

Design of a Deep Learning Model Based on Convolutional Neural Networks (CNN) for Land-Cover Object Recognition in Remote Sensing Imagery

Vu Ngoc Phan

Hanoi University of Natural Resources and Environment, Hanoi, Vietnam.

ABSTRACT: Recognising land-cover objects in high-resolution satellite imagery is a basic step for resource monitoring and land-use planning, yet the design choices behind compact convolutional neural networks (CNN) for local data are seldom reported in detail. This paper presents the design, training and deployment of a patch-based CNN for recognising 11 land-cover classes in PlanetScope imagery of a peri-urban area in Bac Ninh province, Vietnam. The model takes 64×64 pixel patches with six channels (four spectral bands, NDVI and NDWI) and consists of four convolution blocks with batch normalisation, followed by a 128-neuron dense layer with dropout; it has 917,164 parameters and is trained from scratch. A reference dataset was built by manual digitisation, with training and test polygons separated spatially. On 12,995 spatially independent test patches, the model (personal computer, CPU, official run) reached an overall accuracy of 90.08% and a kappa coefficient of 0.8909 (11 classes), and 90.37% (0.8689) after merging into five groups. Under identical hardware (Google Colab, GPU), the final design raised overall accuracy from 85.10% to 92.78% compared with the initial three-block design. Shape-dependent classes benefited most: residential area and industrial zone reached F1-scores of 1.000 and 0.999, and road 0.854. A sliding-window step of 8 pixels produced smoother maps than a step of 32 at about ten times the inference cost. The study also shows that a randomly split validation set neutralises early stopping, and that fixed random seeds do not guarantee identical results across hardware.

KEY WORDS: Convolutional neural network, Land cover, PlanetScope, Remote sensing, Sliding window, Spatial leakage

Date of Submission: 12-10-2026

Date of acceptance: 24-10-2026

I. INTRODUCTION

Land-cover maps are a basic input for natural resource monitoring, land-use planning and environmental modelling. In rapidly urbanising areas, where rice fields, settlements, industrial zones and road networks are interwoven within a few square kilometres, such maps need to be detailed and frequently updated. Very-high-resolution satellite constellations such as PlanetScope, with pixels of a few metres and near-daily revisit, make this technically possible, provided that an automatic method can reliably recognise the land-cover objects on the image.

Convolutional neural networks (CNN) have become the dominant tool for this task. Since LeNet-5 [1] and the breakthrough of AlexNet [2], deeper architectures built from small convolution kernels, such as VGG [3], and residual networks [4] have steadily improved image recognition. Reviews of deep learning in remote sensing show that land-cover and land-use classification is among the most studied applications and that CNN are the most widely used architecture [5–7]. Unlike pixel-based classifiers, a CNN learns spectral and spatial features jointly from the neighbourhood of each pixel, so it can distinguish objects that have similar reflectance but different shapes or textures [8].

In Vietnam, large networks pre-trained on European benchmark data showed clear limitations when transferred to the north-eastern region, because the training data did not represent local land-cover characteristics [9]. By contrast, networks trained on domestic data achieved good results for national land-cover mapping [10], flood mapping [11] and mangrove species classification using Planet imagery [12]. These studies, however, mostly report final accuracies; the design decisions that lead to a working model, such as the input channels, depth, regularisation, sampling strategy, whole-scene inference and reproducibility, are rarely

documented, even though they largely determine whether a model can be reused by others. In addition, the evaluation protocol matters: when validation or test samples lie close to training samples, accuracy can be over-estimated by 8–14% [13], and by up to 28% for CNN [14].

In this paper, “object recognition” refers to assigning each image location to one of the land-cover object classes present in the scene (rivers, ponds, rice paddies, residential blocks, industrial buildings, roads, etc.), using the image patch centred on that location. The paper documents the full design of a compact CNN for this purpose and the evidence behind each design choice.

1.1 Study Area and Data

The study area is a subset of a PlanetScope scene, about $12.2 \text{ km} \times 14.6 \text{ km}$ ($\approx 178 \text{ km}^2$) in Bac Ninh province, northern Vietnam, centred on Dao Vien commune and covering parts of Que Vo, Phu Lang, Dai Lai, Nhan Thang and Gia Binh (Fig. 1). The area lies near the confluence of the Cau, Duong and Thai Binh rivers and is a typical peri-urban landscape in which rivers, ponds, rice paddies, villages, industrial zones, roads and bare land are closely mixed. Several of these classes have overlapping spectral signatures, which makes the area a demanding test for a recognition model.

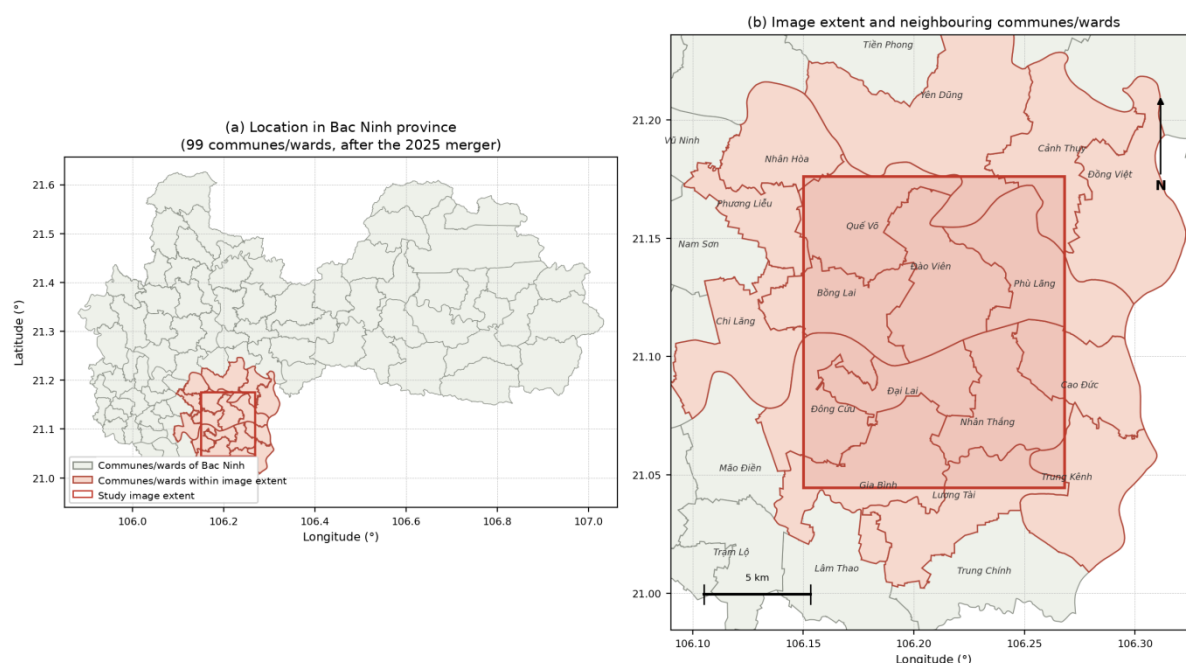


Figure 1: Location of the study area: (a) within Bac Ninh province; (b) image extent and neighbouring communes/wards

The image is a subset of a four-band PlanetScope Ortho Tile scene (sensor generation PS2.SD) with Blue (464–517 nm), Green (547–585 nm), Red (650–682 nm) and near-infrared (NIR, 846–888 nm) bands, a ground sampling distance of about 3.7–4.2 m and a pixel size of 3.125 m [15]. The scene was acquired in August 2020. The subset used measures $3,890 \times 4,687$ pixels and is projected in UTM zone 48N, WGS-84 (EPSG:32648).

1.2 Problem Statements

The work addresses four design questions: (i) how to design a compact CNN that can be trained from scratch on a limited, locally digitised reference dataset; (ii) how much each round of design improvement contributes to recognition accuracy when measured on spatially independent data; (iii) how the whole-scene inference strategy affects map quality and computation time; and (iv) whether the trained model can be reproduced when the software and hardware environment changes.

II. MATERIAL AND METHODS

2.1 Reference Dataset with Spatially Separated Sampling

No suitable labelled dataset existed for the area, so a reference dataset was built by manual digitisation of polygons in ArcGIS on the Planet image, with the help of higher-resolution imagery and local knowledge. Each polygon stores the class code, its area in pixels and a *split* attribute (“train” or “test”) assigned at digitisation time. Because the split is made per polygon rather than per pixel, the training and test samples lie in

geographically separate places (Fig. 2), which prevents spatial leakage caused by the strong spatial autocorrelation of remote sensing images [13, 14]. The polygons were rasterised on the pixel grid of the image into a training mask and a test mask. The dataset contains 11 classes, which are also merged into five thematic groups for map use (Table I); the merging rule was fixed before training.

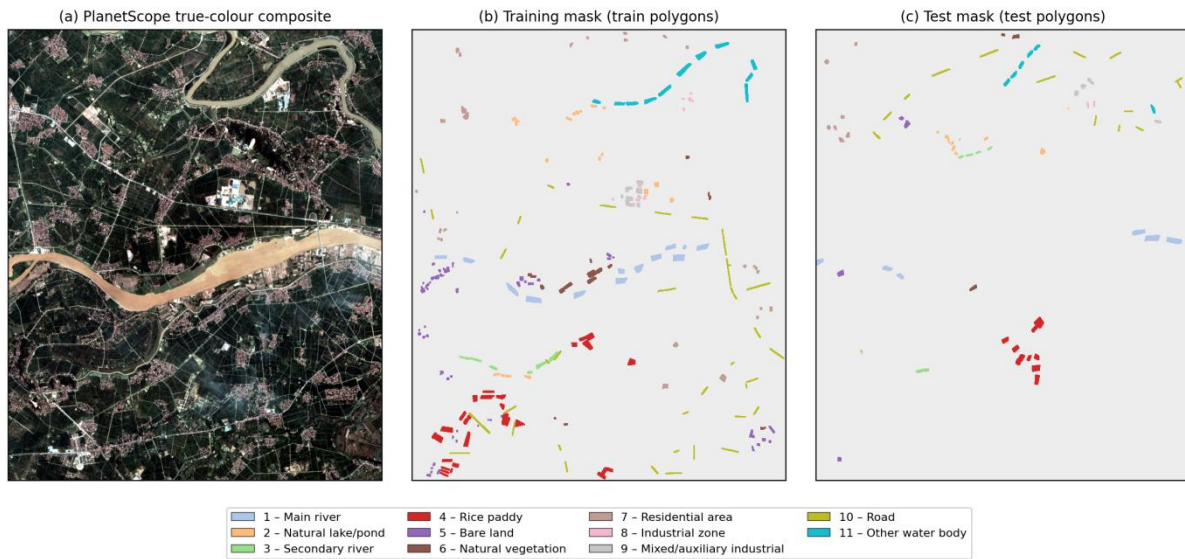


Figure 2: PlanetScope true-colour composite (a) and the training (b) and test (c) masks; small polygons are enlarged for visibility

Table I. Land-cover classes, thematic groups and sample sizes

Class	Group (5 groups)	Labelled pixels (train)	Labelled pixels (test)	Patches (train)	Patches (test)
1. Main river	Water	46,688	21,433	1,200	1,156
2. Natural lake/pond	Water	8,968	3,185	1,200	1,200
3. Secondary river	Water	7,213	3,513	1,200	1,200
4. Rice paddy	Vegetation	53,924	22,158	1,198	1,200
5. Bare land	Bare land	13,469	4,885	1,200	1,200
6. Natural vegetation	Vegetation	13,373	2,746	1,200	1,200
7. Residential area	Built-up and industry	15,361	4,435	1,199	1,200
8. Industrial zone	Built-up and industry	2,856	1,039	1,200	1,039
9. Mixed/auxiliary industrial	Built-up and industry	10,702	3,872	1,200	1,200
10. Road	Road	12,861	4,377	1,200	1,200
11. Other water body	Water	29,127	6,903	1,200	1,200
Total		214,542	78,546	13,197	12,995

2.2 Input Design

Channels. In addition to the four spectral bands, two indices that emphasise vegetation and water were added: NDVI [16] and NDWI [17]. They are computed from the original reflectance values (a constant of 10^{-8} is added to the denominators) and rescaled from $[-1, 1]$ to $[0, 1]$ by $(x + 1)/2$. Each spectral band X is normalised independently with a 2–98% percentile stretch, which limits the influence of saturated pixels and sensor noise:

$$NDVI = \frac{NIR - Red}{NIR + Red}, \quad NDWI = \frac{Green - NIR}{Green + NIR} \dots\dots\dots(1)$$

$$X' = clip\left(\frac{X - P_2}{P_{98} - P_2}, 0, 1\right) \dots\dots\dots(2)$$

Patch size and sampling. The input unit is a $64 \times 64 \times 6$ patch centred on a labelled pixel; at 3.125 m this is a window of $200 \text{ m} \times 200 \text{ m}$, wide enough to capture the linear shape of roads and the block structure of settlements. For each class, up to 1,200 labelled positions were drawn at random without replacement from each mask, and patches whose window crossed the image border were discarded. Capping the number of samples balances classes whose labelled areas differ by a factor of almost 20 (Table I). The result is 13,197 training and 12,995 test patches (Fig. 3). A stratified 20% of the training patches (2,640) was held out as a validation set to monitor training, leaving 10,557 patches for weight updates; the validation set was shuffled before splitting because patches are extracted class by class. The test set was used only for the final evaluation.



Figure 3: Example 64×64 training patches (true-colour display) for the 11 land-cover classes

2.3 CNN Architecture Design

The network (Fig. 4, Table II) follows the VGG principle of stacking small 3×3 kernels [3]. Each convolution layer computes feature maps by sliding learnable kernels over its input,

$$Y(i, j) = \sum_m \sum_n X(i + m, j + n) \cdot K(m, n) + b \dots \dots \dots (3)$$

where X is the input, K the kernel weights and b the bias; a ReLU activation, $\max(0, x)$, follows each convolution. The design choices were as follows. (i) *Depth and width*: four convolution blocks with 32, 64, 128 and 256 filters, doubling the number of filters as the spatial size shrinks; “valid” padding reduces the 64×64 input to 4×4 after the fourth block. (ii) *Batch normalisation* [18] after every convolution stabilises training and speeds convergence. (iii) *Pooling*: 2×2 max pooling after the first three blocks only, so that the last block keeps a 4×4 map of high-level context features. (iv) *Classifier*: the 4,096 features are passed to a 128-neuron dense layer with ReLU and a dropout rate of 0.5 [19] to limit over-fitting, and to a softmax output layer:

$$\text{softmax}(z_k) = \frac{e^{z_k}}{\sum_j e^{z_j}} \dots \dots \dots (4)$$

The output layer has 12 units because the label scheme reserves code 12; that class has no samples, so the model effectively learns and is evaluated on 11 classes. The model has 917,164 parameters, of which 916,204 are trainable, and is saved as a 10.6 MB Keras file, which makes it easy to store and redeploy.

(a) Feature-extraction part

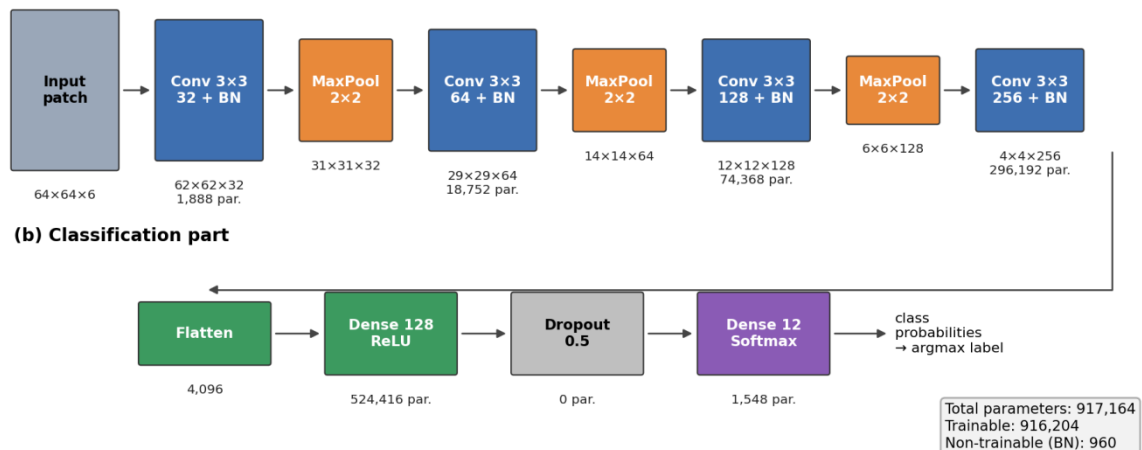


Figure 4: Architecture of the proposed CNN: (a) feature-extraction part; (b) classification part

Table II. Layer-by-layer summary of the proposed CNN (input $64 \times 64 \times 6$)

Layer (block)	Output shape	Parameters
Conv2D 3×3, 32 + BatchNorm (1)	$62 \times 62 \times 32$	1,888
MaxPooling 2×2 (1)	$31 \times 31 \times 32$	0
Conv2D 3×3, 64 + BatchNorm (2)	$29 \times 29 \times 64$	18,752
MaxPooling 2×2 (2)	$14 \times 14 \times 64$	0
Conv2D 3×3, 128 + BatchNorm (3)	$12 \times 12 \times 128$	74,368
MaxPooling 2×2 (3)	$6 \times 6 \times 128$	0
Conv2D 3×3, 256 + BatchNorm (4)	$4 \times 4 \times 256$	296,192
Flatten	4,096	0
Dense 128, ReLU	128	524,416
Dropout 0.5	128	0
Dense 12, Softmax	12	1,548
Total (trainable / non-trainable)		917,164 (916,204 / 960)

2.4 Training Strategy

The network was trained with the Adam optimiser [20] (Keras defaults) to minimise the categorical cross-entropy between the one-hot label y and the predicted probabilities $\hat{y}$ [21]:

$$L = - \sum_k y_k \cdot \log \hat{y}_k \dots\dots\dots(5)$$

The training configuration is summarised in Table III. Two callbacks monitor the validation loss (val_loss): early stopping restores the best weights when val_loss has not improved for 8 epochs, and ReduceLROnPlateau halves the learning rate after 3 epochs without improvement. Data augmentation is applied on the fly to each training batch with random horizontal and vertical flips and random rotations by 0° , 90° , 180° or 270° [22]. These geometric transforms do not change the label because nadir-looking images have no preferred orientation; radiometric transforms (brightness, contrast) were deliberately excluded to preserve the physical meaning of the spectral channels and indices. For reproducibility, a single seed (42) was fixed for Python, NumPy and TensorFlow, PYTHONHASHSEED was set, and TensorFlow's op-determinism mode was enabled [23].

Table III. Training configuration

Item	Setting	Purpose
Optimiser / loss	Adam / categorical cross-entropy	Adaptive learning-rate optimisation
Batch size / max. epochs	64 / 40	Upper bound; actual length set by callbacks
Early stopping	val_loss, patience 8, restore best weights	Stop when validation loss stops improving
ReduceLROnPlateau	val_loss, factor 0.5, patience 3, min. lr 10^{-6}	Refine learning rate on plateaus
Augmentation	Random flips, rotations by multiples of 90°	Increase sample diversity
Reproducibility	Seed 42; op-determinism enabled	Repeatable runs on the same system

2.5 Whole-Scene Inference with a Sliding Window

Because the network accepts fixed-size patches, the whole scene is classified with a sliding window. Window centres are placed on a regular grid with spacing s (step_size); each window is predicted and its label is assigned to the $s \times s$ block around the centre. Predictions are batched for efficiency. For an image of $H \times W$ pixels and patch size 64, the number of predictions is approximately

$$N \approx \frac{(H - 64)(W - 64)}{s^2} \dots\dots\dots(6)$$

so reducing s by a factor of four increases the work by a factor of about 16. The default $s = 32$ (half the patch size) is fast but produces blocky boundaries; $s = 8$ was adopted as the final setting. A 32-pixel border, half the patch size, cannot be classified.

2.6 Accuracy Assessment and Experimental Environment

Accuracy was assessed on the spatially independent test set with the confusion matrix [24], overall accuracy (OA), Cohen's kappa coefficient [25] and, for each class, precision (P), recall (R) and F1-score:

$$OA = \frac{\sum_{i=1}^n x_{ii}}{N}, \quad \kappa = \frac{p_o - p_e}{1 - p_e} \dots\dots\dots(7)$$

$$P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN}, \quad F1 = \frac{2 \cdot P \cdot R}{P + R} \dots\dots\dots(8)$$

where x_{ii} is the number of correctly classified samples of class i , n the number of classes, N the number of test samples, p_o the observed agreement and p_e the chance agreement derived from the row and column totals. Metrics are reported for the 11 classes and for the five merged groups. Development runs were carried out on Google Colab (free tier, GPU); the official run was executed on a personal computer (CPU) with Python 3.12,

TensorFlow and scikit-learn [26], after the Colab session repeatedly ran out of memory when the full workflow was executed in one session.

III. RESULTS AND DISCUSSIONS

3.1 Training Behaviour and the Role of the Validation Split

Fig. 5 shows the learning curves. Training accuracy was about 0.91 after the first epoch and approached 1.0 within a few epochs. The validation curves fluctuated strongly during the first 8–10 epochs: the loss started near 15, dropped close to zero and jumped above 7 in the sixth epoch, while validation accuracy varied between about 0.26 and 0.97. Afterwards, validation accuracy saturated at about 1.0 and the validation loss stayed close to zero until the 40th epoch.

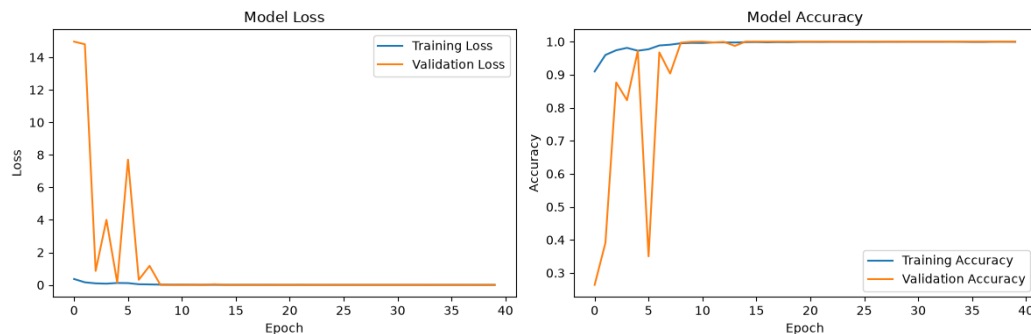


Figure 5: Training and validation loss (left) and accuracy (right) over 40 epochs

A validation accuracy of about 1.0 is roughly ten percentage points higher than the 90.08% (CPU, official run) measured on the test set (Section 3.3). The gap does not reflect the true generalisation ability of the model; it results from the way the validation set was created. Validation patches were split at random from the training patches, and patches cut around adjacent pixels of the same polygon overlap each other, so the network was “validated” on image content it had essentially seen. This is the spatial leakage described by [13] and [14]. The practical consequence for the design is that the two callbacks driven by `val_loss` lose their supervisory role: once `val_loss` is nearly zero, early stopping has no signal to detect over-fitting, so training ran for all 40 epochs; learning-rate reductions can be triggered by tiny fluctuations rather than a genuine plateau; and the “best” weights are selected on an optimistic signal. This may also explain why the training time of the same configuration varied from 36.99 s in a preliminary run to 421.04 s in the official run.

The leakage, however, does not bias the reported accuracy. The test set was separated by polygons at digitisation time and never took part in any training decision, so all figures reported below measure performance on unseen locations. In other words, the validation split affects the *training process* (when to stop, when to lower the learning rate), not the validity of the *reported accuracy*. A spatially separated validation set is therefore recommended for future designs.

3.2 Effect of Design Improvements

The design evolved over several runs (Table IV). Runs 1–3 used the initial design (500 samples per class, three convolution blocks without batch normalisation or learning-rate scheduling) before the random seed was fixed; their OA ranged from 84.10% to 87.67%, a spread of 3.57 percentage points that can be attributed to uncontrolled randomness, because the settings that differed among these runs (the inference step and the addition of a baseline) do not affect the patch-level test evaluation. This motivated fixing the seed and enabling deterministic operations (run 4). Three improvements were then applied together: the number of samples per class was increased from 500 to 1,200; batch normalisation and a fourth convolution block with 256 filters were added; and ReduceLROnPlateau was introduced, with the early-stopping patience increased from 5 to 8 epochs to give it time to act.

Table IV. Evolution of the CNN design over the development runs (test set)

Run	Environment and configuration	OA 11 classes	Kappa 11 classes	OA 5 groups	Kappa 5 groups
1	Colab; initial design; step 32; seed not fixed	84.10%	0.8251	88.46%	0.8409
2	Colab; initial design; step 32; seed not fixed	87.67%	0.8644	88.82%	0.8461
3	Colab; initial design; step 8; seed not fixed	84.17%	0.8259	88.60%	0.8442
4	Colab (GPU); initial design; step 8; seed 42	85.10%	0.8361	88.04%	0.8365
5	Colab (GPU); final design; step 8; seed 42	92.78%	0.9206	93.14%	0.9071
6	Local PC (CPU); final design; step 8; seed 42 (official)	90.08%	0.8909	90.37%	0.8689

Initial design: 500 patches per class, 3 convolution blocks, no BatchNorm/ReduceLROnPlateau. Final design: 1,200 patches per class, 4 blocks with BatchNorm, ReduceLROnPlateau.

Comparing runs 4 and 5, which share the same hardware, code base and seed, the final design raised OA from 85.10% to 92.78% (+7.68 percentage points) and kappa from 0.8361 to 0.9206 for 11 classes, and OA from 88.04% to 93.14% for five groups. The larger gain at the 11-class level shows that the improvements mainly help to separate sub-classes within the same group, such as the two industrial classes and the different water bodies. Because the three improvements were applied jointly, their individual contributions cannot be separated; an ablation in which one factor is changed at a time is left for future work. The official figures used in the rest of the paper come from run 6 (personal computer, CPU), the only run in which the complete workflow finished on a single machine.

3.3 Recognition Performance by Class

On the 12,995 test patches, the official model (personal computer, CPU, run 6) reached OA = 90.08% and κ = 0.8909 for 11 classes, an “almost perfect” agreement on the scale of [27]. Per-class results are given in Table V and the confusion matrix in Fig. 6.

Table V. Per-class precision, recall and F1-score of the proposed CNN (11 classes, test set)

Class	Precision	Recall	F1-score	Test patches
1. Main river	1.0000	0.9680	0.9837	1,156
2. Natural lake/pond	0.5892	0.9850	0.7374	1,200
3. Secondary river	1.0000	0.9175	0.9570	1,200
4. Rice paddy	0.9876	0.9992	0.9934	1,200
5. Bare land	0.7182	0.7517	0.7345	1,200
6. Natural vegetation	1.0000	0.5558	0.7145	1,200
7. Residential area	1.0000	1.0000	1.0000	1,200
8. Industrial zone	1.0000	0.9981	0.9990	1,039
9. Mixed/auxiliary industrial	0.9805	0.9642	0.9723	1,200
10. Road	0.9364	0.7850	0.8540	1,200
11. Other water body	0.9926	1.0000	0.9963	1,200
Macro average	0.9277	0.9022	0.9038	12,995
Weighted average	0.9265	0.9008	0.9024	12,995

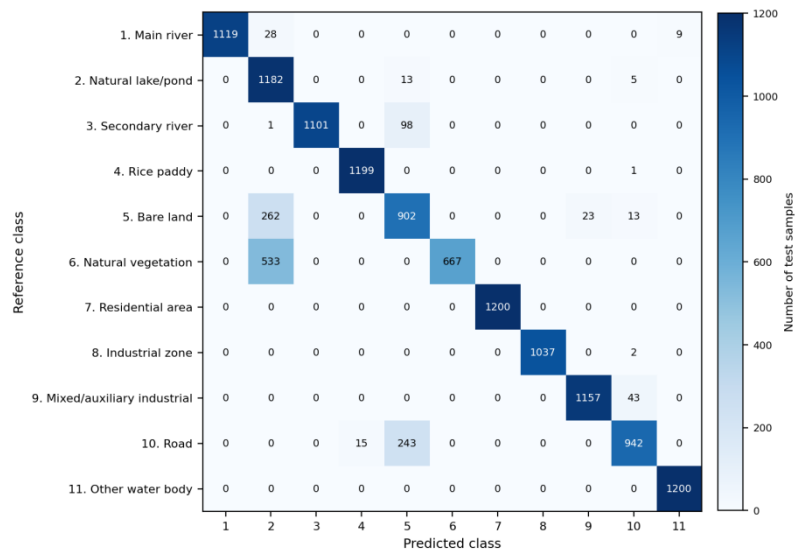


Figure 6: Confusion matrix of the proposed CNN on the test set (11 classes)

Classes with a distinctive spatial pattern were recognised almost perfectly: residential area (F1 = 1.000), industrial zone (0.999), mixed/auxiliary industrial (0.972), rice paddy (0.993) and the three river classes (0.957–0.996). These are exactly the classes whose recognition relies on shape and texture, such as dense roof clusters, large rectangular factory roofs and regular paddy parcels, which a 64×64 patch can capture. Roads, a thin linear object, reached F1 = 0.854; most errors (243/1,200) were assigned to bare land, reflecting earth and gravel roads in rural areas whose reflectance is close to bare soil.

The main confusions involved water and bare surfaces. Of the natural vegetation patches, 533/1,200 were labelled as lake/pond, and 262/1,200 bare-land patches were also assigned to lake/pond, which explains the low precision of the lake/pond class (0.589). For the vegetation class, all misclassified samples of the two pixel-based baselines (Section 3.5) fell in two of the four test polygons, located along the river bank; these polygons have mean NDVI of -0.08 and -0.06 and NDWI of 0.26 and 0.23 , close to the lake class (NDVI -0.20 , NDWI 0.30), whereas all 18 training polygons of the class have NDVI between 0.22 and 0.50 and negative NDWI. The error therefore most likely comes from reference labels (water or sparse aquatic vegetation at the acquisition time) rather than from the model. After merging into five groups, OA was 90.37% ($\kappa = 0.8689$); F1 was 0.990 for built-up and industry, 0.911 for water, 0.872 for vegetation, 0.854 for road and 0.735 for bare land.

3.4 Inference Strategy: Map Quality versus Cost

Table VI compares the two sliding-window steps on the same machine. Reducing the step from 32 to 8 pixels increased the number of predictions by about 16 times and the inference time about ten-fold, from 11.8 s to 119 s. In exchange, block artefacts along roads and field boundaries were largely removed, and the map follows object boundaries more closely (Fig. 7). Large homogeneous areas such as the rivers and paddy fields appear as continuous regions, and the road network is continuous. Normalised by the $17,687,598$ pixels actually classified, the final setting processes about $147,900$ pixels per second on a CPU. Since the scene has a fixed size and needs to be mapped only once, the extra cost was considered acceptable.

Table VI. Effect of the sliding-window step on computation (same machine)

step_size (pixels)	Number of patch predictions (approx.)	Whole-scene inference time
32 (default)	$\approx 17,400$	≈ 11.8 s
8 (adopted)	$\approx 278,000$	≈ 119 s

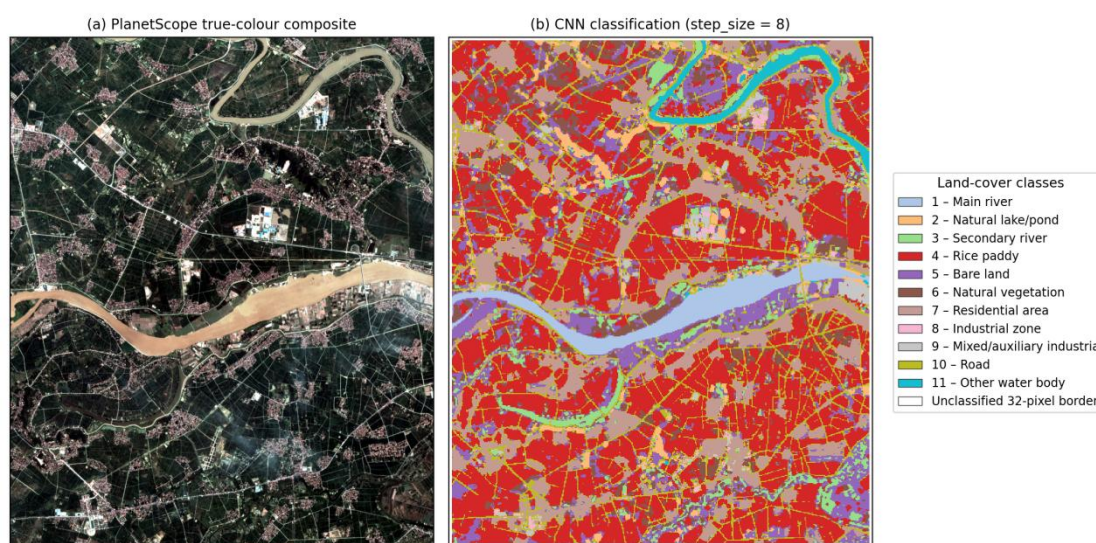


Figure 7: PlanetScope true-colour composite (a) and land-cover map produced by the proposed CNN with step_size = 8 (b)

3.5 Comparison with Traditional Baselines

To place the results in context, Table VII reports two pixel-based baselines evaluated on exactly the same test samples: a random forest (RF) [28] with 300 trees and maximum likelihood classification (MLC). Both use the six-channel vector at the centre pixel of each patch. RF and MLC were trained and evaluated on the same personal computer (CPU) as the official CNN run, so all three models are compared under identical hardware. The CNN was more accurate at both levels; the difference was largest for shape-dependent classes, for example road ($F1 = 0.854$ for the CNN versus 0.565 for RF and 0.392 for MLC) and residential area (1.000 versus 0.684 and 0.703). Part of this advantage comes from the larger spatial context available to the CNN, which is precisely the design goal of a patch-based recognition model.

Table VII. Comparison with pixel-based baselines on the same test set

Model	OA 11 classes	Kappa 11 classes	OA 5 groups	Kappa 5 groups
Proposed CNN	90.08%	0.8909	90.37%	0.8689
Random forest	82.25%	0.8046	83.67%	0.7790
Maximum likelihood	80.16%	0.7816	86.00%	0.8078

3.6 Reproducibility and Computational Cost across Environments

Runs 5 and 6 used the same code, seed and configuration and differed only in the execution environment and library versions. Table VIII shows that the GPU run on Colab was 2.70 percentage points more accurate at the 11-class level, about 1.75 times faster to train and about 8.9 times faster at inference. According to the TensorFlow determinism design [29], op-determinism guarantees identical results only on the same hardware and software stack; a different device (GPU versus CPU) or library version changes the order of floating-point operations and hence the trained weights. For model design this means that fixed seeds are necessary for repeatable experiments on one machine, but differences of a few percentage points across machines should be treated as normal variability rather than as evidence of a better or worse design. The main benefit of the GPU in this application is speed rather than accuracy.

Table VIII. Same model and seed in two execution environments

Indicator	Google Colab (GPU), run 5	Personal computer (CPU), run 6
OA / Kappa, 11 classes	92.78% / 0.9206	90.08% / 0.8909
OA / Kappa, 5 groups	93.14% / 0.9071	90.37% / 0.8689
Training time	240.06 s	421.04 s
Whole-scene inference time (step 8)	13.42 s	119.6 s

3.7 Limitations

Three limitations should be noted. First, the three design improvements were evaluated jointly, and each configuration was run once without statistical testing. Second, the validation set was not spatially separated, which weakened early stopping and learning-rate scheduling. Third, the patch-based sliding-window approach leaves a 32-pixel border unclassified and assigns one label to each 8×8 block; fully convolutional segmentation networks such as U-Net would avoid both issues.

IV. CONCLUSIONS AND RECOMMENDATIONS

This paper documented the design of a compact CNN for recognising 11 land-cover object classes in four-band PlanetScope imagery of a peri-urban area in Bac Ninh province, Vietnam. The main conclusions are:

(1) A six-channel input (four bands, NDVI and NDWI) with 64×64 patches and a four-block VGG-style network with batch normalisation and dropout (917,164 parameters) can be trained from scratch on a locally digitised dataset of about 13,000 patches and reaches OA = 90.08% ($\kappa = 0.8909$) (personal computer, CPU, official run) on spatially independent test data.

(2) Under identical hardware (Google Colab, GPU), increasing the samples per class, adding batch normalisation and a fourth block, and scheduling the learning rate raised OA from 85.10% to 92.78%; fixing the random seed was a prerequisite for measuring such gains, since randomness alone produced a spread of 3.57 percentage points.

(3) The model is most effective for classes defined by shape and texture (residential, industrial, roads), which is the purpose of patch-based recognition; remaining errors concern spectrally similar water, bare land and vegetation classes, partly because of reference-label issues.

(4) A sliding-window step of 8 pixels gives clearly smoother maps than a step of 32 at about ten times the inference cost; a randomly split validation set neutralises early stopping without biasing the test accuracy; and fixed seeds do not guarantee identical results across hardware.

It is recommended that future work use a spatially separated validation set, carry out one-factor-at-a-time ablation experiments with repeated runs and statistical tests, verify doubtful reference polygons in the field, and investigate fully convolutional architectures such as U-Net to obtain maps at native resolution without border loss.

ACKNOWLEDGMENT

This paper is a product of the scientific research project "Thiết kế mô hình học sâu dựa trên mạng nơ-ron tích chập (CNN) nhận dạng đối tượng trên ảnh viễn thám" (Design of a deep learning model based on convolutional neural networks (CNN) for object recognition in remote sensing imagery), code HUNRE.2026.11.07, managed by Hanoi University of Natural Resources and Environment (HUNRE).

REFERENCES

- [1]. LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., (1998). "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, 86(11), 2278–2324.
- [2]. Krizhevsky, A., Sutskever, I. and Hinton, G.E., (2012). "ImageNet classification with deep convolutional neural networks", *Advances in Neural Information Processing Systems*, 25, 1097–1105.
- [3]. Simonyan, K. and Zisserman, A., (2015). "Very deep convolutional networks for large-scale image recognition", *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, USA.
- [4]. He, K., Zhang, X., Ren, S. and Sun, J., (2016). "Deep residual learning for image recognition", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.
- [5]. Zhu, X.X., Tuia, D., Mou, L., Xia, G.-S., Zhang, L., Xu, F. and Fraundorfer, F., (2017). "Deep learning in remote sensing: A comprehensive review and list of resources", *IEEE Geoscience and Remote Sensing Magazine*, 5(4), 8–36.
- [6]. Ma, L., Liu, Y., Zhang, X., Ye, Y., Yin, G. and Johnson, B.A., (2019). "Deep learning in remote sensing applications: A meta-analysis and review", *ISPRS Journal of Photogrammetry and Remote Sensing*, 152, 166–177.
- [7]. Zhao, S., Tu, K., Ye, S., Tang, H., Hu, Y. and Xie, C., (2023). "Land use and land cover classification meets deep learning: A review", *Sensors*, 23(21), 8966.
- [8]. Kussul, N., Lavreniuk, M., Skakun, S. and Shelestov, A., (2017). "Deep learning classification of land cover and crop types using remote sensing data", *IEEE Geoscience and Remote Sensing Letters*, 14(5), 778–782.
- [9]. Tuan, P.L., Ha, D.V., Thanh, L.C., Thao, D.T.P. and Thanh, B.Q., (2024). "Application of EfficientNet in land surface cover classification" (in Vietnamese), *Journal of Geodesy and Cartography*, 59(3), 36–42.
- [10]. Truong, V.T., Hirayama, S., Phan, D.C., Hoang, T.T., Tadono, T. and Nasahara, K.N., (2024). "JAXA's new high-resolution land use land cover map for Vietnam using a time-feature convolutional neural network", *Scientific Reports*, 14, 3926.
- [11]. Lam, C.-N., Niculescu, S. and Bengoufa, S., (2023). "Monitoring and mapping floods and floodable areas in the Mekong Delta (Vietnam) using time-series Sentinel-1 images, convolutional neural network, multi-layer perceptron, and random forest", *Remote Sensing*, 15(8), 2001.
- [12]. Dang Hung, T., Pham, M.H., Huyen, B.T., Hanh, T.H., Hong Tinh, P., Bang, N.T. et al., (2025). "Mangrove species classification using a proposed ensemble U-Net model and Planet satellite imagery: A case study in Ngoc Hien district, Ca Mau province, Vietnam", *PLOS ONE*, 20(8), e0327315.
- [13]. Karasiak, N., Dejoux, J.-F., Monteil, C. and Sheeren, D., (2022). "Spatial dependence between training and test sets: Another pitfall of classification accuracy assessment in remote sensing", *Machine Learning*, 111(7), 2715–2740.
- [14]. Kattenborn, T., Schiefer, F., Frey, J., Feilhauer, H., Mahecha, M.D. and Dormann, C.F., (2022). "Spatially autocorrelated training and validation samples inflate performance assessment of convolutional neural networks", *ISPRS Open Journal of Photogrammetry and Remote Sensing*, 5, 100018.
- [15]. Planet Labs PBC, "PlanetScope Product Specifications", Technical documentation, Planet.com.
- [16]. Rouse, J.W., Haas, R.H., Schell, J.A. and Deering, D.W., (1974). "Monitoring vegetation systems in the Great Plains with ERTS", *Proceedings of the 3rd ERTS Symposium*, NASA SP-351, 1, 309–317.
- [17]. McFeeters, S.K., (1996). "The use of the Normalized Difference Water Index (NDWI) in the delineation of open water features", *International Journal of Remote Sensing*, 17(7), 1425–1432.
- [18]. Ioffe, S. and Szegedy, C., (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift", *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, PMLR 37, 448–456.
- [19]. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., (2014). "Dropout: A simple way to prevent neural networks from overfitting", *Journal of Machine Learning Research*, 15, 1929–1958.
- [20]. Kingma, D.P. and Ba, J., (2015). "Adam: A method for stochastic optimization", *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, USA.
- [21]. Goodfellow, I., Bengio, Y. and Courville, A., (2016). *Deep Learning*, MIT Press, Cambridge, MA.
- [22]. Shorten, C. and Khoshgoftaar, T.M., (2019). "A survey on image data augmentation for deep learning", *Journal of Big Data*, 6, 60.
- [23]. Abadi, M. et al., (2016). "TensorFlow: A system for large-scale machine learning", *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 265–283.
- [24]. Congalton, R.G., (1991). "A review of assessing the accuracy of classifications of remotely sensed data", *Remote Sensing of Environment*, 37(1), 35–46.
- [25]. Cohen, J., (1960). "A coefficient of agreement for nominal scales", *Educational and Psychological Measurement*, 20(1), 37–46.
- [26]. Pedregosa, F. et al., (2011). "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research*, 12, 2825–2830.
- [27]. Landis, J.R. and Koch, G.G., (1977). "The measurement of observer agreement for categorical data", *Biometrics*, 33(1), 159–174.
- [28]. Breiman, L., (2001). "Random forests", *Machine Learning*, 45, 5–32.
- [29]. TensorFlow Community, (2021). "RFC: Enabling determinism in TensorFlow", *tensorflow/community GitHub repository*, <https://github.com/tensorflow/community/blob/master/rfcs/20210119-determinism.md>.