LULC-NEAT: Land Use Land Cover Classification Using NeuroEvolution of Augmenting Topologies

Sumayyea Salahuddin[1], Nasru Minallah[1], Muhammad Athar Javed Sethi[1], Muhammad Ajmal[2], Maryam Mahsal Khan[3]
[1]Department of Computer Systems Engineering, University of Engineering and Technology Peshawar, 25000, Pakistan.
[2]Department of Agricultural Engineering, University of Engineering and Technology Peshawar, 25000, Pakistan.
[3]Department of Computer Science, CECOS University of IT and Emerging Sciences Peshawar, 25000, Pakistan.
***Correspondence:** Sumayyea Salahuddin; Email: sumayyea@uetpeshawar.edu.pk

Citation | Salahuddin. S, Minallah. N, Sethi. M. A. J, Ajmal. M, Khan. M. M, "LULC-NEAT: Land Use Land Cover Classification Using Neuro Evolution of Augmenting Topologies", IJIST, Vol. 6 Issue. 2 pp 876-896, June 2024
Received | June 06, 2024 **Revised |** June 26, 2024 **Accepted |** June 27, 2024 **Published |** June 28, 2024.

Introduction/Importance of Study: NEAT's potency in optimizing neural networks for accurate LULC classification, aimed at better environmental stewardship, is shown.
Novelty statement: LULC-NEAT introduces NeuroEvolution of Augmenting Topologies for optimizing neural networks in land use land cover classification.
Material and Method: The EuroSAT RGB benchmark satellite dataset was preprocessed and evaluated using NEAT to create diverse feed-forward neural networks (FFNNs) with varying hidden layers.
Result and Discussion: The NEAT-evolved FFNN architecture with two hidden layers showed excellent and high accuracy percentages during the training and testing, respectively. Although high training accuracy implies successful feature learning, it also indicates probable overfitting. However, the high accuracy obtained in testing, 99.83%, shows the excellent generalization ability of the model toward unseen data and thus does not overfit. The results were cross-validated with the state-of-the-art CNN models, and the experiments prove that NEAT can be effectively used for LULC classification.
Concluding Remarks: The study supports that NEAT can effectively evolve neural networks for high-accuracy LULC classification, providing a robust alternative to traditional CNN models.
Keywords: Satellite Image Classification; Neuroevolutionary of Augmenting Topologies (NEAT); Deep Learning; Convolutional Neural Network (CNN); EuroSAT.

Introduction:

Efficient land use land cover (LULC) classification using machine learning (ML) techniques is a recent and well-debated venue [1], [2]. Talukdar et al. [1] compared six ML methods such as the support vector machine (SVM), random forest (RF), spectral angle mapper (SAM), Mahalanobis distance (MD), artificial neural network (ANN), and fuzzy adaptive resonance theory-supervised predictive mapping (Fuzzy ARTMAP). They found the RF as the best ML LULC classifier, which needs attention for assessment in other morphoclimatic conditions in the future. Abdi [2] collated the performance of four ML techniques such as SVM, RF, extreme gradient boosting (XGBoost), and deep learning (DL) using Sentinel-2 images. A dataset consisting of multitemporal scenes of summer, winter, autumn, and spring of south-central Sweden was formed and, divided into 70% training and 30% testing subsets using stratified random sampling. They found the SVM as an overall accurate approach followed closely by XGBoost, RF, and DL respectively.

LULC automation is inevitable as the geographic region to be covered is extensive and analysts available to perform probes are few. Remotely sensed datasets are not only vast but also incredibly detailed, often comprising terabytes of data collected from various sources such as satellite imagery, aerial photography, and ground surveys. Usually, satellite application requires manual labeling of objects and structures in the imagery to be valuable in disaster management [3], law enforcement [4], and environmental monitoring [5]. Even though RF [1] and SVM [2] algorithms have shown promising results, classic object detection and classification algorithms are often imprecise and unreliable for these applications. The sheer volume and complexity of these datasets make manual analysis impractical and time-consuming. Automation allows for the efficient and accurate classification, analysis, and interpretation of big data through advanced ML and artificial intelligence (AI) techniques. This increases the scale of applications while simultaneously reducing the time and effort required by human analysts, enabling them to focus on more strategic tasks and deriving insights rather than just gathering and studying data. DL is a family of advanced ML algorithms that have shown assurance for the automation of such tasks. ANNs have gained extreme popularity in DL [6], [7], [8], where these deep ANNs have won competitions in ML, computer vision (CV), pattern recognition (PR), and natural language processing (NLP). However, many existing ML issues can be resolved using smaller neural networks. Neuroevolution [9] is an AI technique that can play a significant role in these cases to determine an optimized network architecture and connection weights. NeuroEvolution of Augmenting Topologies (NEAT) [10] algorithm authorizes the optimized evolution of complex neural network architectures using a genetic algorithm.

The EuroSAT RGB dataset serves as a benchmark colored dataset constructed from Sentinel-2 satellite images [11], [12], [13]. This dataset finds extensive use in ML and DL and is the basis for investigating and practicing the design, estimating, and employing neural networks for LULC classification [14], [15]. In this study, we propose LULC-NEAT, which is an ANN implemented using the NEAT algorithm [10], [16] for the EuroSAT RGB dataset to classify LULC satellite images. In the following sections, we present a review of different studies related to EuroSAT RGB dataset using DL and its applications in this domain.

Helber et al. [12] introduced the EuroSAT dataset to tackle the problem of LULC classification by Sentinel-2 satellite images. The dataset was patch-based and consisted of ten classes from 27,000 labeled and georeferenced images. The two types of images are available: RGB and multispectral MS. The classes are as follows: industrial and residential buildings, annual and permanent crops, highways, rivers, sea/lakes, pastures, forests, and herbaceous vegetation. The dataset was evaluated on a few deep neural networks such as ResNet-50 and GoogleNet. GoogleNet achieved 98.18% of accuracy for EuroSAT RGB and ResNet-50 was the best model with 98.57% of accuracy. For EuroSAT MS: the combination of short-wave infrared (SWIR) bands results in 97.05% of accuracy for ResNet-50, whereas the combination of color-infrared (CI) bands gives 98.30% of accuracy for RGB using ResNet-50. Li et al. [17] described a new variation of the convolutional neural network (CNN) called the Deep Discriminative Representation Learning with - Attention Map (DDRL-AM), which was applied on the EuroSAT RGB data, yielding an accuracy of 98.74%.

To boost the efficiency of small neural networks for satellite image classification, Chen et al. [18] developed a knowledge distillation model and experimentally validated it on different datasets. Their proposed framework achieved 94.74% accuracy on the EuroSAT RGB dataset, 87.03% accuracy on the NWPU-RESISC dataset, and 84.38% on the UC-Merced dataset. Sonune et al. [19] evaluated both ML and DL techniques on the EuroSAT RGB dataset. He concluded that the RF achieved 61.46% accuracy, ResNet-50 achieved 94.25% accuracy, and Visual Geometry Group 19 (VGG-19) achieved 97.66% accuracy. It was also found that the DL techniques outperformed the ML approach on the EuroSAT RGB dataset. But the performance of RF can be improved by hyper tuning its parameters (such as the number of trees, etc.).

Naushad et al. [14] fine-tuned the pre-trained networks such as the Visual Geometry Group 16 (VGG-16) and Wide Residual Networks (Wide ResNet-50) on the EuroSAT RGB dataset. The network performance and execution time are improved using early stopping, adaptive learning rate, gradient clipping, and data augmentation. The results showed that the VGG-16 without data augmentation obtained 98.14% accuracy, VGG-16 with data augmentation obtained 98.55% accuracy, Wide ResNet-50 without data augmentation achieved 99.04% accuracy, and Wide ResNet-50 with data augmentation achieved 99.17% accuracy. Thus, the Wide ResNet-50 with data augmentation outperformed the other three networks for the EuroSAT RGB dataset. Jain et al. [20] presented different variations of RSDnet that used the distillation network (BYOL) for satellite image classification. The RSDnet-3 variation with three channels achieved 90% accuracy for the EuroSAT RGB dataset. Stateczny et al. [21] presented a technique for LULC classification using images obtained from remote sensing. They used Haralick texture features, a directed gradient histogram, a local Gabor binary pattern histogram series, and Harris Corner Detection. An Improved Mayfly Optimization (IMO) technique was employed for efficient feature subset selection. For the actual classification, the Multiplicative Long Short-Term Memory (mLSTM) network was used. The developed IMO-mLSTM approach achieved a 98.52% accuracy rate on the EuroSAT RGB dataset. Aksoy et al. [22] showed that injecting traditional features into small-scale CNN models increases their accuracy, unlike models without this feature injection. The proposed method was tested on the EuroSAT RGB dataset. The chosen features were the sample mean, grey-level co-occurrence matrix (GLCM) features, Hu moments, local binary patterns, histogram of oriented gradients (HOG), and color invariants. The chosen DL models were SqueezeNet, MobileNetV2, ShuffleNetV2, VGG16, and ResNet50V2. When relying solely on DL models, the results were as follows: SqueezeNet achieved a maximum accuracy of 0.6778, while ShuffleNetV2 demonstrated a maximum accuracy of 0.8502. When all traditional features were injected into the DL models, the results were as follows: SqueezeNet reached a maximum accuracy of 0.7618, and ShuffleNetV2 demonstrated a maximum accuracy of 0.8998. Thus, an impressive improvement in accuracy can be observed.

Rangel et al. [23] compared CNNs against transformer-based methods using EuroSAT RGB dataset for enhanced accuracy and efficiency in LULC analysis. The convolutional models were AlexNet, ResNet-50, ResNeXt, DenseNet, MobileNetV3, EfficientNetV2, and ConvNeXt. The transformer models were ViT, Swin Transformer, and MaxViT. Results were obtained both when trained from scratch and when retrained with pre-trained weights (obtained from models pre-trained on ImageNet). All models show improved accuracy when retrained with pre-trained weights compared to being trained from scratch. Transformer-based models (ViT32, SwinB, and MaxViT) showed notable improvements, with MaxViT achieving the highest accuracy overall of 0.99. Yadav et al. [24] classified satellite images based on their topologies and geographical features utilizing the CNNs on the EuroSAT RGB dataset. The three pre-trained baseline models are ResNet-50, ResNet-101, and GoogleNet. Additional sequence layers were incorporated into them in relation to CNNs, and the data is pre-processed using LAB channel operations. The GoogleNet model achieved the highest accuracy (99.68%), precision (99.42%), recall (99.51%), and F1-Score (99.45%) when applied to the preprocessed dataset. It was reported that increasing the number of layers in a CNN does not always lead to improved results for a medium-sized dataset. GoogleNet, a 22-layer CNN, outperformed the 50-layer ResNet-50 and the 101-layer ResNet-101 in terms of both speed and accuracy. Table 1 lists the comparison of LULC classification studies for the EuroSAT RGB dataset. Among these approaches, the GoogleNet with preprocessing by Yadav [24] and the Wide ResNet-50 with data augmentation by Naushad [14] outperformed all the other approaches.

Table 1: Comparison of LULC Classification Studies for the EuroSAT RGB Dataset.

Authors	Year	Model	Accuracy
Chen et al. [18]	2018	Knowledge distillation	94.74%
Helber et al. [12]	2019	Google Net	98.18%
Helber et al. [12]	2019	ResNet-50	98.57%
Li et al. [17]	2020	DDRL-AM	98.74%
Sonune et al. [19]	2020	Random Forest	61.46%
Sonune et al. [19]	2020	ResNet-50	94.25%
Sonune et al. [19]	2020	VGG19	97.66%
Naushad et al. [14]	2021	VGG16 (Without Data Augmentation)	98.14%
Naushad et al. [14]	2021	VGG16 (With Data Augmentation)	98.55%
Naushad et al. [14]	2021	Wide ResNet-50 (Without Data Augmentation)	99.04%
Naushad et al. [14]	2021	Wide ResNet-50 (With Data Augmentation)	99.17%
Jain et al. [20]	2022	RSDnet-3	90.00%
Stateczny et al. [21]	2023	IMO-mLSTM	
Aksoy et al. [22]	2023	SqueezeNet	98.52%
Aksoy et al. [22]	2023	SqueezeNet + All Traditional Features	67.78%
Aksoy et al. [22]	2023	ShuffleNetV2	76.18%
Aksoy et al. [22]	2023	ShuffleNetV2 + All Traditional Features	89.98%
Rangel et al. [23]	2024	ViT32 (Retrained with Pre-trained Weights)	97.20%
Rangel et al. [23]	2024	SwinB (Retrained with Pre-trained Weights)	98.70%
Rangel et al. [23]	2024	MaxViT (Retrained with Pre-trained Weights)	99.00%
Yadav et al. [24]	2024	GoogleNet (With Preprocessing)	99.68%

Over the years, artificial neural networks (ANNs) have demonstrated strong performance in solving LULC classification tasks. Table 1 indicates that numerous DL models have been developed to improve the classification accuracy using a large number of layers and parameters. It should be noted that there are other alternatives: the classification of LULC can be carried out using small neural networks with fewer parameters, for example, implemented by Aksoy et al. [22], Yadav et al. [24], and neuroevolution, which also allows developing an optimal architecture of a neural network and determining connection weights. For this task, it is recommended to use the NEAT approach, since it quickly adapts the structure and weights of neural networks, which reduces premature convergence and maintains a variety of potential solutions. Thus, NEAT can significantly improve LULC classification, create more optimal and efficient neural networks. The aim of current study is to enhance the accuracy of LULC classification using NEAT. Following are specific objectives aligned with the overall aim:

- To assess NEAT in classifying different LULC categories on the baseline EuroSAT RGB satellite dataset.

- To fine-tune the LULC-NEAT approach by adjusting different hyperparameters, such as population size, number of layers, and activation functions, to achieve optimal classification accuracy.
- To provide a comprehensive analysis of the results produced by the LULC-NEAT approach, focusing on classification accuracy, computational efficiency, and robustness.
- To compare performance of the LULC-NEAT with leading CNN models such as GoogleNet and Wide ResNet-50 for LULC classification and presenting the results of the comparison.

Material and Methods:

Over the years, the classification of LULC has been done using Neural Networks. Table 1 depicts various architectures employed for satellite image classification [12], [14], [17], [18], [19], [20]. It is clear that the architectures based on neural networks and their variations did really excellent. For this reason, we implemented neural network models via NEAT and compared them with CNNs. The models underwent fine-tuning on the EuroSAT RGB dataset and were trained using Python's TensorFlow framework.

Dataset:

In this section, we provide EuroSAT dataset characteristics, training and testing split, and the applied pre-processing steps [11], [12], [13].

- **Source:** It is a novel patch-based dataset derived from Sentinel-2 satellite [25]. Sentinel-2 satellite provides free and open-source imagery founded by the Copernicus Program run in partnership with the European Space Agency (ESA) [26].
- **Bands:** There are two versions of the dataset: a) RGB containing Red, Green, and Blue frequency bands only, and b) multispectral containing thirteen spectral bands. The EuroSAT RGB is utilized in this study.
- **Image Size and Spatial Resolution:** The size of each image is 64 × 64, while the spatial resolution is 10 meters.
- **Classes:** It has ten classes covering industrial and residential buildings, annual and permanent crops, highways, rivers, sea and lakes, pastures, forests, and herbaceous vegetation. Figure 1 shows some sample images from this dataset.
- **Image Count:** There are 2000 to 3000 images per class with a total of 27000 labeled and georeferenced images. Thus, the dataset is imbalanced.
- **Training and Testing Split:** The dataset was reduced into a compact, balanced dataset of 1000 images using a random sampling technique containing 100 images per class. The dataset was then split into 70% and 30% for training and testing using a stratified sampling approach.
- **Pre-processing:** Operations such as normalization (i.e. rescaling in the range of 0-1), splitting (70/30), and label encoding were applied.

Artificial Neural Networks (ANNs):

The ANNs consist of interconnected processing units known as neurons, where each neuron has a connection weight known as a synaptic weight. There are three types of neurons: input, hidden, and output. The number of input neurons is determined by the number of features in the dataset, while the number of output neurons is based on the number of classes being predicted for a particular problem. The quantity of hidden neurons can be adjusted to meet the desired level of functionality. Due to the generic nature of NN, they are known as the universal function approximators and are used to solve highly complex, non-linear problems besides regression and classification problems. They emulate the behavioral and adaptive abilities of the central nervous system [27], [28]. The ANN architecture is a particular network formed by a precise structure of nodes and connections between them. Usually, the architecture is fixed and specified before training. There can be numerous solution architectures for a given problem, but the most efficient might not be obvious. The weights of specified network architecture must be tuned to minimize the loss function and to reach an optimized form. Figure 2 shows a sample ANN architecture consisting of three input nodes, one hidden layer with four neurons, and two output nodes. Backpropagation (BP) is a popular algorithm to train the ANN [29]. It determines the error between the actual output and the network output, propagates it back through the network from the output layer to the hidden layer, and adjusts the weights and biases accordingly. It is based on gradient descent, an optimization technique that adjusts the weights in proportion to the negative gradient of the error surface. But, its limitations are slow convergence, getting stuck in local minima, and sensitivity to input data [30]. Alternatively, the genetic algorithm can be used as a search heuristic to determine the neural network weights [31][32].

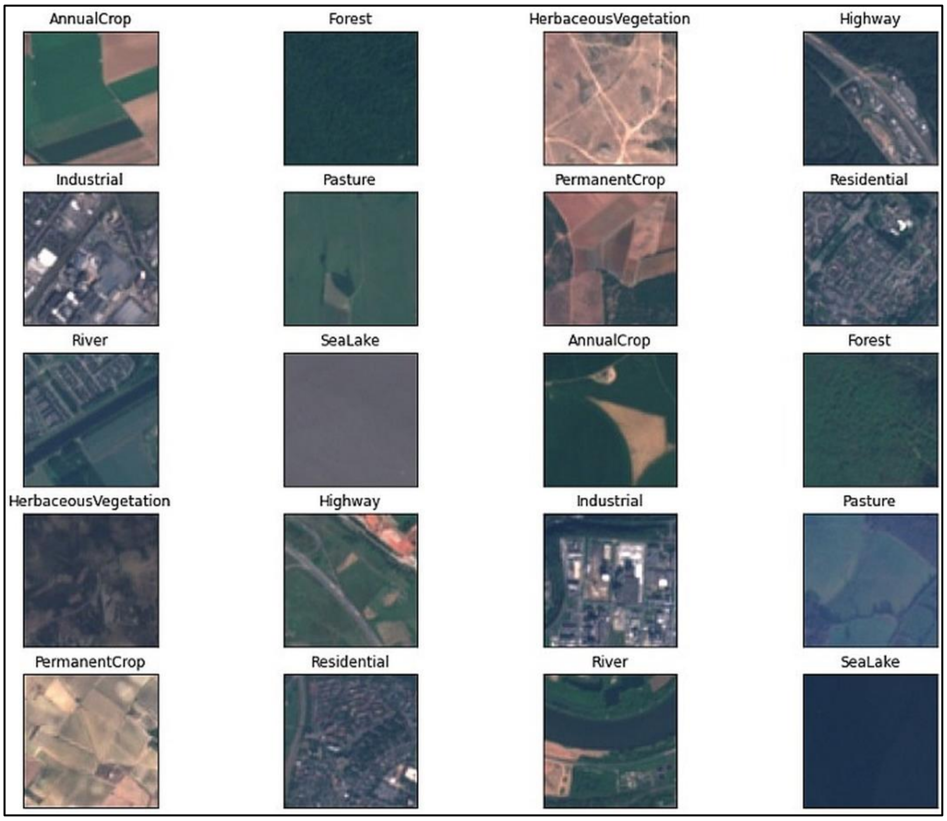

Figure 1: Sample EuroSAT RGB Dataset Images with Spatial Resolution 64×64×3.

Genetic Algorithm (GA):

In 1975, John Holland presented a metaheuristic inspired by natural evolution theory known as a genetic algorithm (GA) [33]. It solves complex problems and focuses on optimization. It takes a population of possible solutions for a given task and selects the fittest individuals to produce the children for the next generation. It is an iterative process, which continues until the population with the best solutions has been reached [34]. The five-phase GA process is shown in Figure 3 that starts with an initial population of solutions with a set of genes encoded as binary, integer, float, or permutation. A fitness function computes the fitness score of each chromosome, and the fittest individuals are selected based on their fitness scores. The crossover point is chosen randomly within the gene, and offspring are created by swapping parent genes among themselves. After crossover, the offspring chromosomes may undergo mutation, where random changes are introduced to some genes. Once the population satisfies the stopping criteria, the genetic algorithm terminates with an optimal solution. Unlike gradient descent and Backpropagation, GA bypasses the local minima problem and can give better weights.

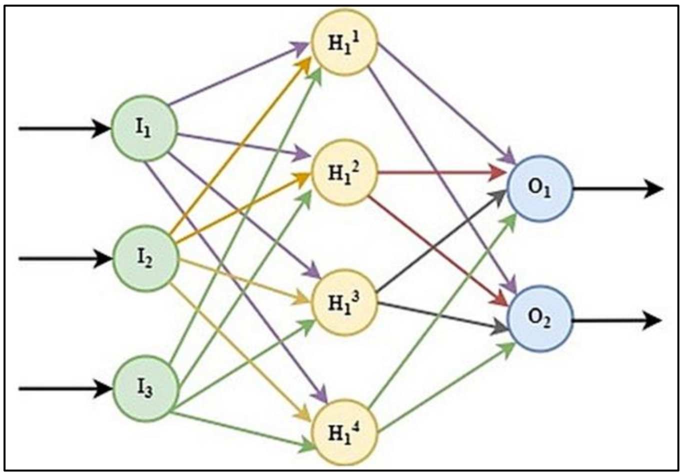

Figure 2. A Sample Neural Network Architecture.

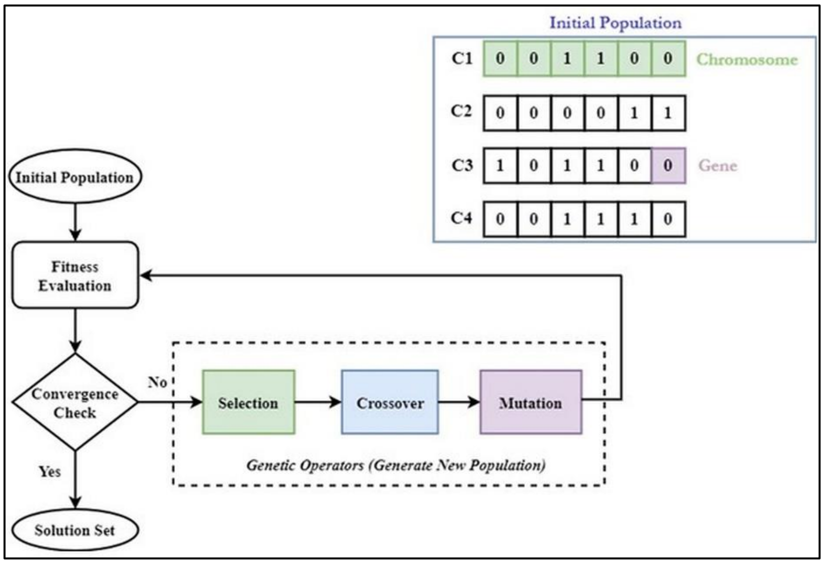

Figure 3. Flow Chart of Genetic Algorithm with Sample Initial Population.

Neuroevolution (NE):

Neuroevolution (NE) is an AI technique that uses evolutionary algorithms to evolve ANNs and yield better architectures, weights, parameters, and rules [9]. It follows population-based optimization and assigns a fitness score to each genome based on phenotype (i.e. actual neural network) performance. The optimization loop then begins, allowing innovation, testing, and classification to lead to a neural network that solves the applied problem exquisitely. Figure 4 illustrates the process of NE inspired by GA and broadening its principles to the domain of neural network optimization, combining the strengths of evolutionary computation with the power of neural networks. The population is segregated into species so that all members of the same species have similar genomes. Speciation prevents the loss of diversity in the population and premature convergence. Recombination (or crossover) and mutation generate new candidate neural networks. These are trained by standard methods (e.g., backpropagation or other training algorithms such as stochastic gradient descent (SGD), adaptive moment estimation (ADAM), momentum, etc.). The trained neural networks are used to solve the problem. The performance of each neural network is evaluated using a fitness function on its ability to carry out the task. The fittest genomes are chosen for creating the next generation according to fitness assessments. Poorly performing genomes are removed from the pool, and the genomes are clustered into species to retain diversity. The best genomes are selected as parents to produce the next generation, and they associate with each other through genetic operations. Genetic operations, fitness evaluations, and selection are performed repeatedly until the neural networks evolve to accomplish the task optimally.

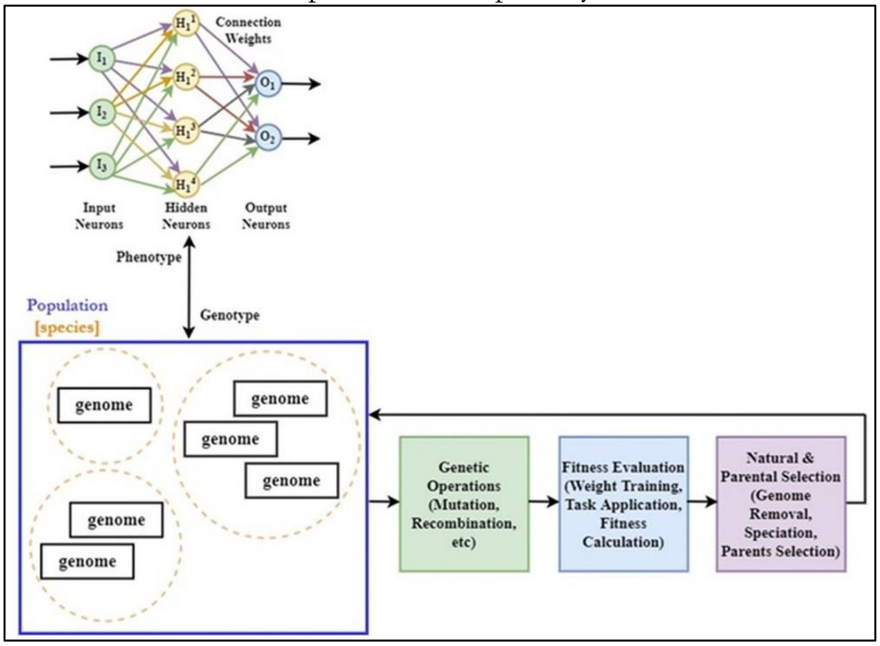
Connection Weights
I1
I2
I3
H11
H12
H13
H14
O1
O2
Input Neurons
Hidden Neurons
Output Neurons
Phenotype
Genotype
Population [species]
genome
Genetic Operations (Mutation, Recombination, etc)
Fitness Evaluation (Weight Training, Task Application, Fitness Calculation)
Natural & Parental Selection (Genome Removal, Speciation, Parents Selection)

Figure 4: Flow Chart of Neuroevolution.

NeuroEvolution of Augmenting Topologies (NEAT):

The NEAT utilizes an optimization approach derived from NE. In the fixed architecture of NE, the objective is to optimize the connection weights only that set up the network functionality. The question NEAT raise is: can evolving both weights and architectures give an edge over just evolving weights on the fixed architecture? In a feed-forward neural network with fixed architecture, depending on the task at hand, the loss function is determined and then adjusted using a gradient-based approach like Adam or SDG. NEAT on the other hand says why not learn or search the suitable architectures and then tune their connection weights accordingly [35]. So, it starts with simplistic possible neural networks with direct connections between input and output neurons and their connection weights. It creates the initial population of such neural networks. These networks are then evolved using crossover and mutation to search for better and improved networks by tuning the parameters, thereby creating new populations [36], [37]. This arises technical challenges such as how to genetically represent dissimilar architectures to crossover in a significant way to avoid suboptimal solutions, how to save the architectural changes that can be optimized in a few generations disappearing prematurely from the population, and how to keep minimal architectures throughout evolution without a specialized fitness function? The NEAT solves these challenges by using direct encoding, global historical markings, and speciation [10].

For every neural network in the population, NEAT keeps the genome known as genotype in the background that corresponds directly to a network known as phenotype. The genome consists of node genes (that express artificial neurons) and connection genes (that express synaptic weights). Node genes list input, hidden, and output nodes (or neurons). The connection genes illustrate global innovation number (or historical marker), from gene, to gene, weight, and activeness of connections. The connection genes can be active or inactive and can be recurrent or non-recurrent. The combination of genotype and phenotype is known as a chromosome [35]. Figure 5 shows the sample NEAT chromosome encoding a neural network. There are two input nodes, one hidden node, and one output node. There are six connection specifications, one of which is recurrent (from node 4 to node 3) with innovation number 7, and one is inactive (from node 1 to node 4) with innovation number 1.

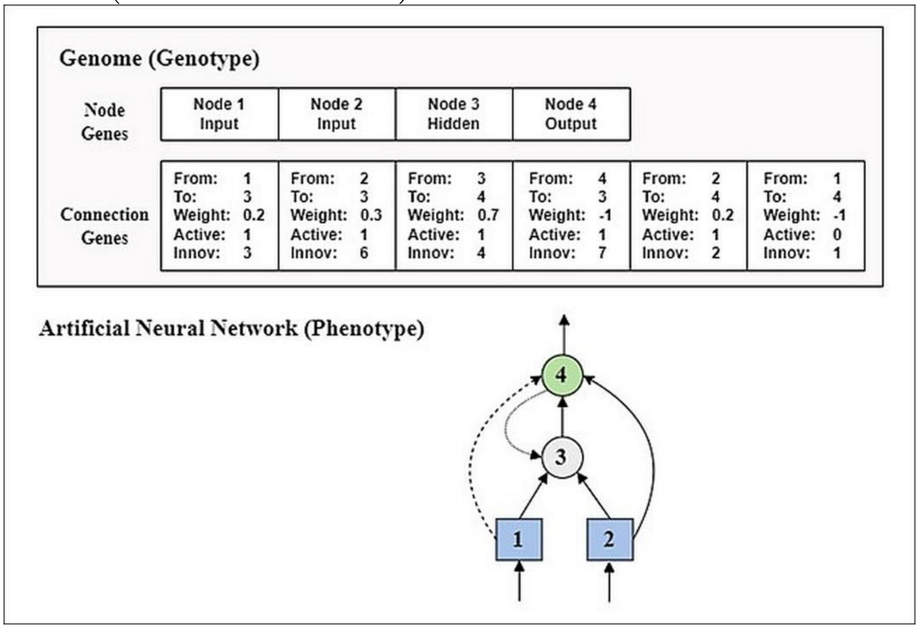
Genome (Genotype)
Node Genes
Node 1 Input
Node 2 Input
Node 3 Hidden
Node 4 Output
Connection Genes
From: 1 To: 3 Weight: 0.2 Active: 1 Innov: 3
From: 2 To: 3 Weight: 0.3 Active: 1 Innov: 6
From: 3 To: 4 Weight: 0.7 Active: 1 Innov: 4
From: 4 To: 3 Weight: -1 Active: 1 Innov: 7
From: 2 To: 4 Weight: 0.2 Active: 1 Innov: 2
From: 1 To: 4 Weight: -1 Active: 0 Innov: 1
Artificial Neural Network (Phenotype)
4
3
1
2

Figure 5: A Sample of NEAT Genotype and Phenotype.

When neural network architecture in the population crosses over with another network, the important information may vanish. This is known as the competing conventions problem [10]. To solve this problem, the NEAT uses global innovation numbers. These numbers are the historical markers that maintain the historical origin of each gene throughout its evolution. During the process of crossover, genes may either match or differs between the two parents. Matching genes have the same innovation numbers in both parents. Disjoint genes are found within the range of the other parent's innovation numbers, while excess genes occur outside this range. To construct an offspring, both parent genes are crossed over randomly from the matching genes while for the excess or disjoint genes, a more fit parent is preferred. The NEAT performs mutation by adding a new node, new connection between existing nodes, removing an existing node, and removing the connection between existing nodes. Once mutated, these changes in the connection genes are shown by fixed innovation numbers. These architectural changes must be protected using population speciation [36]. The grouping of genomes into species is done by calculating the compatibility distance δ between pairs of genomes using Eq. 1 considering:

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 . \overline{W} \qquad (1)$$

Here, E, D, $\overline{W}$, N are the excess genes, disjoint genes, the average weight difference of matching genes, and the number of genes in the larger genome, respectively. c_1, c_2, and c_3 are coefficients that control the relative importance of excess genes, disjoint genes, and average weight difference. The genomes with compatibility distance δ above a threshold δ_t are placed into a new species while lower are placed in existing species with the most matching and compatible representative genome.

Convolution Neural Network (CNN):

The ANNs need meticulous consideration of the architecture and its thorough tuning of the hyperparameters such as the number of hidden layers, number of nodes in a layer, number of epochs, learning rate, batch size, activation function, backpropagation technique, and regularization techniques [38], [39]. The standard Neural Network (ANN) is inefficient for high-dimensional data. For example, if the dimension of a color image is 100x100x3 and there are

100 nodes in the first hidden layer, then there are 3 million weights parameters. When the image dimension is doubled to 200×200×3 with the same number of nodes, the weight parameters reach to 12 million. Therefore, an efficient solution is needed to handle this problem. The CNN is the most popular DL technique to solve Image classification [40]. It has a convolutional layer where filters (or kernels) scan the image and look for specific features, a pooling layer where the image reduces into a smaller dimension, and a fully connected layer to classify it. Thus, CNN provides feature learning, dimensionality reduction, and classification [41]. Figure 6 shows a 5-layer CNN with two convolutional layers, two pooling layers, and a fully connected layer. It can be used to classify land cover in satellite images.

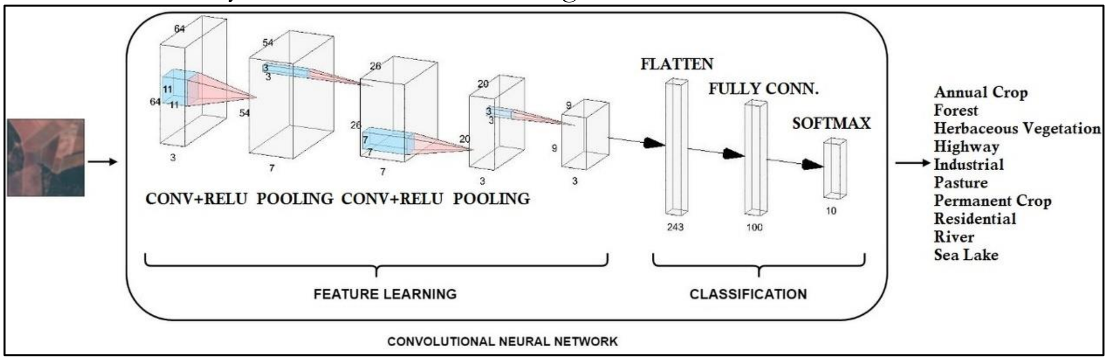

Figure 6: An Example of a Convolutional Neural Network.

Classification Based on NEAT:

In this paper, categorical LULC images were classified using the EuroSAT RGB Dataset. The simplified framework of object categorization system is depicted in Figure 7. It consists of four main steps: pre-processing, data encoding, model training and validation, and model evaluation and prediction. First, the class labels are obtained as a list of strings, with each string representing a directory in the dataset directory. Subsequently, the count for the images in each category is acquired. Then, the following four steps are performed:

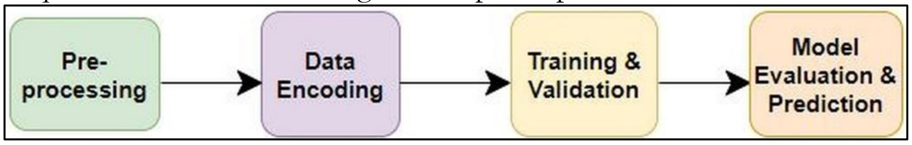

Figure 7. Methodology.

Pre-Processing:

First, a dictionary **data** is created that stores the full image path as the key and its corresponding label as the value. This is done by first iterating through each label in the **labels** list and then iterating through each image in the directory corresponding to that label. For each image, the data dictionary is updated by adding a key-value pair, where the key is the full image path and the value is the image label. Then two panda's series **X** and **y** are created from the data dictionary, where **X** contains the image paths and **y** contains the label. Next, the data is split into training (70%) and testing (30%) sets using the Scikit-learn class **Stratified Shuffle Split** [42]. Subsequently, by iterating through the split data and for each training and testing set, the old directory paths are replaced with the new directory paths for the training and testing sets using the re library. Finally, two Numpy lists i.e. **train_path_map** and **test_path_map** was created that contain the old and new paths for the training and testing images, respectively, in the form of tuples (**old path**, **new path**). Next, the Keras class **Image Data Generator** is used to generate the batches of image data for training and testing and represented as a tuple of two Numpy arrays (images and labels). Operations such as normalization (i.e. rescaling in the range of 0-1) and creation of a one-hot encoded vector for verifying the output of neural networks were performed using Keras.

Data Encoding:

The data generated by the training and testing generators in the pre-processing stage was extracted using the **next ()** method [43], [44] and stored in variables **X_train**, **y_train**, **X_test**, and **y_test**. The **next ()** method returns a tuple containing two elements, the first is the training/testing images and the second is the corresponding labels. The **X_train** and **X_test** is 4D array of shape (batch_size, height, width, channels) and the y_train and y_test is 2D array of shape (batch_size, num_classes). The reshaping of images from a 4D array (batch_size, height, width, channels) to a 2D array (batch_size, height*width*channels) is performed using the **reshape ()** method. The reshaped Numpy arrays are then converted to Python lists, with the names **lx_train**, **ly_train**, **lx_test**, and **ly_test** respectively. It was accomplished by calling the **tolist ()** method on the **X_train**, **y_train**, **X_test**, and **y_test** arrays.

Model Training and Validation:

Once the images are in a list format, they are ready to be classified using the NEAT algorithm. For EuroSAT RGB, the input nodes are set equal to 12288 and output nodes are set equal to 10. One, two, & three hidden layer neural network architectures were tried with a population size of five and ten for sigmoid and Rectified Linear Unit (RELU) [45] activation functions. The fitness threshold was set equal to the batch_size. The training accuracy, generation time, and average training accuracy were determined for each network.

Model Evaluation and Prediction:

Eventually, use the testing dataset to calculate the testing accuracy and average testing accuracy of the NEAT-based neural networks.

Performance Metrics:

In this research study, a balanced EuroSAT RGB dataset was used for experiments with each class having the same number of samples. Accuracy is a reliable metric for estimating the performance of a technique on the balanced datasets. Additionally, computational time was used to evaluate the technique's efficiency.

Training Accuracy:

It is determined by comparing the model's outputs with the actual labels of the training data. The training accuracy measures model's ability to predict the output correctly for a given input during the training phase.

Testing Accuracy:

It is determined by comparing the model's predictions with the true labels of the testing data. The testing accuracy measures how well the model generalizes to new, unseen data.

Overall Accuracy:

Overall accuracy is a measure of the model's ability to correctly classify instances, both during the training and testing phases, out of the total number of instances. It can be calculated using Eq. 2.

$$\text{Overall Accuracy (OA)} = \frac{(\text{Training Acc.} + \text{Testing Acc.})}{2} \qquad (2)$$

Computational Time:

Computational time quantifies the time required for a model to train and test on a dataset. It depends on factors including the hardware used, the dataset size, and the intricacy of the model.

System Specifications:

For model training, the following specifications were used:

- Apple MacBook Pro (Retina, 13-inch)
- Processor: 2.9 GHz Intel Core i5 (Turbo Boost up to 3.3 GHz)
- Memory: 16 GB 1867 MHz DDR3
- Graphics: Intel Iris Graphics 6100 1536 MB

Results:

A brief and precise summary of our experimental results and their interpretation using the NEAT-based neural network architecture is presented below.

Hyperparameter Tuning in NEAT– An Overview of Key Parameters:

NEAT has many parameters that control its implementation and various aspects of the training process [16]. Table 2 shows various NEAT parameters used for the EuroSAT RGB dataset. These values can be adjusted to fine-tune the training process and optimize the performance of the final model.

Table 2: NEAT Parameters for LULC Classification using the EuroSAT RGB Dataset.

Parameters	Values	Description
fitness_criterion	Max	It is the criterion used to evaluate the fitness of a network. Here, the setting is configured to 'max', indicating that the network with the highest fitness score will be regarded as the optimal one.
fitness_threshold	Batch size	It is the minimum fitness value required for a network to be considered fit enough to be included in the population of networks. Here, it is set to 'batch size', which is 700 for training and 300 for testing.
pop_size	5 and 10	It specifies the number of networks in a population. It affects the diversity and the exploration of the solution space. Here, it is set to five and ten.
reset_on_extinction	False	It controls whether or not the algorithm should reset the population if it becomes extinct. Here, it is configured as 'False'.
activation_default	Sigmoid	It specifies the default activation function used for nodes within the network. The current setting is configured to use the 'Sigmoid' function.
activation_mutate_rate	0.0	It is the probability that the activation function of a node will be randomly changed during mutation. Here, it is set to '0'.
activation_options	Sigmoid	It lists the options for activation functions that can be used in the network. Here, it is set to only 'Sigmoid'.
conn_add_prob	0.5	It is the probability that a new connection will be added to the network during mutation. For this study, it is set to '0.5'.
conn_delete_prob	0.5	It is the probability that a connection will be removed from the network during mutation. Here, it is set to '0.5'.
enabled_default	True	This controls the default status of a node, whether it is enabled or not. Here, it is 'True'.
enabled_mutate_rate	0.01	It is the likelihood that the node's status will be randomly changed during mutation. Here, it is '0.01'.
feed_forward	True	It controls the type of network topology to use. Here, it is set to True, which means the network is feedforward.
initial_connection	full_direct	This parameter specified the initial connection of the network. For this study, it is set to 'full_direct' which means all input nodes are connected to all output nodes.
node_add_prob	0.2	It is the probability that a new node will be added to the network during mutation. For this study, it is 0.2.

node_delete_prob	0.2	It is the likelihood that a node will be removed from the network during mutation. For this study, it is 0.2.
num_hidden	1, 2, and 3	This parameter controls the number of hidden layers in the network. Neural network architectures with one, two, and three hidden layers were attempted.
num_inputs	12288	This parameter specifies the number of input nodes in the network. The shape of an individual image in EuroSAT RGB is 64x64x3, which on reshaping gives a 12288 elements list. Hence, the number of input nodes is set to 12288.
num_outputs	10	This parameter specifies the number of output nodes in the network. As there are ten different classes in EuroSAT RGB among which neural network has to categorize an image, so for this study, it is set to 10.

Data Splitting and Evaluation in NEAT– Training, Testing, and Fitness Threshold:

As mentioned earlier that the EuroSAT RGB dataset was reduced into a compact, balanced dataset of 1000 images using random sampling and then split into 70% and 30% for training and testing sets using stratified sampling. This means that a single batch of 700 images will be utilized for training and another batch of 300 images will be employed for testing. Therefore, the fitness threshold is 700 for the training and 300 for testing dataset and their details are shown in Table 3 and Table 4.

Experiment 1– The Effectiveness of Training Accuracy in the NEAT Evolved Neural Networks:

With the help of training dataset, experiment 1 was performed to assess how various parameters impact the training accuracy of the NEAT evolved Neural Networks for LULC classification using EuroSAT RGB dataset. Table 3 presents a summary of the results from seven experimental models (M1-M7). The variations in population size (5 and 10), hidden layers (1, 2, 3), and activation functions (sigmoid and relu) were observed for the fixed number of generations, which represents the number of times the NEAT algorithm will repeat the process of selection, reproduction and mutation of individuals in a given population to find the best solution. It is a stopping criterion for the NEAT algorithm and was set to 25 in this study. This number has been selected after extensive experimentation as the NEAT algorithm can converge to a good solution before reaching the maximum number of generations. For four different runs of the experiment, training accuracy for all models were obtained.

Table 3 shows that the maximum and average training accuracy achieved was 100% by model M5. The best result was obtained using sigmoid as an activation function for a population size of ten and neural networks with two hidden layers. Rectified Linear Unit (RELU) as an activation function resulted in a drastically poor accuracy as shown by model M7 contrary to the CNNs that give the best results on RELU instead of a sigmoid. It can be inferred that the NEAT algorithm has the potential to generate efficient neural networks for classifying satellite images.

Experiment 2– The Effectiveness of Testing Accuracy in the NEAT Evolved Neural Networks:

With the help of testing dataset, experiment 2 was performed to assess how various parameters impact the testing accuracy of the NEAT evolved Neural Networks for LULC classification using EuroSAT RGB dataset. For each model, four runs were performed and testing accuracy was reported. The results of seven experimental models (M1-M7) are summarized in Table 4. Testing accuracy is a measure of how well the network can correctly predict the output from a set of testing dataset inputs. The model designated as M5, which employed a population size of 10, two hidden layers, and the sigmoid activation function depicted the highest average (99.83%) testing accuracy.

Experiment 3– Computational Time Comparison of the NEAT Evolved Neural Networks:

In terms of both training accuracy and testing accuracy, excellent results were achieved for the following two models:

- Best model (M5): The population size of 10 with two hidden layers and an activation function of sigmoid achieved the highest average training accuracy of 100% and the highest average testing accuracy of 99.83%. The overall accuracy is 99.91%.
- Runner-up model (M2): The population size of 5 with two hidden layers and an activation function of sigmoid achieved the second highest average training accuracy of 99.49% and the second highest average testing accuracy of 99.42%. The Overall accuracy for this case is 99.45%.

Table 3: Training Accuracy of NEAT evolved Neural Network Architectures for LULC Classification using the EuroSAT RGB Dataset.

Mod. No.	Pop Size	Fitness Thresh.	Act. Function	Hidden Layers	Training Accuracy				Avg. Training Accuracy
					Run1	Run2	Run3	Run4	
M1	5	700	Sigmoid	1	99.43	100	96.86	99.57	98.96
M2	5	700	Sigmoid	2	99.57	99.57	99.71	99.14	99.49
M3	5	700	Sigmoid	3	99.29	90.00	99.86	96.86	96.50
M4	10	700	Sigmoid	1	99.86	100	100	100	99.96
M5	**10**	**700**	**Sigmoid**	**2**	**100**	**100**	**100**	**100**	**100**
M6	10	700	Sigmoid	3	90.00	100	100	100	97.50
M7	5	700	Relu	1	19	17.57	16.71	33.14	21.60

Table 4: Testing Accuracy of NEAT evolved Neural Network Architectures for LULC Classification using the EuroSAT RGB Dataset.

Mod. No.	Pop Size	Fitness Thresh.	Act. Function	Hidden Layers	Testing Accuracy				Avg. Testing Accuracy
					Run1	Run2	Run3	Run4	
M1	5	300	Sigmoid	1	98.66	100	96.67	99.00	98.58
M2	5	300	Sigmoid	2	99.67	99.67	100	98.33	99.42
M3	5	300	Sigmoid	3	98.00	90.00	100	96.00	96.00
M4	10	300	Sigmoid	1	99.67	99.33	100	99.33	99.58
M5	**10**	**300**	**Sigmoid**	**2**	**100**	**100**	**99.33**	**100**	**99.83**
M6	10	300	Sigmoid	3	90.00	100	100	100	97.50
M7	5	300	Relu	1	19	18	16.64	27.67	20.33

Time Per Generation:

It refers to the time taken by the NEAT algorithm to complete one generation of training on the dataset. As an example, in the first run of the best configuration (M5), the time taken for the first generation was 256.78 seconds, while in the first run of the runner-up configuration (M2) the time taken for the first generation was 130.21 seconds. The reason for the runner-up configuration's shorter time is its smaller population size, which includes 5 networks to train, as opposed to the best configuration's 10 networks to train.

Average Time Per Run:

It refers to the mean duration required for the NEAT algorithm to finish all generations of training on the dataset for one run, calculated by taking the average of all runs. For best configuration (M5), the average time of 1st, 2nd, 3rd, and 4th runs were 203.23, 144.92, 222.49, and 116.14 seconds, respectively. Thus, the average time per run was 171.69 seconds. For runner-up configuration (M2), the average time of 1st, 2nd, 3rd, and 4th runs were 160.68, 139.97, 130.63, and 130.16 seconds, respectively. The average time per run was 140.38 seconds.

Total Training Time Per Run:

It refers to the time it takes for the NEAT algorithm to complete all generations of training on the dataset for one run. It is shown in Table 5 both in seconds and in hours. For best configuration (M5), 1st, 2nd, 3rd, and 4th runs took one hour 41 seconds, one hour, one hour 54 minutes, and 48 minutes, respectively. For runner-up configuration (M2), 1st, 2nd, 3rd, and 4th runs took one hour 12 minutes, 58 minutes, and 54 minutes, respectively. This showed that the best configuration (M5) on average took slightly more time than the runner-up configuration (M2).

Table 5: Time Analysis of the Best and Runner-up NEAT evolved Neural Network Models for LULC Classification using the EuroSAT RGB Dataset.

Gen.	**Best Configuration (M5)**				**Runner-up Configuration (M2)**			
	Time per gen. (Run1)	**Time per gen. (Run2)**	**Time per gen. (Run3)**	**Time per gen. (Run4)**	**Time per gen. (Run1)**	**Time per gen. (Run2)**	**Time per gen. (Run3)**	**Time per gen. (Run4)**
1	256.78	**277.45**	**273.81**	296.55	130.21	134.39	134.38	141.43
2	248.47	256.85	260.64	277.05	128.94	129.02	131.46	142.79
3	250.97	261.06	262.81	280.28	131.71	133.04	128.91	138.39
4	244.19	255.34	250.15	**271.37**	166.76	132.36	129.11	136.41
5	237.91	245.11	252.41	295.00	135.70	131.97	126.89	134.58
6	242.80	**242.36**	232.71	311.01	135.81	127.24	159.83	138.82
7	252.41	258.09	248.97	290.54	137.81	118.55	131.08	134.31
8	240.86	252.26	**226.59**	280.65	130.61	122.65	128.79	136.28
9	**257.62**	267.82	267.72	**319.16**	171.66	118.88	126.27	**154.22**
10	244.33	266.74	227.53	281.94	129.78	119.95	130.33	133.10
11	249.86	256.84	240.41		168.67	120.42	128.28	127.41
12	241.69	265.80	235.09		129.53	119.35	126.76	137.75
13	248.01	255.12	234.55		171.29	120.47	127.15	124.93
14	246.01	262.15	231.73		136.73	**494.52**	127.03	131.26
15	229.70		239.49		164.22	122.72	163.24	125.65
16	256.32		235.81		143.62	121.57	122.99	125.18
17	230.25		237.89		274.47	120.52	116.22	**114.82**
18	237.46		233.01		142.47	117.78	**205.44**	116.14
19	224.50		236.59		133.29	119.78	121.06	117.66
20	**211.84**		231.46		139.19	**117.60**	116.97	122.94
21	228.81		237.44		**490.69**	168.54	125.19	120.16
22			234.87		**122.38**	118.99	117.17	125.50
23			230.59		135.17	124.98	**108.34**	121.64
24					125.39	118.69	117.76	126.88
25					140.79	125.19	115.12	125.78
Avg. Time (s):	203.23	144.92	**222.49**	**116.14**	**160.68**	139.97	130.63	**130.16**
Tot. Time (s):	5080.86	3623.04	5562.33	2903.57	4016.94	3499.21	3265.75	3254.02
Tot. Time (hr.):	1.41	1.00	**1.54**	**0.81**	**1.12**	0.97	0.91	**0.90**

Discussion:

Analysis of LULC-NEAT Performance Based on Accuracy:

In essence, Table 3 from experiment 1 and Table 4 from experiment 2 demonstrates that utilizing sigmoid as the activation function and increasing the number of hidden layers from one to two has increased the training and testing accuracy. Using relu as the activation function, on the other hand, decreased the training and testing accuracy. The training and testing accuracy improved when the population size was increased from 5 to 10.

Analysis of LULC-NEAT Performance Based on Solution Convergence:

From the results of experiment 3 as shown in Table 5, it is evident that the NEAT algorithm can converge to a solution in fewer generations than the set number of generations. The M5 best configuration achieves this convergence in an average of 17 generations, likely due to its larger population size of 10. In contrast, the runner-up M2 configuration requires an average of 25 generations to converge, which can be attributed to its smaller population size of 5.

Analysis of LULC-NEAT Performance Based on Overall Accuracy and Average Computational Time:

Table 6 shows the overall accuracy and computation time for LULC-NEAT in terms of hyperparameters such as population size, activation function, and number of hidden layers. Both models have high overall accuracy, but the second model with a population size of 10 has slightly better testing and training accuracy. However, it takes more time to train. Depending on the specific use and resources available, one model may be more suitable than the other.

Table 6: LULC-NEAT Overall Accuracy and Average Computation Time.

Population Size	Activation Function	Hidden Layers	Average Training Accuracy	Average Testing Accuracy	Overall Accuracy	Average Computational Time
5	sigmoid	2	99.49	99.42	99.45	58 m 30s
10	sigmoid	2	100	99.83	99.91	1 h 11 m 12 s

Comparison of LULC-NEAT with State-of-the-Art CNN Models:

Table 1 presented detail comparison of LULC classification studies on the EuroSAT RGB dataset. The results indicated that the GoogleNet with preprocessing by Yadav [24] and the Wide ResNet-50 with data augmentation by Naushad [14] were the most successful among all the compared state-of-the-art CNN methods. LULC-NEAT is compared against both in terms of overall accuracy and total time as indicated in Table 7.

Table 7: Comparison LULC-NEAT with State-of-the-art CNN Model.

Authors	Number of Hidden Layers	Model	Overall Accuracy	Total Time
Naushad et al. [14]	>2	Wide ResNet-50 (With Data Augmentation)	99.17%	2 h 7 min 53 s
Yadav et al. [24]	22	GoogleNet (With Preprocessing)	99.68%	-
This study	2	LULC-NEAT (Population Size =5)	99.45%	58 m 30s
		LULC-NEAT (Population Size =10)	99.91%	1 h 11 m 12 s

The overall accuracy of GoogleNet leveraging LAB color model during the pre-processing stage as proposed by Yadav [24] is 99.68%, while the overall accuracy of the Wide ResNet-50 model, as proposed by Naushad [14] is 99.17%, with a total time of 2 hours, 7 minutes, and 53 seconds. The LULC-NEAT model with a population size of 5 achieved an overall accuracy of 99.45%, with a total time of 58 minutes and 30 seconds. Similarly, the LULC-NEAT model with a population size of 10 achieved an overall accuracy of 99.91%, with a total time of 1 hour 11 minutes, and 12 seconds. It is important to note that both LULC-NEAT models are simple models with only two hidden layers than the GoogleNet and Wide ResNet-50 CNN models. The LULC-NEAT models improve prediction from the GoogleNet and Wide

ResNet-50 CNN models because it is less complex with fewer parameters to learn. The generalization and training times are also improved as shown in Table 7. Additionally, a model with fewer layers may not overfit the training data as easily as a model with more layers. In some cases, a simpler model can be more effective in capturing the underlying patterns in the data, as it is less likely to be influenced by noise or irrelevant features. Another reason could be that the model with fewer layers is well-suited for the specific task, dataset, and computational resources available. The same reasoning was validated by Yadav et al. [24], who found that increasing the number of layers in a CNN does not always lead to improved results for a medium-sized dataset. In their study, they reported that GoogleNet, a 22-layer CNN, outperformed the 50-layer ResNet50 and the 101-layer ResNet101 in terms of both speed and accuracy.

Comparison of LULC-NEAT with a Contemporary e-NEAT:

In 2022, Guilherme [46] proposed the e-NEAT framework, which introduced a novel approach to tropical forest deforestation detection using pattern classifiers based on the NEAT. The framework utilized artificial neural networks and an aggregation method to enhance the classification results for Landsat-8 satellite images with minimal cloud cover captured near July 31, 2017. The e-NEAT achieved a balanced accuracy score of over 90% using a limited and reduced training set. The dataset was limited by using a reduced training scenario, meaning that each learning technique had only 91 segments to train the model, and 57,646 segments were classified on the test set. The mask used for image seg-mentation excluded all non-forest areas before August 1st, 2016.

The balanced accuracy measure in the e-NEAT considers both false positives and false negatives and gives equal weightage to each type of error, making it an unbiased evaluation method. This approach encourages diversity among the base classifiers, resulting in a better performance of the classification models compared to those using single or multiple classifiers. However, the e-NEAT has two limitations. Firstly, it only focuses on detecting newly deforested regions and does not account for other types of land cover changes such as pasture, herbaceous vegetation, annual and permanent crops, industrial and residential buildings, highway, sea and lakes, rivers, etc. Secondly, the e-NEAT framework was tested with a restricted training set, which may limit its generalizability to other datasets or scenarios.

In contrast to e-NEAT, the LULC-NEAT provides a novel solution to the first limitation by considering diversified land covers and demonstrates good performance. However, since the purpose of this study was to show the potential of NEAT for solving LULC classification, it employed only the balanced EuroSAT RGB dataset. In the future, the LULC-NEAT will be improved for the multispectral dataset instead of the RGB dataset and most importantly could be implemented for real-world Sentinel-2 Satellite images of a specific region with sufficient training and testing datasets. Additionally, performance metrics such as precision, recall, f1-score, confusion matrix, and overall accuracy will be used. Also, in the future, the power of graphics processing units (GPUs) and DL will be employed for LULC-NEAT to achieve better results with respect to both accuracy and speed.

Conclusion:

In this study, NEAT was implemented to solve the LULC classification problem. The results show that the NEAT algorithm has the potential to produce efficient results for satellite images. The proposed algorithm performed extremely well. The results of the experiment using NEAT evolved FFNN with two hidden layers on a balanced dataset showed a high training accuracy of 100% and a good testing accuracy of 99.83%. This indicates that the algorithm has learned the features of the training data very well and has the ability to generalize to new unseen data. The comparison with state-of-the-art CNN models demonstrated the robustness of the proposed approach. However, the concern of overfitting due to a training accuracy of 100% should be monitored in future experiments. Additionally, the effectiveness of the proposed technique will be further evaluated using multispectral datasets, real-world satellite images, and temporal datasets by utilizing the computational power of GPU-based implementation of NEAT. Performance metrics such as precision, recall, F1-score, and confusion matrix, in addition to overall accuracy, will be employed to quantify the performance of the proposed approach in comparison to existing state-of-the-art DL models. Applying NEAT to LULC classification contributes to advancing AI techniques in remote sensing and GIS, which is likely to inspire new methods and applications. The success of NEAT in LULC classification may only encourage its applications in other interdisciplinary fields of environmental monitoring, urban planning, agriculture, or disaster management where such classification challenges are real.

Abbreviations: Abbreviations used in this article are listed below:

ADAM	Adaptive Moment Estimation
ANN	Artificial Neural Network
CI	Color Infrared
CNN	Convolutional Neural Network
CV	Computer Vision
DL	Deep Learning
DDRL-AM	Deep Discriminative Representation Learning with Attention Map
Fuzzy ARTMAP	Fuzzy Adaptive Resonance Theory-supervised Predictive Mapping
GLCM	Grey-Level Co-occurrence Matrix
GPU	Graphics Processing Unit
HOG	Histogram of Oriented Gradients
LULC	Land Use Land Cover
MD	Mahalanobis Distance
ML	Machine Learning
MS	Multispectral
NE	Neuroevolution
NEAT	NeuroEvolution of Augmenting Topologies
NLP	Natural Language Processing
PR	Pattern Recognition
ResNet-50	Residual Network 50
RF	Random Forest
RGB	Red, Green and Blue
SAM	Spectral Angle Mapper
SDG	Stochastic Gradient Descent
SVM	Support Vector Machine
SWIR	Short Wave Infrared
TL	Transfer Learning
VGG-16	Visual Geometry Group 16
VGG-19	Visual Geometry Group 19
Wide ResNet-50	Wide Residual Network 50
XGBoost	Extreme Gradient Boosting

Acknowledgement: This work is financially supported by the National Center of Big Data and Cloud Computer (NCBC), University of Engineering and Technology, Peshawar, under the auspices of the Higher Education Commission (HEC), Pakistan.

Author's Contribution: All authors have equally contributed towards the paper.

Conflict of Interest: The authors have no conflicts of interest to declare.

Project Details: NA

References:

[1] S. Talukdar, P. Singha, S. Mahato, Shahfahad, S. Pal, Y.-A. Liou, and A. Rahman, "Land-Use Land-Cover Classification by Machine Learning Classifiers for Satellite Observations—A Review," Remote Sensing, vol. 12, no. 7, p. 1135, Apr. 2020, doi: 10.3390/RS12071135.

[2] A. M. Abdi, "Land cover and land use classification performance of machine learning algorithms in a boreal landscape using Sentinel-2 data," GIScience Remote Sens., vol. 57, no. 1, pp. 1–20, Jan. 2020, doi: 10.1080/15481603.2019.1650447.

[3] K. Kaku, "Satellite remote sensing for disaster management support: A holistic and staged approach based on case studies in Sentinel Asia," Int. J. Disaster Risk Reduct., vol. 33, pp. 417–432, Feb. 2019, doi: 10.1016/J.IJDRR.2018.09.015.

[4] "5 crimes solved using Google Earth | The Week." Accessed: Jun. 30, 2024. [Online]. Available: https://theweek.com/articles/491975/5-crimes-solved-using-google-earth

[5] M. Majeed, A. Tariq, M. M. Anwar, A. M. Khan, F. Arshad, F. Mumtaz, M. Farhan, L. Zhang, A. Zafar, M. Aziz, S. Abbasi, G. Rahman, S. Hussain, M. Waheed, K. Fatima, and S. Shaukat, "Monitoring of Land Use–Land Cover Change and Potential Causal Factors of Climate Change in Jhelum District, Punjab, Pakistan, through GIS and Multi-Temporal Satellite Data," Land, vol. 10, no. 10, p. 1026, Sep. 2021, doi: 10.3390/LAND10101026.

[6] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," Nat. 2015 5217553, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[7] J. Schmidhuber, "Deep learning in neural networks: An overview," Neural Networks, vol. 61, pp. 85–117, Jan. 2015, doi: 10.1016/J.NEUNET.2014.09.003.

[8] Y. Bengio, "Learning Deep Architectures for AI," Found. Trends® Mach. Learn., vol. 2, no. 1, pp. 1–127, Nov. 2009, doi: 10.1561/2200000006.

[9] D. Floreano, P. Dürr, and C. Mattiussi, "Neuroevolution: From architectures to learning," Evol. Intell., vol. 1, no. 1, pp. 47–62, Jan. 2008, doi: 10.1007/S12065-007-0002-4/METRICS.

[10] K. O. Stanley and R. Miikkulainen, "Evolving Neural Networks Through Augmenting Topologies." Evolutionary Computation, vol. 10, pp. 99–127, Jun. 2002, doi: 10.1162/106365602320169811.

[11] "GitHub - phelber/EuroSAT: EuroSAT: Land Use and Land Cover Classification with Sentinel-2." Accessed: Jun. 23, 2024. [Online]. Available: https://github.com/phelber/EuroSAT

[12] P. Helber, B. Bischke, A. Dengel, and D. Borth, "EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification," IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens., vol. 12, no. 7, pp. 2217–2226, Jul. 2019, doi: 10.1109/JSTARS.2019.2918242.

[13] P. Helber, B. Bischke, A. Dengel, and D. Borth, "Introducing eurosat: A novel dataset and deep learning benchmark for land use and land cover classification," Int. Geosci. Remote Sens. Symp., pp. 204–207, Oct. 2018, doi: 10.1109/IGARSS.2018.8519248.

[14] R. Naushad, T. Kaur, and E. Ghaderpour, "Deep Transfer Learning for Land Use and Land Cover Classification: A Comparative Study," Sensors, vol. 21, no. 23, p. 8083, Dec. 2021, doi: 10.3390/S21238083.

[15] A. Loganathan, S. Koushmitha, and Y. N. K. Arun, "Land Use/Land Cover Classification Using Machine Learning and Deep Learning Algorithms for EuroSAT Dataset – A Review," Lect. Notes Networks Syst., vol. 418 LNNS, pp. 1363–1374, 2022, doi: 10.1007/978-3-030-96308-8_126.

[16] "Welcome to NEAT-Python's documentation! — NEAT-Python 0.92 documentation." Accessed: Jun. 30, 2024. [Online]. Available: https://neat-python.readthedocs.io/en/latest/

[17] J. Li, D. Lin, Y. Wang, G. Xu, Y. Zhang, C. Ding, and Y. Zhou, "Deep Discriminative Representation Learning with Attention Map for Scene Classification," Remote Sensing, vol. 12, no. 9, p. 1366, Apr. 2020, doi: 10.3390/RS12091366.

[18] G. Chen, X. Zhang, X. Tan, Y. Cheng, F. Dai, K. Zhu, Y. Gong, and Q. Wang, "Training Small Networks for Scene Classification of Remote Sensing Images via Knowledge Distillation," Remote Sensing, vol. 10, no. 5, p. 719, May 2018, doi: 10.3390/RS10050719.

[19] "Land Cover Classification with EuroSAT Dataset." Accessed: Jun. 30, 2024. [Online]. Available: https://www.kaggle.com/code/nilesh789/land-cover-classification-with-eurosat-dataset

[20] P. Jain, B. Schoen-Phelan, and R. Ross, "Self-Supervised Learning for Invariant Representations from Multi-Spectral and SAR Images," IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens., vol. 15, pp. 7797–7808, May 2022, doi: 10.1109/JSTARS.2022.3204888.

[21] A. Stateczny, S. M. Bolugallu, P. B. Divakarachari, K. Ganesan, and J. R. Muthu, "Multiplicative Long Short-Term Memory with Improved Mayfly Optimization for LULC Classification," Remote Sensing, vol. 14, no. 19, p. 4837, Sep. 2022, doi: 10.3390/RS14194837.

[22] M. Ç. Aksoy, B. Sirmacek, and C. Ünsalan, "Land classification in satellite images by injecting traditional features to CNN models," Remote Sens. Lett., vol. 14, no. 2, pp. 157–167, Feb. 2023, doi: 10.1080/2150704X.2023.2167057.

[23] A. Rangel, J. Terven, D. M. Cordova-Esparza, and E. A. Chavez-Urbiola, "Land Cover Image Classification," Jan. 2024, Accessed: Jun. 23, 2024. [Online]. Available: http://arxiv.org/abs/2401.09607

[24] D. Yadav, K. Kapoor, A. K. Yadav, M. Kumar, A. Jain, and J. Morato, "Satellite image classification using deep learning approach," Earth Sci. Informatics, vol. 17, no. 3, pp. 2495–2508, Jun. 2024, doi: 10.1007/S12145-024-01301-X/METRICS.

[25] D. Phiri, M. Simwanda, S. Salekin, V. R. Nyirenda, Y. Murayama, and M. Ranagalage, "Sentinel-2 Data for Land Cover/Use Mapping: A Review," Remote Sensing, vol. 12, no. 14, p. 2291, Jul. 2020, doi: 10.3390/RS12142291.

[26] "S2 Mission." Accessed: Jun. 30, 2024. [Online]. Available: https://sentiwiki.copernicus.eu/web/s2-mission

[27] "Artificial Neural Network (ANN) with Practical Implementation | by Amir Ali | The Art of Data Scicne | Medium." Accessed: Jun. 23, 2024. [Online]. Available: https://medium.com/machine-learning-researcher/artificial-neural-network-ann-4481fa33d85a

[28] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. E. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," Heliyon, vol. 4, no. 11, p. e00938, Nov. 2018, doi: 10.1016/J.HELIYON.2018.E00938.

[29] "Neural networks and back-propagation explained in a simple way | by Assaad MOAWAD | DataThings | Medium." Accessed: Jun. 23, 2024. [Online]. Available: https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e

[30] S. Ruder, "An overview of gradient descent optimization algorithms," Sep. 2016, Accessed: Jun. 23, 2024. [Online]. Available: http://arxiv.org/abs/1609.04747

[31] M. G. M. Abdolrasol, S. M. S. Hussain, T. S. Ustun, M. R. Sarker, M. A. Hannan, R. Mohamed, J. A. Ali, S. Mekhilef, and A. Milad, "Artificial Neural Networks Based Optimization Techniques: A Review," Electronics, vol. 10, no. 21, p. 2689, Nov. 2021, doi: 10.3390/ELECTRONICS10212689.

[32] J. N. D. Gupta and R. S. Sexton, "Comparing backpropagation with a genetic algorithm for neural network training," Omega, vol. 27, no. 6, pp. 679–684, Dec. 1999, doi: 10.1016/S0305-0483(99)00027-4.

[33] J. H. Holland, "Genetic Algorithms", Scientific American, vol. 267, no. 1, pp. 66–73, Jul. 1992, doi: 10.2307/24939139.

[34] "Introduction to Genetic Algorithms — Including Example Code | by Vijini Mallawaarachchi | Towards Data Science." Accessed: Jun. 30, 2024. [Online]. Available: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3

[35] K. O. Stanley and R. Miikkulainen, "Efficient Evolution of Neural Network Topologies". Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002 (Cat. No.02TH8600) 2002, doi: 10.1109/CEC.2002.1004508.

[36] K. O. Stanley, "Efficient Evolution of Neural Networks Through Complexification." 2004. Accessed: Jun. 30, 2024. [Online]. Available: https://nn.cs.utexas.edu/?stanley:phd2004

[37] "How I Built an Intelligent Agent to Play Flappy Bird | by Danny Zhu | Analytics Vidhya | Medium." Accessed: Jun. 23, 2024. [Online]. Available: https://medium.com/analytics-vidhya/how-i-built-an-ai-to-play-flappy-bird-81b672b66521

[38] Y. Bengio, "Practical Recommendations for Gradient-Based Training of Deep Architectures," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 7700 LECTURE NO, pp. 437–478, Jun. 2012, doi: 10.1007/978-3-642-35289-8_26.

[39] "Simple Guide to Hyperparameter Tuning in Neural Networks | by Matthew Stewart, PhD | Towards Data Science." Accessed: Jun. 23, 2024. [Online]. Available: https://towardsdatascience.com/simple-guide-to-hyperparameter-tuning-in-neural-networks-3fe03dad8594

[40] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaria, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," J. Big Data, vol. 8, no. 1, pp. 1–74, Mar. 2021, doi: 10.1186/S40537-021-00444-8.

[41] A. Vali, S. Comai, and M. Matteucci, "Deep Learning for Land Use and Land Cover Classification Based on Hyperspectral and Multispectral Earth Observation Data: A Review," Remote Sensing, vol. 12, no. 15, p. 2495, Aug. 2020, doi: 10.3390/RS12152495.

[42] "StratifiedShuffleSplit — scikit-learn 1.5.0 documentation." Accessed: Jun. 30, 2024. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html

[43] "Built-in Functions — Python 3.12.4 documentation." Accessed: Jun. 30, 2024. [Online]. Available: https://docs.python.org/3/library/functions.html#next

[44] "Image Data Generators in Keras. How to effectively and efficiently use… | by Manpreet Singh Minhas | Towards Data Science." Accessed: Jun. 30, 2024. [Online]. Available: https://towardsdatascience.com/image-data-generators-in-keras-7c5fc6928400

[45] A. M. Fred Agarap, "Deep Learning using Rectified Linear Units (ReLU)," Mar. 2018, Accessed: Jun. 23, 2024. [Online]. Available: https://arxiv.org/abs/1803.08375v2

[46] A. P. Guilherme, B. J. R. D. Fernanda, A. Fazenda, and F. A. Faria, "Neuroevolution-based Classifiers for Deforestation Detection in Tropical Forests," Proc. - 2022 35th Conf. Graph. Patterns, Images, SIBGRAPI 2022, vol. 1, pp. 13–18, Oct. 2022, doi: 10.1109/SIBGRAPI55357.2022.9991798.

Copyright © by authors and 50Sea. This work is licensed under Creative Commons Attribution 4.0 International License.