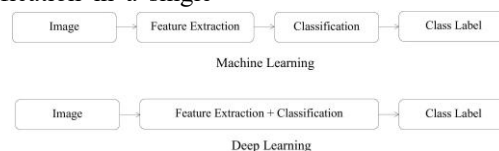


Figure 1: Feature Extraction and Classification approach Machine Learning/Deep Learning

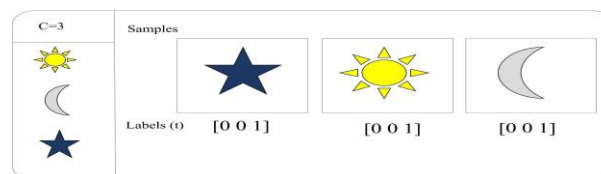


Figure 2: Multiclass Classification

1Department of Computational Sciences, MRSPTU Bathinda, India.

* nishasbs2019@gmail.com

2Department of Applied Sciences, Aryabhata Group of Institutes, Barnala, India.

* mittalajay@gmail.com

3Department of Electronics and Communication, IKGPTU Kapurthala, India.

* drsatvir.in@gmail.com

*Address correspondence to: nishasbs2019@gmail.com

DeepCNN (Deep Convolutional Neural Network) is a multilayer neural architecture used in deep learning to handle massive volumes of data. Deep neural networks use neurons and convolution filters to classify objects, comparable to the human brain [2]. Traditional machine

learning approaches used filters such edge and texture histograms [3]. As seen in figure3, CNNs use a

combination of layers to convert an image into output that the model may use for picture categorization.

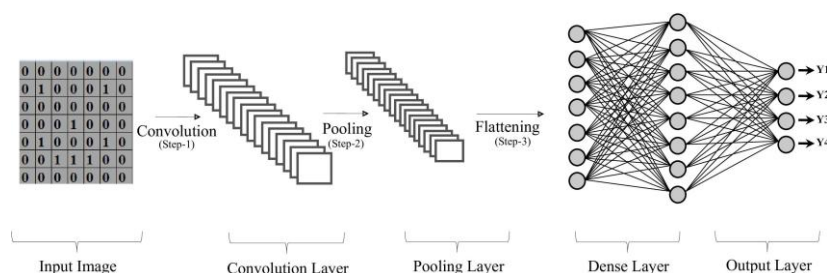


Figure 3: Convolution Neural Networks

The convolutional layer builds the feature map by repeatedly scanning the full picture's pixels with filters. CNN uses learnable filters [4]. CNN operates in two steps. The initial level is learning, which involves a convolution and pooling layer. This convolution layer is where learnable filters and feature extractors go to work. The pooling layer scales down the information produced by the convolutional layer. The pooling layer compresses data. A compact vector generated by the pooling layer is stored. The completely connected input layer reduces the outputs to a single vector [5]. The fully linked layer assigns weights to the inputs generated by the Dense Layer. The completely linked output layer provides the final probability for picture classification. Classification occurs in the final fully linked output layer [6]. CNNs are good at training with small quantities of data, but they suffer from overfitting or underfitting as the data size increases. This results in poor classification results when training with large-scale datasets [7]. This paper addresses key challenges encountered while training deep CNNs and provides confined strategies to address these training issues [8].

2 Related Literature Survey

The researchers used a variety of machine learning and deep learning methods to extract and classify photos. [9] addressed the need for large labelled samples for training CNNs. The author proposed the SBS-CNN (Similarity Based Supervised Learning Using Convolution Neural Network) method, which is based on similarity learning and applies transfer learning to CNN training, transforming similarity learning into deep ordinal classification with CNN experts pre-trained on large-scale labelled everyday image

sets. It computes picture similarity and applies pseudo labels for categorisation. SBS-CNN features a small network size, compact feature vectors, and faster retrieval times. Gradient descent is employed at each level to minimise errors. Major research problems include calculating gradient descent at each level to reduce error. Negative values are fed into the ReLU (Rectified Linear Unit) activation function, which turns them to zero, decreasing the model's ability to fit or train from data properly. The author tested the Everyday ImageSet, UC-Merced, and PatterNet datasets, and the best ANMR obtained was 0.2185 on the benchmark dataset. Wang (2020) presented OSA-HSR + CNN (Object Scale Adaptive High Spatial Resolution Remote Sensing Image Classification with Convolution Neural Networks).

There are two phases involved: segmentation and classification. Segmentation creates heterogeneous segments that are then utilised to extract features. Classification is performed using the obtained features. The key research issues are the growing segmentation scale of adjacent items and the decrease in classification quality. Over segmentation and under segmentation may lead to misclassification. OSA-HSR+CNN increases the additional run time. The stochastic gradient is evaluated at each step to reduce errors, which increases processing time.

Experiments are carried out using aerial images obtained by the Ohio Statewide Imagery Program. The total computational time is 188 seconds. [10] proposed EfficientNet-B3-Attn-2, which uses a pre-trained EfficientNet-B3 CNN and an attention mechanism. Experiments are carried out with a CNN feature extractor

that was constructed from scratch, transferred, trained, and fine-tuned for the LCLU (Land Cover and Landuse) classification system utilising remote sensed images.

The fine-tuned deep learning model fared better than the UCM dataset in terms of accuracy. Experimental results on six popular remote sensing datasets, namely KSA, UC Merced, WHU-RS19, RSSCN7, OPTIMAL-31, and AID, show that the proposed network performs accurately and efficiently, reducing computation time and improving accuracy in hyperspectral image classification. Huang, Gao, Liu, Zhuang, Van Der Maaten, Laurens, and Weinberger, Kilian Q (2017) [11] developed the DenseNet (Dense Convolutional Network), which uses a feed-forward approach to connect one layer to the next.

L layers are connected one after the other in traditional convolutional networks. A network's direct links are $(L(L+1))/2$. The feature-maps of each layer were utilised as input for following layers, while the feature-maps of previous layers were used as input for this layer. DenseNets resolves the issue of fading gradients. The tests are carried out using four benchmark datasets: CIFAR-100 (Canadian Institute for Advanced Research), SVHN, CIFAR- 10, and Imagenet. The author achieved an

accuracy of 97.44% on AID, 99.50% on UC-Merced, 95.89% on Optimal, and 94.98% on NWPU-RESIS45 datasets. Zhang Jianming, Lu Chaoquan, Li Xudong, Kim Hye-Jin, and Wang Jin (2019) [12] discovered two prevalent issues in Convolutional Neural Networks.

The first is that these models generate overfitting because they have too many parameters, and the second is because they are insufficiently deep to retrieve abstract data. To address these two issues, the author proposed a pre-trained DenseNet model for remote sensing image categorisation. DenseNet produces several reusable features using fewer convolutional kernels. Dense connections take the network to over 100 levels.

Data augmentation is used. Experiments are conducted on the AID (Aerial Image Dataset), UCM, NWPU-RESISC45, and Optimal-31 datasets. The author achieved accuracy of 98.67% (50% training ratio), 99.50% (80% training ratio) on the UCM dataset, 95.37% (20% training ratio), 97.19% (50% training ratio) on the AID dataset, 95.41% (80% training ratio) on optimal-31, and 92.90 (10% training ratio), 94.95 (20% training ratio) on the NWPU-RESISC45 dataset.

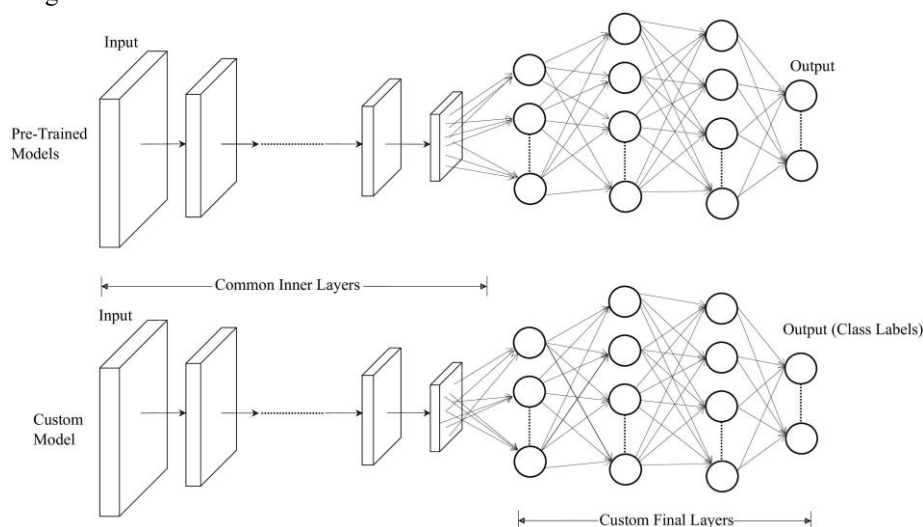


Figure 4: Transfer Learning

Tan, Pooi Shiang, Lim, Kian Ming, Tan, Cheah Heng, and Lee, Chin Poo (2023) [13] proposed using DenseNet-121 deep learning pre-trained with transfer learning technique to extract significant image attributes. DenseNet dramatically reduces processing resources. Experiments are run on three benchmark datasets: soundscapes1, soundscapes2, and urbansound8k. The proposed pre-trained model DenseNet-121 with multilayer perceptron

outperforms past studies on the soundscapes1, soundscapes2, and urbansound8k datasets, with F1- scores of 80.7%, 87.3%, and 69.6%, respectively. Li, Guoqing, Zhang, Meng, Li, Jiaojie, Lv, Feng, and Tong, Guodong (2021) [14] proposed two CNN architectures: DenseDsc and Dense2Net.

These two CNNs are tightly connected, making it

simpler to reuse features across networks. Dense2net employs efficient group convolution, whereas DenseDsc employs more efficient separable convolution based on depth. Both solutions improved parameter efficiency. The proposed strategies are evaluated on the CIFAR and ImageNet datasets. The author achieved 74.2% accuracy on CIFAR-100 and 76.3% on ImageNet (top-1) using DenseDsc. CiFAR-100 achieved 73.68% accuracy, while ImageNet scored 77% using Dense2Net. Still, there is room for improvement in image classification.

3 Transfer Learning

Neural networks trained on large datasets generate knowledge, known as the network's weights [15]. Only learnt characteristics in the form of weights can be retrieved and transferred to any other neural network, rather than training the neural network from scratch.

Instead of beginning from scratch, pre-trained models are trained on a massive dataset. Pre-trained models extract features by removing the output layer [16]. Finally, the network is transformed into a fixed feature extractor by freezing the initial layer weights and retraining only the higher layers with the new problem-specific dataset [17]. In order to fine-tune pre-trained models, convolution operators must first acquire generic properties in the first layer before moving on to dataset-specific features. The model is trained in the last layer. Transfer learning is performed on the network's early and central layers. It employs the tagged data from the job upon which it was trained. Only the final layers are retrained, as shown in figure 4.

Transfer learning can be used with data from completely different but equally relevant source and target areas.

Table 1: Summarizing Deep Learning Pre-Trained Models available in Kera's API documentation

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Para- meters	Depth	CPU time per Inference Step (ms)	GPU time per Inference Step (ms)
Xception	88	0.79	0.945	22.9M	81	109.4	8.1
VGG16	528	0.713	0.901	138.4M	16	69.5	4.2
VGG19	549	0.713	0.9	143.7M	19	84.8	4.4
ResNet50	98	0.749	0.921	25.6M	103	45.6	4.4
Inceptionv3	92	0.779	0.937	23.9M	189	42.2	6.9
DenseNet-169	57	0.762	0.932	14.3M	338	96.4	6.3
v DenseNet-121	33	0.75	0.923	8.1M	242	77.1	5.4
DenseNet-201	80	0.773	0.936	20.2M	402	127.2	6.7
MobileNet	16	0.704	0.895	4.3M	55	22.6	3.4

Pre-trained models are taught on the source domain and then learn to produce significantly higher accuracy results on the target task [18]. The correct network weights are identified through multiple forward and backward rounds. Pre- trained model weights and architecture can be directly applied to our target problem of transfer learning [19]. The ImageNet dataset, which contains millions of annotated photographs from thousands of classifications, is an excellent source for training deep learning pre-trained

models. The pre-trained model on the ImageNet dataset learns a comprehensive set of features and weights, which helps adapt the model to specific target tasks and enhances accuracy [20]. AlexNet, ResNet, GoogleNet[21], VGG (Visual Geometry Group variations include VGG16 and VGG19)[22], and DenseNet[14] are some of the commonly used convolution pre-trained models using transfer learning. These pre-trained models vary in terms of layered structure and convolution approaches.

4 Pre-Trained Models

Pre-trained CNN models are neural networks that have been trained on a large dataset, such as ImageNet, for general image recognition and are now ready to use in different applications. These models are trained to identify a wider variety of characteristics and patterns in photos. citekrishna2019deep. Pre-trained models use Transfer Learning, which implies they are used as a starting point for applications that differ from the model's original training domain [23]. Pre-trained models save time and computing resources over training a neural network from scratch [24]. Transfer Learning, which involves fine-tuning pre-trained models for a given application, needs less data to train.

5 Model Architecture Selection Image Classification

The initial stage in image categorization is to choose the suitable architecture. Experimenting with different designs and layer configurations could produce better results. The CNN design was chosen based on its lower mistake rates and superior performance in similar problems. Keras Applications enable deep learning pre-trained models. Table 1 illustrates some cases with reduced error rates. According to Keras Applications, the models are set up as follows. The pre-trained models are used for feature extraction and fine-tuning.

The top-1 and top-5 accuracy numbers show the model's performance on the large-scale ImageNet validation dataset. Depth represents the network's topological depth. Depth is made up of activation layers, batch normalization layers, and so on. The average time per inference step is calculated using 30 batches and 10 repetitions. Depth refers

to the number of layers and parameters. The experimental set for training these models is as follows:

1. CPU: AMD EPYC Processor (with IBPB) (92 core)
2. RAM: 1.7T
3. GPU: Tesla A100
4. Batch size: 32

The accuracy of the top-1 and top-5 models, the model's depth, and the time required for each inference step are all unquestionably trade-offs.

6 Experimented Datasets

Remote sensing image retrieval systems have grown in

popularity as new feature extraction approaches are developed and tested on fresh datasets. Benchmark datasets provide the foundation for performance evaluation and the use of RSIR methodologies. The literature has seen significant development in the creation of benchmark datasets for the RSIR system. Datasets are classified into two groups based on the retrieval strategy used. Unisource retrieval occurs first, followed by cross-source retrieval. Both the query and the images retrieved by unisource retrieval are from the same source. Both the query and the images obtained by cross-source retrieval come from two different sources. The following sections describe popular remote sensing datasets.

6.1 UC Merced Land Use Dataset

Another name for the dataset is UCM/UC-Merced. There are 100 images and 21 classes at UCM. UCM has limited classes and is small in size. These categories belong to the land cover/land use category. Each image has 256x256 pixel dimensions and the same spatial resolution. A series of large aerial pictures with a spatial resolution of 0.3 meters were obtained from the US Geological Survey's USGS National Map Urban Area. These images include Urban and Built Environment (buildings, crossroads, motorways, overpasses, parking lots, storage tanks, runways), Residential Areas (sparse, dense, mobile home parks), Recreational and Specialised Areas (baseball diamonds, tennis courts), Natural and Agricultural Areas (farms, woods, beaches, streams), Transportation and Infrastructure (aircraft, harbours). [25][26].

6.2 AID

The vast collection of aerial pictures from Google Earth Imagery is known as AID. The spatial resolution of the scene photographs ranges between 0.5 and 8 meters. The dataset includes the following classes: The dataset includes thirty classes, including Urban Infrastructure and Built Environment (Commercial, Industrial, Port, Railway Station, Airport, Parking, Bridge, Viaduct, Storage tanks), Residential Areas (Dense residential, Medium residential, Sparse residential), Public and Recreational Spaces (Park, Playground, Square, Stadium, Resort, School, Centre, Church), Natural Landscapes and Landforms (Forest, Desert, Mountain, River, Pond, Meadow, Farmland, Bare land), Specialised Areas (Baseball pitch)[27][28].

6.3 PatterNet

PatterNet is a vast archive of high-resolution pictures from

Google Earth imagery collected for a select US sites via the Google Map API. 800 256×256 pixel pictures are grouped into 38 categories. Urban and built environment (coastal mansion, nursing home, mobile home park, dense and sparse residential, cemetery, transformer station, wastewater treatment plant, storage tank). Transportation and Infrastructure (freeway, highway, bridge, crosswalk, intersection, railroad, runway, runway marking, closed road, ferry terminal, harbour, shipping yard, aircraft), Recreational and Specialised Areas (baseball field, basketball court, football field, golf course, swimming pool, tennis court), Natural and Agricultural Areas (beach, forest, river, chaparral, Christmas tree farm, solar panel). PatterNet is a good collection of tagged data for image classification and retrieval in remote sensing. Deep learning methods that need a lot of labelled data will profit tremendously from this labelled data[29][30].




































6.4 NWPU-RESISC45

The NWPU-RESISC45 benchmark dataset serves as the basis for remote sensing scene classification. It features forty-five scene classes. Each class includes seven hundred photographs. There are 31,500 256×256 pixel photographs in total. It was created by Northwest Polytechnic University, or NWPU. The image's spatial resolution ranges from 0.2 to 30 meters. The images in this dataset belong to Urban and Built Environment (Commercial area, Industrial area, Palace, Railway station, Church, Thermal power station, Terrace, Parking lot, Freeway, Fly-over, Roundabout, Intersection), Residential Areas (Dense residential, Medium residential, Sparse residential, Mobile home park), Transportation and Infrastructure (Airport, Runway, Aeroplane, Harbour, Ship), Recreational and Specialised Areas (Baseball diamond, Basketball court) [25][30].

Table 2: Remote Sensing Datasets









































DATASET SELECTED	SCENE CLASSES	TOTAL IMAGES CONTAINED IN DATASET	IMAGE SIZE (Pixels)	SPATIAL RESOLUTION (m)
UC-Merced	21	2100	256×256	0.3
AID	30	10000	600×600	8-0.5
NWPU-RESISC45	45	31500	256×256	30-0.2
PatterNet	38	30400	256×256	4.69-0.06

Table 3: UC-MERCED:Remote Sensing Dataset

UC-Merced: Remote Sensing Dataset		
Class 1.	AGRICULTURAL	    
Class 2.	AIRPLANE	    
Class 3.	BASE BALL DIAMOND	    
Class 4.	BEACH	    
Class 5.	BUILDINGS	    
Class 6.	CHAPARRAL	    
Class 7.	DENSE RESIDENTIAL	    

Class 8.	FOREST	    
Class 9.	FREEWAY	    
Class 10.	GOLF COURSE	    
Class 11.	HARBOR	    
Class 12.	INTERSECTION	    
Class 13.	MEDIUM RESIDENTIAL	    
Class 14.	MOBILE HOME PARK	    
Class 15.	OVERPASS	    
Class 16.	PARKING LOT	    
Class 17.	RIVER	    
Class 18.	RUNWAY	    
Class 19.	SPARSE RESIDENTIAL	    
Class 20.	STORAGE TANKS	    
Class 21.	TENNIS COURT	    

Table 4: AID: Remote Sensing Dataset

REMOTE SENSING IMAGES DATASET : AID		
Class 1.	AIRPORT	    
Class 2.	BARE LAND	    
Class 3.	BASEBALL FIELD	    
Class 4.	BEACH	    
Class 5.	BRIDGE	    
Class 6.	CENTER	    
Class 7.	CHURCH	    
Class 8.	COMMERCIAL	    





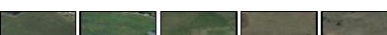








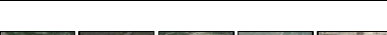






Class 9.	DENSE RESIDENTIAL	
Class 10.	DESERT	
Class 11.	FARMLAND	
Class 12.	FOREST	
Class 13.	INDUSTRIAL	
Class 14.	MEADOW	
Class 15.	MEDIUM RESIDENTIAL	
Class 16.	MOUNTAIN	
Class 17.	PARK	
Class 18.	PARKING	
Class 19.	PLAYGROUND	
Class 20.	POND	
Class 21.	PORT	
Class 22.	RAILWAY STATION	
Class 23.	RESORT	
Class 24.	RIVER	
Class 25.	SCHOOL	
Class 26.	SPARSE RESIDENTIAL	
Class 27.	SQUARE	
Class 28.	STADIUM	
Class 29.	STORAGE TANK	
Class 30.	VIADUCT	

Table 5: NWPU-RESISC45:Remote Sensing Dataset


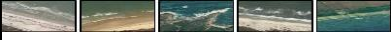


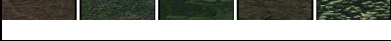
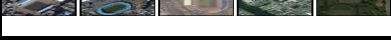











REMOTE SENSING IMAGES DATASET : NWPU-RESISC45		
Class 1.	AIRFEILD	
Class 2.	ANCHORAGE	
Class 3.	BEACH	
Class 4.	DENSE RESIDENTIAL	
Class 5.	FARM	
Class 6.	FLYOVER	
Class 7.	FOREST	
Class 8.	GAME SPACE	
Class 9.	PARKING SPACE	
Class 10.	RIVER	
Class 11.	SPARSE RESIDENTIAL	
Class 12.	STORAGE CISTERNS	

Table 6: PATTERNET: Remote Sensing Dataset

REMOTE SENSING IMAGES DATASET: PATTERNET		
Class 1.	AIRPLANE	
Class 2.	BASEBALL FIELD	
Class 3.	BASKETBALL COURT	
Class 4.	BEACH	
Class 5.	BRIDGE	
Class 6.	CEMETERY	
Class 7.	CHAPARRAL	
Class 8.	CHRISTMAS TREE FARM	
Class 9.	CLOSED ROAD	
Class 10.	COASTAL MANSION	
Class 11.	CROSSWALK	
Class 12.	DENSE RESIDENTIAL	
Class 13.	FERRY TERMINAL	
Class 14.	FOOTBALL FIELD	

Class 15.	FOREST	
Class 16.	FREEWAY	
Class 17.	GOLF COURSE	
Class 18.	HARBOR	
Class 19.	INTERSECTION	
Class 20.	MOBILE HOME PARK	
Class 21.	NURSING HOME	
Class 22.	OIL GAS FIELD	
Class 23.	OIL WELL	
Class 24.	OVERPASS	
Class 25.	PARKING LOT	
Class 26.	PARKING SPACE	
Class 27.	RAILWAY	
Class 28.	RIVER	
Class 29.	RUNWAY	

Class 30.	RUNWAY MARKING	
Class 31.	SHIPPING YARD	
Class 32.	SOLAR PANEL	
Class 33.	SPARSE RESIDENTIAL	
Class 34.	STORAGE TANK	
Class 35.	SWIMMING POOL	
Class 36.	TENNIS COURT	
Class 37.	TRANSFORMER STATION	
Class 38.	WASTEWATER TREATMENT PLANT	

7 Estimating Cost Function to Accelerate Classification Accuracy in DCNN's

The cost function measures the difference in error between actual and anticipated y at its current position. It provides feedback to the model, allowing it to adjust the parameters in order to find global and local minima and reduce error. It iterates in the direction of the steepest decrease until the cost function is 0 or near zero. The model decides to learn at the sharpest gradient point. The loss function represents one training example's mistake. The cost function [31] calculates the average error of an entire training set.

7.1 Forward and Backward Propagation to Compute Cost

A neural network consists of an input layer, one or more hidden layers, and one output layer. During forward propagation, CNN weights are randomly assigned. At each hidden layer node, the input attributes are multiplied by the relevant weights [32]. Every node contributes a bias to the overall total. An activation function is then employed to transform this value into the node's output. The output is obtained by multiplying the network's weights and bias values with the hidden layer output [33]. An activation function is used to change the total. Figure 5 illustrates how the network predicts a value for a specific input value.

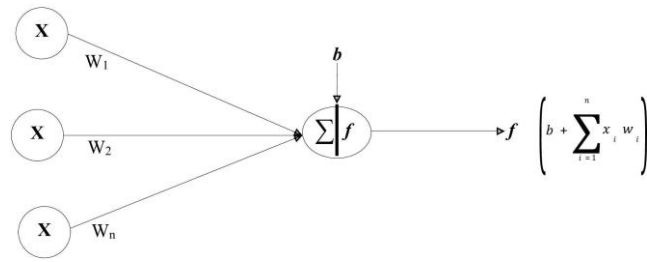


Figure 5: CNN Output Generation

The loss function (C) indicates how effectively the network predicted the output while producing it. Backpropagation is utilised by the network to mitigate losses. Back propagation techniques reduce loss function by altering neural network weights and biases [34]. This

method computes the gradient of the loss function for each network weight. In back propagation, a node's new weight W_{new} is calculated by multiplying its old weight W_{old} by the learning rate (η) and the loss function gradient $\frac{\partial C}{\partial w}$.

$$\frac{\partial C}{\partial w} W_{new} = \underline{W_{old}} - \eta * \frac{\partial C}{\partial w} \quad (1)$$

The gradient of the loss function is calculated using the chain rule of partial derivatives as a product of the gradients of each node's activation function relative to its weight. As a result, the gradients of each node's activation

function influence how the network's node weights change. Fine-tuning involves iteratively updating weights to minimize loss [11].

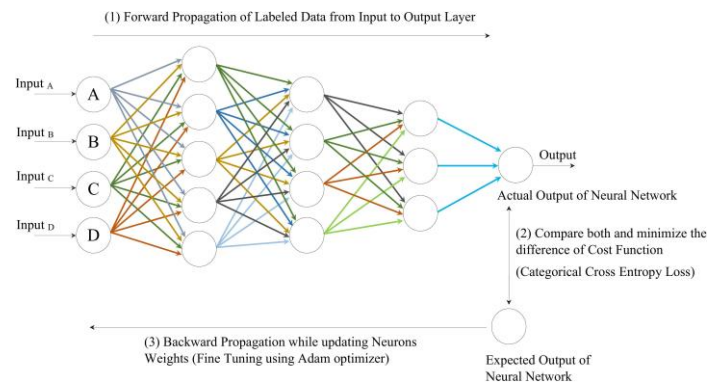


Figure 6: Forward and Backward Propagation

Deep learning neural network models enable machines to identify and grasp intricate patterns in order to make intelligent judgements based on large quantities of data fed into the model for learning (a process known as training; see zhou2020theory). The problem of training deep learning models is complex. The next part discusses the primary research topics that were highlighted during the classification-based training of the pre-trained models, as

well as the steps taken to reduce loss and increase accuracy.

8 Remote Sensing Image Classification based on Pre-Trained Models

Certain pre-trained models, such as VGG-19, Inception-v3, and DenseNet, among others, serve as the foundation for the research. The model that performs the best in terms

of testing accuracy and test loss for image classification among all of these models may be used for the

8.1 General Flow of Transfer Learning Algorithm for Remote Sensing Image Classification

- Load the remote sensing datasets selected for experiments

The dataset is partitioned into train, test, and validation splits, with 70% and 30% ratios. The model is trained using pictures from the training dataset. The validation dataset includes all of the pictures used to validate the model at each epoch. They are used to calculate training and validation accuracies, as well as loss, during each epoch of model training. The test dataset includes all of the

classification and retrieval system of remote sensing.

previously unseen photographs from the Remote Sensing dataset.

- Set the size of input image given as input to pre-trained model (224,224).

Pre-trained models often take 224×224 photos from remote sensing datasets as input, despite the images in the dataset having different dimensions. The photographs in the collection must thus be scaled to fit the required dimensions.

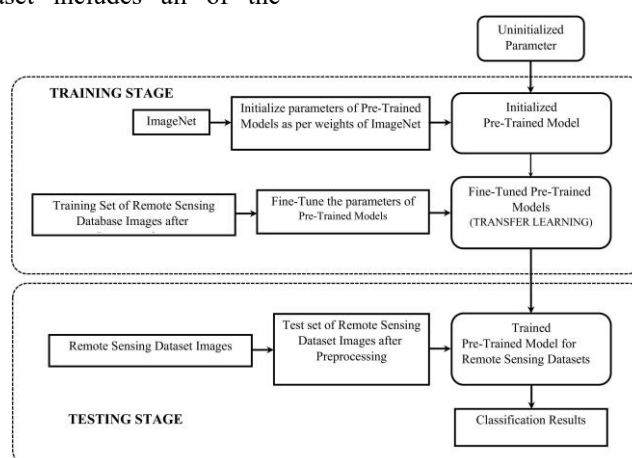


Figure 7: Remote Sensing Image Classification based on Pre-Trained Models

- Model Training

1. Model building

- (a) Load pre-trained base model with its pre-trained weights.
- (b) Customise the mode by changing the last layer based on the number of classes contained in the dataset.
- (c) Multiclass classification is performed using the Softmax activation function at the dense output layer.

2. Compiling Model

The Adam optimiser is used to evaluate the categorical cross-entropy function. Adam optimiser determines the optimal training rate for model compilation.

3. Fitting the Model

Now the model is ready to train. Early stopping is used to cease training the model if the loss from continued validation grows.

4. Performance Evaluation

The accuracy and loss learning curves are plotted. Accuracy, precision, recall, and F1-scores are evaluated.

General flow of transfer learning is presented in figure8.

9 Evaluation Metrics

The evaluation of algorithms is a crucial component of every research project. The Confusion Matrix, Accuracy, Precision,

Recall, and F1-Score are common evaluation measures used by scholars.

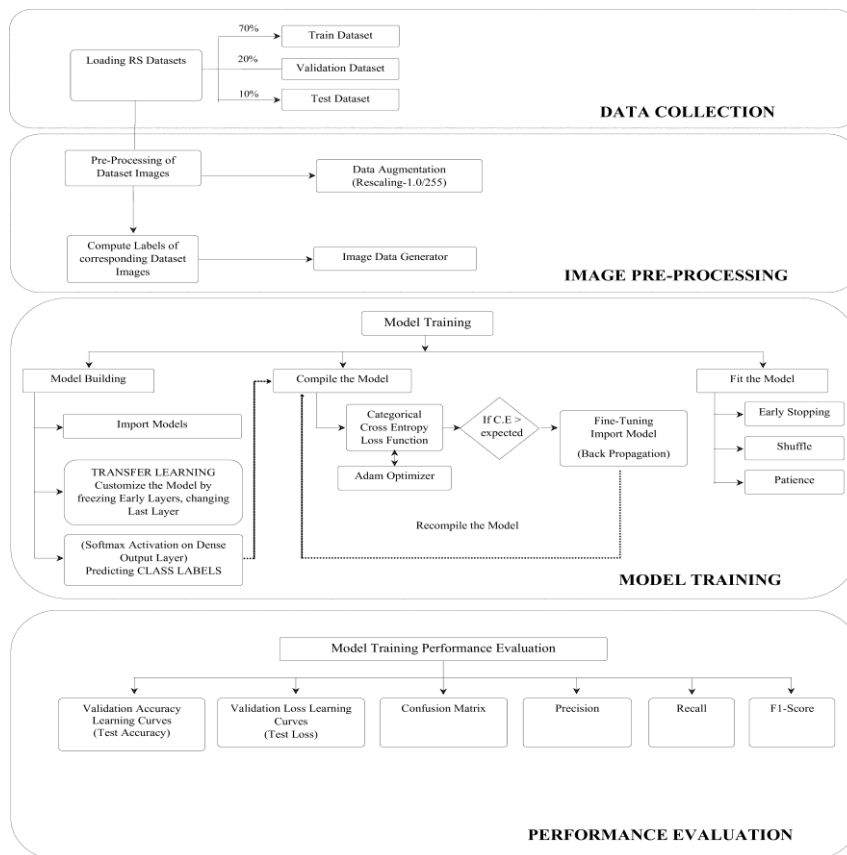


Figure 8: Remote Sensing Image Classification based on Transfer Learning

9.1 Confusion Matrix

Confusion measures are utilised for binary classification scenarios, such as having two classes: YES or NO, as seen in table7.

Table 7: Confusion Matrix

n=165	Predicted No	Predicted Yes
Actual: NO	50	10
Actual: YES	5	100

True positives are those where the classifier predicted yes and the outcome was also yes. True negatives happen when the classifier predicted no and the result was also no. False positives occur when a classifier predicts yes but the outcome is negative. A false negative happens when the classifier predicted no but the actual outcome was yes.

9.2 Accuracy

The accuracy is defined as the number of forecasts divided by the total number of predictions. The proportion of correct predictions made by our model is known as its accuracy. Accuracy is calculated by dividing the number of true positive and negative outcomes by the total number of samples.

$$Accuracy = \frac{TruePositives + TrueNegatives}{TotalSamples} \quad (2)$$

9.3 Precision

Precision is defined as the ratio of actual positive results to those predicted by the classifier.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \quad (3)$$

9.4 Recall

The ratio of accurate positive results to the total number of real samples is known as recall.

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \quad (4)$$

9.5 F1-Score

The F1-Score is the average of precision and recall.

$$F1Score = \frac{2P \times R}{P + R} \quad (5)$$

The F1-Score ranges from 0 to 1. It shows the classifier's accuracy by measuring the proportion of examples it correctly classifies and does not overlook. The F1 Score is calculated by balancing recall and precision. Low recall and high accuracy produce incredibly accurate results, but they miss many difficult-to-classify occurrences. F1 score, accuracy, precision, and recall all improve as the chosen model performs better.

9.6 Training Accuracy

The model's accuracy is based on the training dataset samples. In general, the model's accuracy improves as the number of epochs grows.

9.7 Validation Accuracy

Testing accuracy, also known as validation accuracy, is a statistic obtained using datasets that were not used to train the model. During training, the model was not shown these dataset instances. It exhibits the model's generalisability. It is assumed that the validation accuracy will be lower or equal to the training accuracy. Validation and training accuracy vary greatly, indicating underfitting and overfitting, respectively.

9.8 Training Loss

It is used to update the model's parameters in order to decrease loss on training data. It shows how well the model fit the training data. The model gradually deteriorates as it learns from the input.

9.9 Validation Loss

During training, the non-training data is set aside as a validation set. It evaluates how well the model performs using the validation set. Throughout training, it assesses the model's capacity to generalise to new situations. When validation loss grows while training loss decreases, overfitting occurs.

10 Key Issues Considerations while Training CNN's for Image Classification

A model with too many parameters may perform well on training data but struggle extrapolating to test or unknown data. We refer to this phenomenon as "over-fitting." Long-term training with many parameters and little training data causes overfitting [24]. The suggested pre-trained models leverage CNN's regularization technique to avoid overfitting. Regularization is the process of assigning significant weights to the model. The dropping strategy is another option. Dropping refers to the random elimination of neurons during training. Because it is no longer dependent on a single neuron, it optimizes the model [35].

Under fitting When the model is underfit, it performs

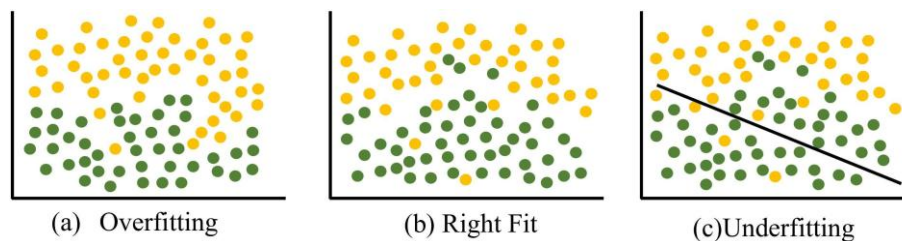


Figure 9: Overfitting and Underfitting

badly on both training and validation data samples because it cannot understand the patterns in the data [36]. Increasing the model's complexity by adding more layers or neurons—basically, more data—resolves underfitting and helps the model interpret complex patterns efficiently [37]. Figure 9 depicts both overfitting and underfitting. CNNs, because to their relatively shallow architecture (less than 30 layers), may fail to capture the complex elements included in remote sensing photos, which is required for more successful picture categorisation that involves the

extraction of high-level semantic properties. Deeper networks can extract more abstract and significant information from incoming data (Shao 2020, Multilabel). Transfer learning has been demonstrated to increase classification accuracy by addressing overfitting and underfitting in CNN's deep learning pre-trained models, such as AlexNet, ResNet, GoogleNet, etc. [1].

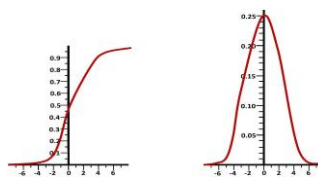


Figure 10: Vanishig Gradient

10.1 Insufficient or Imbalanced Data

A small or imbalanced dataset produces poor classification results. With only 21 classes, the suggested UC-Merced dataset is limited and may not give useful results. To improve training samples and avoid underfitting, transformations like as flipping, scaling, and rotation are applied to existing dataset images to generate new dataset samples. The phrase [38] refers to data augmentation.

10.2 Vanishing and Exploding Gradients

The model struggles to train when the gradients during backpropagation grow too small, a phenomenon known as disappearing gradients. When gradients are large and multiply exponentially over time, the model becomes unstable and incapable of learning new data. This is called the exploding gradient problem [39]. The use of the sigmoid function limits the neural network's ability to learn during forward propagation. A sigmoid function is a logarithmic function whose output value ranges from 0 to 1. It activates the output layers in binary classification [40].

$$\sigma(x) = 1 / (1 + e^{-x})$$

First derivatives of sigmoid functions are bell curves having values ranging from 0 to 0.25.

The partial derivative of nodes with sigmoid activation functions reaches a maximum of 0.25. As the number of network layers increases, the derivative value product falls until a certain point. The partial derivative disappears as the loss function's partial derivative approaches zero. The network weights in deep networks remain constant once the derivative is removed. To lower the loss function during back propagation, a neural network learns by adjusting its weights and bias parameters.

The vanishing gradient problem occurs when a network's performance rapidly degrades because its weights cannot be adjusted, stopping it from learning. The disappearing gradient problem is caused by the derivative of the sigmoid activation function (Hanin 2018). For experimentation, the activation function ReLU (Rectified Linear Units) is employed instead of the sigmoid. When applied to positive input values, ReLU gives a positive linear output [41]. When given negative input, the function will return zero.

The derivative of a ReLU function is 1 for larger-than-zero inputs and 0 for negative inputs. The graph below shows the derivative of a ReLU function.

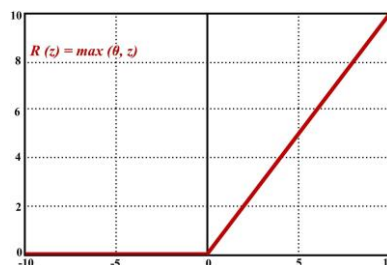


Figure 11: ReLu Activation Function

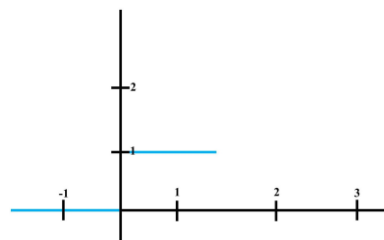


Figure 12: ReLu Derivative

The vanishing gradient research problem is handled utilising the ReLU activation function instead of the sigmoid function. This prevents the gradient from fading since the partial derivative of the loss function will be either 0 or 1. Previously trained models The VGG-19 DenseNet pre-trained model solves the vanishing gradient research problem with the ReLU activation function.

10.3 Batch Normalization

Batch normalization stabilises gradients by ensuring constant mean and variance for each neural network layer [42]. This optimising technique accelerates and stabilises deep neural network training by decreasing the loss function

[42]. The current batch's means=0 and variance=1 are fixed, and activation functions from the hidden layer are normalized [43].

The nonlinear function is applied either before or after the normalization step. Furthermore, batch normalization reduces the internal covariate shift, which is a shift in the network activation function distribution caused by changes to network parameters during training [44]. Batch normalization introduces higher learning rates while avoiding deviating from local minima. Batch normalization regularizes the model by reducing the need for dropouts[44]. The following is a mathematical depiction of adding a batch normalisation layer to a minibatch β :

$$\mu_{\beta} = \frac{1}{m} \sum_{i=1}^m x_i \quad (7)$$

$$\sigma_{\beta^2} = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\beta})^2 \quad (8)$$

$$\hat{x}_i = \frac{x_i - \mu_{\beta}}{\sqrt{\sigma_{\beta^2} + \epsilon}}$$

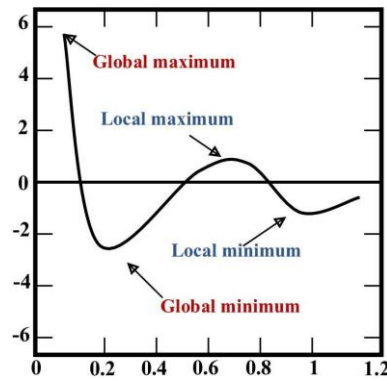


Figure 13: Caption

$$y_i = \gamma x_i + \beta = BN_{\gamma, \beta}(x_i) \quad (10)$$

10.4 To Locate Local Minima and Global Minimum

The loss function is optimized using both local and global minima. The loss function computes the difference between the ground truth and a model's predictions [45]. A local minimum is a point in the parameter space where the local neighborhood's loss function is at its lowest. A global minimum occurs when the loss function is globally minimised in the parameter space [46]. When the cost

function reaches 0 or near to zero, the model stops learning.

Minima local attempts to approximate the global minimum form. On both sides of the current point, the slope of the cost function increases. Global minima produce a saddle point, which is a local minimum [47]. A saddle point occurs when the slope of the cost function rises on both sides of the current point. A saddle point has a negative

gradient on one side, which leads to a local minimum on the opposite side [48].

10.5 Gradient Descent

Gradient descent helps neural networks learn more consistently by computing the cost function. By altering the parameters, the cost function is computed iteratively [49]. To give as little inaccuracy as feasible, the model adjusts its parameters until the function approaches or equals zero. Gradient descent acts as a watchdog, ensuring the model's accuracy at each iterative step until it reaches or equals zero loss[50].

The recommended strategy employs the gradient descent optimisation method. Gradient descent begins at a randomly chosen position. The parameters are progressively adjusted in the direction of the steepest fall. Gradient descent eventually converges to a local minimum. Although gradient descent may trap a local minimum, it does not find the global minimum. To avoid overfitting and trapping in local minima, the proposed technique employs regularisation [51]. Gradient descent methods are classified into three types: batch, stochastic, and mini-batch.

Batch gradient descent calculates the total of errors for each point in a training set. Once each data sample has been trained, the model is updated. This is referred to as an epoch.

Where Θ = Model parameters α = Learning rate, g_t = Gradient of the cost function with respect to the parameters. To minimise loss, this update adjusts the parameters Θ in the negative direction of the gradient. The learning rate, α , defines the step size. The traditional gradient descent technique begins with a high learning rate, manually adjusts the alpha in increments, and maintains a fixed learning rate α . A extremely high learning rate at the start may miss the minima, as indicated in gradient descent, but a lesser learning rate causes very slow convergence. Adam addresses this issue by modifying the learning rate α for each parameter Θ , resulting in faster convergence compared to using a constant global learning rate and standard gradient descent. Adam encounters two stages:

Stochastic Gradient Descent Stochastic Gradient Descent (SGD) runs a training epoch for each sample in the training dataset. It updates the parameters of each training sample one by one [52]. They are easy to store in memory because only one training sample is kept at a time. However, these regular adjustments lead to losses. Thus, batch gradient descent is more computationally efficient than SGD [53]. It is still regarded to be a superior method for avoiding local minima and locating global ones.

Mini Batch Gradient Descent Mini-batch gradient descent combines batch and stochastic gradient descent techniques (Hinton, 2012). The training dataset is separated into manageable batches. It updates each of the batches. Mini batch gradient maintains a balance between stochastic gradient descent for speed and batch gradient descent for computational efficiency [54][55].

10.6 Adam Optimization

Adam stands for Adaptive Moment Estimation. The Adam optimiser, which trains neural networks by changing model parameters to minimise loss, is based on gradient descent [56]. It evaluates each person's adaptive learning rate based on a variety of parameters. Adam is known as the Adaptive Learning Rate Algorithm. Rapid convergence accelerates neural network training [57]. Adam can be represented mathematically as:

$$\Theta = \Theta - \alpha * \beta_t \quad (11)$$

momentum and root mean square propagation.

10.7 Momentum

Momentum accelerates gradients in the correct direction by adding a fraction of the preceding gradient to the current gradient, which speeds up training [58]. When the momentum term proportional to the prior gradients accumulates, the optimisation will move faster in that direction [59]. If the previous few steps were all in the same direction, momentum encourages the current step to follow suit and take fewer steps. The purpose of momentum is to accelerate in predictable directions [58].

$$V_t = \gamma * V_{t-1} + \eta * g_t \quad (12)$$

$$\theta = \theta - V_t \quad (13)$$

Since V_t is a function of the prior momentum vector v_{t-1} , it is the momentum vector at any given time. The learning rate utilised to take the step in the gradient's negative direction is η , while the hyperparameter γ is the momentum decay that exponentially decays the previous momentum vector.

Root Mean Square Propagation (RMSPROP) To adaptively update the learning rate, RMSProp examines the steepness of the error surface for each parameter. Smaller steps are utilised to update parameters with steep

gradients. Larger steps are utilised to update low gradients (kurbiel2017training). Adam employs the hyperparameters β_1 and β_2 to combine momentum and RMSPROP. Adam begins when m_{t-1} and v_{t-1} reach zero in the final version. The t value varies depending on the number of steps taken. Adam outperformed other adaptive learning rate algorithms due to its stability over difficulties and faster convergence [60]. α -Step size for optimization β_1 -Decay rate for momentum. (Typical value is 0.9) β_2 -Decay rate for squared gradients.(Typical value is 0.999) ϵ -Small value to prevent division by 0.(Typically around 8)

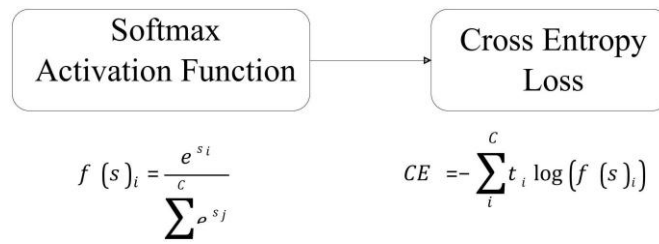


Figure 14: Categorical Cross Entropy

10.8 Computing Categorical Cross Entropy Loss

Categorical cross entropy loss refers to the usage of the Softmax activation function with cross entropy loss. To generate a probability over the C classes for each image, it will train proposed pre-trained models [61]. Figure 14 shows the categorical cross entropy function for multiclass categorisation.

The associated labels in multiclass categorisation are extremely popular. Loss is just the responsibility of the positive class C_p . As with $t_i = t_p$, there is only one non-zero target vector (t). Equation (2) can be changed as follows by removing the summing elements that are 0 due to their target labels:

Where S_p is the CNN score for the positive classification. After defining the loss, the next step is to compute the gradient of CE loss with respect to CNN output neurones, which translates to computing loss with respect to each CNN class score in S . CNN optimises the loss function by tuning net parameters [62].

For classes that are negative, the loss is 0. However, because the Softmax of the positive class is equally determined by the scores of the negative classes, the negative classes' loss gradient is not removed. With the exception of the ground truth class C_p , where the score of $C_p (S_p)$ appears in the nominator, all classes C will have the identical gradient expression. In terms of positive classes, the derivative is:

$$CE = -\log \frac{e^{s_p}}{\sum_j e^{s_j}} \quad \text{!}$$

$$\frac{\partial}{\partial s_p} -\log \frac{e^{s_p}}{\sum_j e^{s_j}} = \frac{e^{s_p}}{\sum_j e^{s_j}} - 1 \quad \text{!!} \quad (15)$$

Derivative with respect to negative class is :

$$\frac{\partial}{\partial s_n} -\log \frac{e^{s_p}}{\sum_j e^{s_j}} = \frac{e^{s_n}}{\sum_j e^{s_j}} \quad \text{!!} \quad (16)$$

The first equation represents gradient variance, often known as the weighted moving average of squared gradients. In the θ update, the learning rate divides the square root of the moving average of squared gradients. This means that

when the gradient variance is large, we reduce the learning rate to get smaller updates. To come closer to the optima, increase the learning rate while keeping the gradient variance to a minimum. Adam is presented as follows:

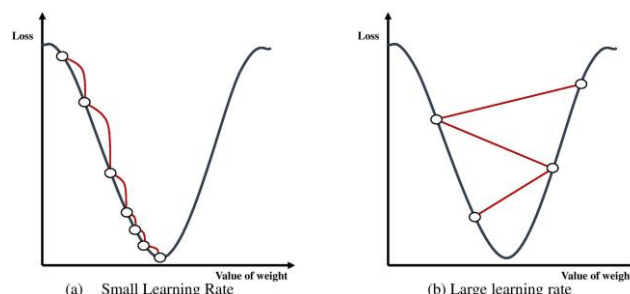


Figure 15: Caption

10.9 Learning Rate or Step Size

The learning rate refers to the number of steps taken to reach the minimum. The cost function analyses and updates the extremely low value [63]. Higher alpha or learning rate values result in larger steps, but it is possible that the minimum will be exceeded. Smaller steps correspond to lower levels of α or learning rate. It has high precision but requires longer calculation time to produce minima [64]. The suggested method sets the learning rate at 0.1. The learning rate determines how often the neural network changes the weights it has learnt. The complex data patterns can be learnt by increasing the training epochs [65].

10.10 Early Stopping

At the epoch when model training loses improvement score, training is ended if the validation loss does not improve further with increasing epochs [66]. We call this an early stop. The ratios of photos in the train, validation, and test datasets are randomised to prevent the model from encountering the same collection of images repeatedly across batches[67].The term "patience" refers to how many epochs should elapse before an early stop is made in the case that the validation set does not improve.

11 Conclusion

This paper's main objective is to make clear the crucial elements to take into account when training pre-trained models and how to steer clear of research issues in order to optimise validation accuracy and reduce validation loss by adjusting the parameters influencing model training during transfer learning. Batch normalisation, gradient descent, the Adam optimiser, estimating categorical cross entropy loss,

10.11 Slow Learning

Deep learning model training can take a significant amount of time. Training the massive ImageNet dataset requires a significant amount of time. Mini-batch gradient descent is used to train models on smaller datasets [68]. This expedites the training process. GPUs are utilised to parallelise the training process.

10.12 Softmax output Activation Function

Prior to calculating loss, activation functions are employed to transform the output vectors (S) produced by neurones. During the training phase, activation functions are used to adjust vectors before calculating the loss [69]. The output vector S is activated using the Softmax function. The class probabilities are displayed in the output. It calculates the likelihood of each possible outcome [70]. The probabilities in output vector S add up to 1 for each possible class outcome (Sharma, 2017). Because S_i corresponds to each element of S, the softmax function can be applied separately to each S_i . Figure 14 shows the mathematical representation of the Softmax function for each S_i class.

and the softmax activation function have all been examined in this study. Taking these training elements into account would undoubtedly improve the pre-trained models' training process in an efficient manner, resulting in increased testing accuracy and reduced testing loss.

References

- [1] I. Dimitrovski, I. Kitanovski, D. Koccev, and N.

- Simidjievski, "Current trends in deep learning for earth observation: An open-source benchmark arena for image classification," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 197, pp. 18–35, 2023.
- [2] K. O'shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.
- [3] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, *et al.*, "Recent advances in convolutional neural networks," *Pattern recognition*, vol. 77, pp. 354–377, 2018.
- [4] N. Ketkar, J. Moolayil, N. Ketkar, and J. Moolayil, "Convolutional neural networks," *Deep Learning with Python: Learn Best Practices of Deep Learning Models with PyTorch*, pp. 197–242, 2021.
- [5] J. S. Ren, L. Xu, Q. Yan, and W. Sun, "Shepard convolutional neural networks," *Advances in neural information processing systems*, vol. 28, 2015.
- [6] W. Zhou, H. Guan, Z. Li, Z. Shao, and M. R. Delavar, "Remote sensing image retrieval in the past decade: Achievements, challenges, and future directions," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2023.
- [7] H. Habibi Aghdam, E. Jahani Heravi, H. Habibi Aghdam, and E. Jahani Heravi, "Convolutional neural networks," 2017.
- [8] U. Markowska-Kaczmar and H. Kwaśnicka, "Deep learning—a new era in bridging the semantic gap," *Bridging the Semantic Gap in Image and Video Analysis*, pp. 123–159, 2018.
- [9] Y. Liu, L. Ding, C. Chen, and Y. Liu, "Similarity-based unsupervised deep transfer learning for remote sensing image retrieval," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 11, pp. 7872–7889, 2020.
- [10] M. E. Paoletti, J. M. Haut, J. Plaza, and A. Plaza, "A new deep convolutional neural network for fast hyperspectral image classification," *ISPRS journal of photogrammetry and remote sensing*, vol. 145, pp. 120–147, 2018.
- [11] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.
- [12] J. Zhang, C. Lu, X. Li, H.-J. Kim, and J. Wang, "A full convolutional network based on densenet for remote sensing scene classification," *Mathematical Biosciences and Engineering*, vol. 16, no. 5, pp. 3345–3367, 2019.
- [13] P. S. Tan, K. M. Lim, C. H. Tan, and C. P. Lee, "Pre-trained densenet-121 with multilayer perceptron for acoustic event classification," *IAENG International Journal of Computer Science*, vol. 50, no. 1, 2023.
- [14] G. Li, M. Zhang, J. Li, F. Lv, and G. Tong, "Efficient densely connected convolutional neural networks," *Pattern Recognition*, vol. 109, p. 107610, 2021.
- [15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.
- [16] L. Zhen, P. Hu, X. Peng, R. S. M. Goh, and J. T. Zhou, "Deep multimodal transfer learning for cross-modal retrieval," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 2, pp. 798–810, 2020.
- [17] Y. Wang, R. Xiao, J. Qi, and C. Tao, "Cross-sensor remote-sensing images scene understanding based on transfer learning between heterogeneous networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2021.
- [18] E. d. S. Puls, M. V. Todescato, and J. L. Carbonera, "An evaluation of pre-trained models for feature extraction in image classification," *arXiv preprint arXiv:2310.02037*, 2023.
- [19] C. Desai, "Image classification using transfer learning and deep learning," *International Journal of Engineering and Computer Science*, vol. 10, no. 9, pp. 25394–25398, 2021.
- [20] A. Alem and S. Kumar, "Deep learning models performance evaluations for remote sensed image classification," *IEEE Access*, vol. 10, pp. 111784–111793, 2022.
- [21] C. Wang, D. Chen, L. Hao, X. Liu, Y. Zeng, J. Chen, and G. Zhang, "Pulmonary image classification based on inception-v3 transfer learning model," *IEEE Access*, vol. 7, pp. 146533–146541, 2019.
- [22] S. Zhou, W. Liang, J. Li, and J.-U. Kim, "Improved vgg model for road traffic sign recognition," *Computers, Materials & Continua*, vol. 57, no. 1, pp. 11–24, 2018.
- [23] U. Muhammad, W. Wang, S. P. Chattha, and S. Ali, "Pre-trained vggnet architecture for remote-sensing image scene classification," in *2018 24th International*

Conference on Pattern Recognition (ICPR), pp. 1622–1627, IEEE, 2018.

[24] V. Risojević and V. Stojnić, “Do we still need imagenet pre-training in remote sensing scene classification?,” *arXiv preprint arXiv:2111.03690*, 2021.

[25] S. Jiang, H. Zhao, W. Wu, and Q. Tan, “A novel framework for remote sensing image scene classification,” *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 42, pp. 657–663, 2018.

[26] Y. Dong and Q. Zhang, “A combined deep learning model for the scene classification of high-resolution remote sensing image,” *IEEE Geoscience and Remote Sensing Letters*, vol. 16, no. 10, pp. 1540–1544, 2019.

[27] G.-S. Xia, J. Hu, F. Hu, B. Shi, X. Bai, Y. Zhong, L. Zhang, and X. Lu, “Aid: A benchmark data set for performance evaluation of aerial scene classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 7, pp. 3965–3981, 2017.

[28] W. Zhang, P. Tang, and L. Zhao, “Remote sensing image scene classification using cnn-capsnet,” *Remote Sensing*, vol. 11, no. 5, p. 494, 2019.

[29] W. Zhou, S. Newsam, C. Li, and Z. Shao, “Patternnet: A benchmark dataset for performance evaluation of remote sensing image retrieval,” *ISPRS journal of photogrammetry and remote sensing*, vol. 145, pp. 197–209, 2018.

[30] Y. Liu, Y. Liu, C. Chen, and L. Ding, “Remote-sensing image retrieval with tree-triplet-classification networks,” *Neurocomputing*, vol. 405, pp. 48–61, 2020.

[31] B. B. Traore, B. Kamsu-Foguem, and F. Tangara, “Deep convolution neural network for image recognition,” *Ecological informatics*, vol. 48, pp. 257–268, 2018.

[32] A. Khan, A. Sohail, U. Zahoor, and A. S. Qureshi, “A survey of the recent architectures of deep convolutional neural networks,” *Artificial intelligence review*, vol. 53, pp. 5455–5516, 2020.

[33] W. Zhou, S. Newsam, C. Li, and Z. Shao, “Learning low dimensional convolutional neural networks for high-resolution remote sensing image retrieval,” *Remote Sensing*, vol. 9, no. 5, p. 489, 2017.

[34] J. Wang, Y. Zheng, M. Wang, Q. Shen, and J. Huang, “Object-scale adaptive convolutional neural networks for high-spatial resolution remote sensing image classification,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 283–

299, 2020.

[35] S. Salman and X. Liu, “Overfitting mechanism and avoidance in deep neural networks,” *arXiv preprint arXiv:1901.06566*, 2019.

[36] J. Li, C. Xu, W. Yang, C. Sun, and D. Tao, “Discriminative multi-view interactive image re-ranking,” *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3113–3127, 2017.

[37] H. Alhichri, A. S. Alswayed, Y. Bazi, N. Ammour, and N. A. Alajlan, “Classification of remote sensing images using efficientnet-b3 cnn model with attention,” *IEEE access*, vol. 9, pp. 14078–14094, 2021.

[38] J. Xin, F. Ye, Y. Xia, Y. Luo, and X. Chen, “A new remote sensing image retrieval method based on cnn and yolo,” *Journal of Internet Technology*, vol. 24, no. 2, pp. 233–242, 2023.

[39] S. Hochreiter, “The vanishing gradient problem during learning recurrent neural nets and problem solutions,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[40] B. Hanin, “Which neural net architectures give rise to exploding and vanishing gradients?,” *Advances in neural information processing systems*, vol. 31, 2018.

[41] Y. Hu, A. Huber, J. Anumula, and S.-C. Liu, “Overcoming the vanishing gradient problem in plain recurrent networks,” *arXiv preprint arXiv:1801.06105*, 2018.

[42] N. Bjorck, C. P. Gomes, B. Selman, and K. Q. Weinberger, “Understanding batch normalization,” *Advances in neural information processing systems*, vol. 31, 2018.

[43] P. Luo, X. Wang, W. Shao, and Z. Peng, “Towards understanding regularization in batch normalization,” *arXiv preprint arXiv:1809.00846*, 2018.

[44] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International conference on machine learning*, pp. 448–456, pmlr, 2015.

[45] K. Kawaguchi, “Deep learning without poor local minima,” *Advances in neural information processing systems*, vol. 29, 2016.

[46] H. Li and A. Yezzi, “Local or global minima: Flexible dual-front active contours,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 1,

pp. 1–14, 2006.

[47] C. A. Floudas and H. T. Jongen, “Global optimization: local minima and transition points,” *Journal of Global Optimization*, vol. 32, pp. 409–415, 2005.

[48] F. Boselie, “Local versus global minima in visual pattern completion,” *Perception & Psychophysics*, vol. 43, pp. 431–445, 1988.

[49] S.-i. Amari, “Backpropagation and stochastic gradient descent method,” *Neurocomputing*, vol. 5, no. 4-5, pp. 185–196, 1993.

[50] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, “Learning to learn by gradient descent by gradient descent,” *Advances in neural information processing systems*, vol. 29, 2016.

[51] L. Mason, J. Baxter, P. Bartlett, and M. Frean, “Boosting algorithms as gradient descent,” *Advances in neural information processing systems*, vol. 12, 1999.

[52] W. A. Gardner, “Learning characteristics of stochastic-gradient-descent algorithms: A general study, analysis, and critique,” *Signal processing*, vol. 6, no. 2, pp. 113–133, 1984.

[53] N. Ketkar and N. Ketkar, “Stochastic gradient descent,” *Deep learning with Python: A hands-on introduction*, pp. 113–132, 2017.

[54] J. Konečný, J. Liu, P. Richtárik, and M. Takáč, “Mini-batch semi-stochastic gradient descent in the proximal setting,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 2, pp. 242–255, 2015.

[55] X. Qian and D. Klabjan, “The impact of the mini-batch size on the variance of gradients in stochastic gradient descent,” *arXiv preprint arXiv:2004.13146*, 2020.

[56] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[57] D. Yi, J. Ahn, and S. Ji, “An effective optimization method for machine learning based on adam,” *Applied Sciences*, vol. 10, no. 3, p. 1073, 2020.

[58] Y. Yan, T. Yang, Z. Li, Q. Lin, and Y. Yang, “A unified analysis of stochastic momentum methods for deep learning,” *arXiv preprint arXiv:1808.10396*, 2018.

[59] N. B. Kovachki and A. M. Stuart, “Analysis of

momentum methods,” *arXiv preprint arXiv:1906.04285*, 2019.

[60] T. O. Hodson, “Root mean square error (rmse) or mean absolute error (mae): When to use them or not,” *Geoscientific Model Development Discussions*, vol. 2022, pp. 1–10, 2022.

[61] P. Li, X. He, X. Cheng, M. Qiao, D. Song, M. Chen, T. Zhou, J. Li, X. Guo, S. Hu, *et al.*, “An improved categorical cross entropy for remote sensing image classification based on noisy labels,” *Expert Systems with Applications*, vol. 205, p. 117296, 2022.

[62] P. Murugan, “Implementation of deep convolutional neural network in multi-class categorical image classification,” *arXiv preprint arXiv:1801.01397*, 2018.

[63] S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le, “Don’t decay the learning rate, increase the batch size,” *arXiv preprint arXiv:1711.00489*, 2017.

[64] R. A. Jacobs, “Increased rates of convergence through learning rate adaptation,” *Neural networks*, vol. 1, no. 4, pp. 295–307, 1988.

[65] Z. Zhang, W. Lu, X. Feng, J. Cao, and G. Xie, “A discriminative feature learning approach with distinguishable distance metrics for remote sensing image classification and retrieval,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 16, pp. 889–901, 2022.

[66] Y. Yao, L. Rosasco, and A. Caponnetto, “On early stopping in gradient descent learning,” *Constructive Approximation*, vol. 26, no. 2, pp. 289–315, 2007.

[67] L. Prechelt, “Automatic early stopping using cross validation: quantifying the criteria,” *Neural networks*, vol. 11, no. 4, pp. 761–767, 1998.

[68] T. Anthony, Z. Tian, and D. Barber, “Thinking fast and slow with deep learning and tree search,” *Advances in neural information processing systems*, vol. 30, 2017.

[69] B. Asadi and H. Jiang, “On approximation capabilities of relu activation and softmax output layer in neural networks,” *arXiv preprint arXiv:2002.04060*, 2020.

[70] B. Wang, X. Luo, Z. Li, W. Zhu, Z. Shi, and S. Osher, “Deep neural nets with interpolating function as output activation,” *Advances in neural information processing systems*, vol. 31, 2018.

