remote sensing

MDPI

Article

Fine-Tuning-Based Transfer Learning for Building Extraction from Off-Nadir Remote Sensing Images

Bipul Neupane [1,2,*], Jagannath Aryal [1,2] and Abbas Rajabifard [2]

[1] Earth Observation and AI Research Group, Faculty of Engineering and Information Technology, The University of Melbourne, Parkville, VIC 3010, Australia; jagannath.aryal@unimelb.edu.au
[2] Department of Infrastructure Engineering, Faculty of Engineering and Information Technology, The University of Melbourne, Parkville, VIC 3010, Australia; abbas.r@unimelb.edu.au
* Correspondence: bneupane@student.unimelb.edu.au

Abstract: Building extraction—needed for urban planning and monitoring—is affected by the misalignment between labels and off-nadir remote sensing imagery. A computer vision approach to teacher–student learning between large–noisy and small–clean data has been introduced as a solution, but with limited accuracy and efficiency. This paper proposes fine-tuning-based transfer learning (FTL) to adapt a pre-trained model from a noisy source to a clean target dataset, improving segmentation accuracy in off-nadir images. A standardized experimental framework is developed with three new building datasets containing large–noisy and small–clean image–label pairs of multiple spatial resolutions. These datasets cover a range of building types, from low-rise to skyscrapers. Additionally, this paper presents one of the most extensive benchmarking efforts in teacher–student learning for building extraction from off-nadir images. Results demonstrate that FTL outperforms the existing methods with higher F1 scores—0.943 (low-rise), 0.868 (mid-rise), 0.912 (high-rise), and 0.697 (skyscrapers)—and higher computational efficiency. A notable gain in mean difference is observed in taller buildings from complex urban environments. The proposed method, datasets, and benchmarking framework provide a robust foundation for accurate building extraction and broader remote sensing applications.

Keywords: building dataset; convolutional neural networks; earth observation; remote sensing; semantic segmentation; transfer learning

Academic Editors: Ahmed Mustafa and Andreas Rienow

Received: 23 February 2025
Revised: 26 March 2025
Accepted: 31 March 2025
Published: 1 April 2025

Citation: Neupane, B.; Aryal, J.; Rajabifard, A. Fine-Tuning-Based Transfer Learning for Building Extraction from Off-Nadir Remote Sensing Images. *Remote Sens.* **2025**, *17*, 1251. https://doi.org/10.3390/rs17071251

1. Introduction

Building polygon (footprint/roof) datasets and their inventories enhance the management of the urban environment and improve societal aspects of urban living with applications such as mapping, urban planning, disaster management, sprawl and green space management, urban heat monitoring, change detection, and humanitarian efforts [1–3]. Additionally, building data aids in tracking housing development, managing zoning regulations, and ensuring equitable land use, enabling planners to monitor urban expansion, assess population densities, and identify infrastructure needs [4]. Furthermore, it enhances disaster preparedness by mapping vulnerable areas, planning evacuation routes, and improving urban resilience. This paper aims to improve the development of urban building datasets by addressing a major problem in remote sensing-based extraction within complex urban environments.

Urban buildings are extracted from high-resolution remote sensing (RS) images using deep learning-based semantic segmentation models [5,6]. These models classify each image pixel into 'building' and 'non-building' categories. Most segmentation models follow

an encoder–decoder network (EDN) architecture, with U-Net [7] being a widely used example. Global building datasets such as Microsoft's Building Footprints [8], Google's Open Buildings [9], and ESRI (See: https://www.arcgis.com/home/item.html?id=4e38dec1577b4b7da5365294d8a66534 (accessed on 30 March 2025), ESRI's Deep Learning Model to Extract Buildings) are produced using EDN-based segmentation models.

Global data providers face a major problem in building extraction due to label displacement error, where annotated building labels fail to align accurately with true building footprints in remote sensing imagery. This misalignment arises from the off-nadir angle of aerial and satellite sensors, which may be primarily caused by central projection errors and push-broom scanning approaches of high-resolution satellite sensors [10]. A potential solution in photogrammetry is true orthophotos, but their high computational cost, limited global availability, and processing complexity hinder large-scale adoption. Instead, data providers like Microsoft and Google rely on deep learning techniques, stereo imagery, and post-processing corrections to minimize the effects of off-nadir distortions [8,9]. Without true orthophotos, displacements can only be reduced rather than eliminated. As a result, extracting true building footprints is a challenge, and most building extraction studies assume the extracted roof polygons are the footprints. The label displacement error can increase the number of pixels for taller building heights and higher spatial resolution of off-nadir images. As illustrated in Figure 1, when spatial resolution rises from 120 cm/px (cm/px = centimeter per pixel, a unit of ground sample distance) to 60 cm/px, each original pixel is replaced by four smaller pixels, effectively quadrupling the displacement error. This label displacement error results in inaccurate training datasets, ultimately compromising the accuracy of building extraction models trained on these misaligned labels. Given its impact on large-scale data products, this problem has gained increasing attention, including research efforts such as the SpaceNet 4 Challenge [11]).

Tackling the misalignment problem is ongoing research, and the problem has recently devised many methods. Relief displacement correction [12], multitask learning [13], multilabel learning [14], and learning from depth [15] are studied as potential solutions. The segmented outputs in these methods are improved with several post-processing steps, including morphological operations that limit their robustness and end-to-end usage. Apart from addressing the misalignment problem, some studies distinguish between the facade and roof in a multi-view oblique image [16–18]. Wang et al. [19] developed the BONAI building dataset with labels for the roof, footprints, and the offset between them and employed a learning offset vector (LOFT) method with CNN that learned from the offset. The accuracy of the BONAI dataset was further improved recently by [14] using multilabel learning of footprint, roof, and additional 'shape' labels. These methods have provided remarkable achievements in tackling the misalignment problem, given that the annotations (labels) are available for multiple features of oblique-view buildings. However, developing a dataset that constitutes these features (footprint, roof, facade, and offset) is an additional challenge to the already difficult process of labeling footprints alone on the limited remote sensing images. While these methods require multiple post-processing steps or additional annotations for other building features over oblique-view images, other studies focus on knowledge distillation with a teacher–student learning approach.

Knowledge distillation (KD) [20] is a technique where a teacher model, trained on a large dataset, transfers its knowledge to a student model, which is typically smaller and more efficient. KD has been widely applied in learning from noisy labels (aka. supervision from label noise) [21,22] and has shown potential in remote sensing applications [23]. Xu et al. [24] adopted this KD-based teacher–student learning approach to address the label displacement problem. Their approach involved training a teacher on a large but noisy dataset (with misaligned labels) and distilling its knowledge into a student that learns from

a smaller, cleaner dataset (without misalignment), making the student more robust to label noise. They compare their results to deep mutual learning (DML) [25], where multiple student models collaborate and teach each other while distilling knowledge from the teacher. Expanding on their work, Ref. [26] introduced fine-tuning learning (FTL), where models trained on noisy off-nadir images are adapted to true orthophotos, demonstrating that FTL was more effective than KD for training misalignment-tolerant students. Originally, KD was designed to transfer knowledge from large, high-performing models to smaller, deployable students [20], but this process inherently limits the student's ability to surpass the teacher's performance since smaller models retain less information. This constraint reduces accuracy in segmenting complex urban buildings, whereas FTL, unlike KD and DML, allows re-training beyond the pre-trained teacher's limitations, enabling the student to develop greater noise tolerance and adapt more effectively to new datasets.

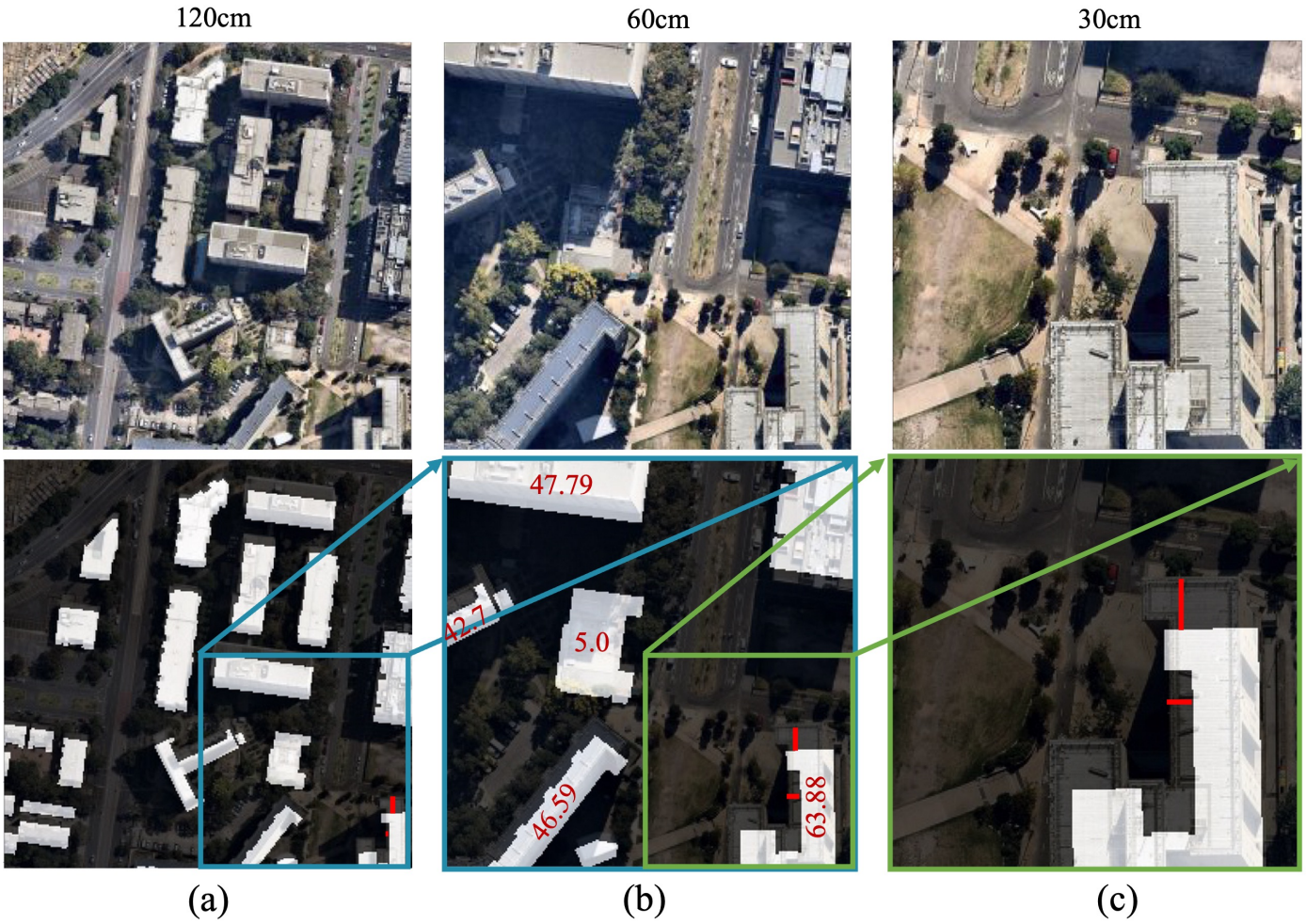

Figure 1. An illustrative example demonstrating the increase in misalignment (label displacement error) with higher spatial resolution and building height. (**a**) Image–label pair at 120 cm/px resolution, (**b**) at 60 cm/px resolution, and (**c**) at 30 cm/px resolution. The red numbers indicate building heights, while the red lines represent the extent of misalignment. The blue and green rectangles highlight the scale change from 120 to 60 cm/px and from 60 to 30 cm/px, respectively. As shown, a 63.88 m tall building exhibits greater displacement in higher-resolution images. Images were obtained from the Nearmap Tile API over the City of Melbourne, Australia.

FTL adopts a pre-trained model on a source domain by further training it on a target domain using labeled target data while retaining knowledge from the source. FTL can also be called supervised domain adaptation if it explicitly addresses domain shifts between

the source and target. Given that our datasets contain labels in both domains, FTL is more suitable than unsupervised domain adaptation (UDA) to adapt a student model to the target dataset. Studies have shown that FTL enhances building extraction accuracy [27,28] and is effective in fine-tuning models trained on large–noisy datasets for improved performance on small–clean datasets [29–31].

With FTL showing promising results in training misalignment-tolerant students compared with other teacher–student learning methods like KD, several research questions remain unanswered. These methods are introduced to address the misalignment problem, yet their experimentation in varied settings remains minimal. From the available studies of [24,26], two knowledge gaps are identified: (i) the teacher–student learning methods have not yet been studied for different building types of varying heights and images of different spatial resolutions; (ii) the available CNNs and EDNs are not benchmarked to identify the optimal teachers (large immovable networks) and students (lightweight and deployable) for the three methods.

To develop accurate building extraction models that are more tolerant to misalignment in off-nadir images and to investigate the two identified knowledge gaps, this study makes the following contributions:

- FTL is introduced to mitigate misalignment between building image–label pairs caused by off-nadir angles. FTL is implemented within a teacher–student learning framework, where a large teacher model is pre-trained on a misaligned dataset (source domain) and subsequently fine-tuned on a smaller, error-free student dataset (target domain) to improve segmentation accuracy.
- A structured experimental setup is designed to systematically compare FTL against KD, DML, and the models without FTL. Three new urban building datasets, including high-rise and skyscraper buildings, are developed with "large–noisy" and "small–clean" image–label pairs. The datasets and codes are openly released for reproducibility (see the data availability statement).
- The methods are evaluated across four building types (low-rise, mid-rise, high-rise, and skyscrapers) and three spatial resolutions, ensuring a comprehensive performance analysis.
- A comparative analysis of 43 representative lightweight CNNs, five optimizers, nine loss functions, and seven EDNs is conducted to identify the optimal trade-off between lightweight efficiency and high-performance models. The CNNs include state-of-the-art lightweight architectures from Google, Apple, Meta, Amazon, and Huawei. To the best of our knowledge, such a comprehensive benchmark of lightweight CNNs and their integration into an EDN is novel for building extraction.

2. Materials and Methods

The proposed method of FTL between large-noisy and small–clean datasets to achieve misalignment-tolerant models is carried out in a four-step workflow: (i) data preparation, (ii) selection of teacher and student network architectures, (iii) fine-tuning-based transfer learning, and (iv) evaluation of the fine-tuned students against the students distilled from KD and DML. The workflow is illustrated in Figure 2 and is detailed stepwise in this section.

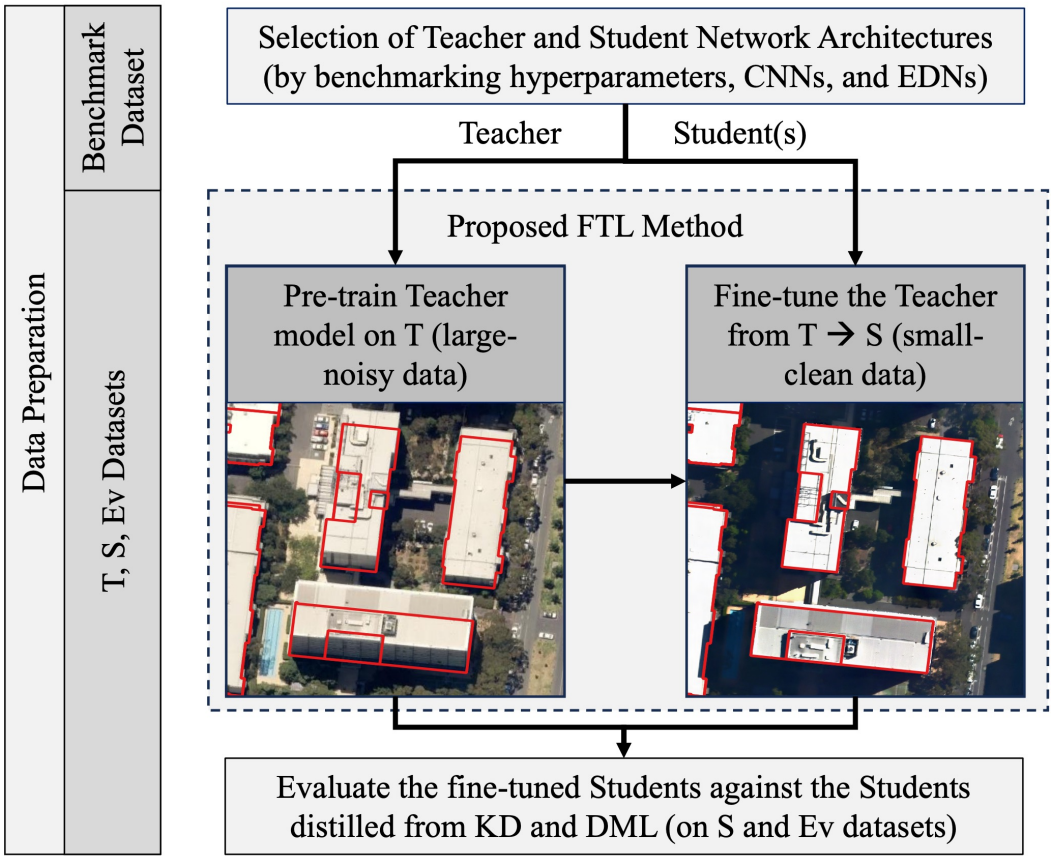

Figure 2. The workflow of the proposed FTL method to train noise-tolerant models against the misalignment of building labels in training data due to off-nadir images. Samples of large–noisy (**left**) and small–clean (**right**) datasets are provided to visualize the misalignment between the two datasets. The noise refers to the label displacement error caused by the off-nadir angle of source imagery.

2.1. Data Preparation

This study uses four datasets: one benchmark dataset for benchmarking the hyperparameters, CNNs, and EDNs for our experimental setup of FTL and three newly developed datasets for training, fine-tuning/distillation, and evaluation of EDNs. The benchmark dataset helps identify optimal configurations for setting up the teacher (pre-trained network) and students (fine-tuned/distilled networks) in FTL. Once configured, the teacher model is trained and evaluated on the teacher's dataset (T), while the student models undergo fine-tuning/distillation and are evaluated using the student's dataset (S) and Evaluation dataset (Ev). The effectiveness of the student models in reducing misalignment is assessed based on their performance on these two datasets.

The benchmark dataset is derived from the widely used Massachusetts Building dataset [32]. A smaller subset of the dataset, as provided in [5], is used for experimental consistency and lower memory usage to benchmark the lightweight models. This subset crops the original 1500 × 1500 px (px = number of pixels) tiles to 256 × 256 px, and the training and test images are reduced by 4× and 2×. The number of validation images is kept the same as in the original dataset.

The teacher's dataset (T) is developed by masking and tiling the building roof polygons provided by the City of Melbourne. See https://data.melbourne.vic.gov.au/explore/dataset/2020-building-footprints/information/ (accessed on 15 April 2024), 2020 Building Footprints from the City of Melbourne. Multi-resolution image tiles of 30, 60, and 120 cm/px spatial resolution are collected from the *Nearmap Tile API*. The vector labels are

rasterized with reference to the image using Algorithm 1. The misalignment between these image–label pairs is not accounted for, making it the "large–noisy" teacher's dataset.

Algorithm 1 Building Dataset Preparation from Nearmap API Service

Require: Building Geodatabase (GDB), API key, zoom level
Ensure: Geospatial tiles and rasterised building polygons

function FETCHANDRASTERISETILES(*polygons*, *apiKey*, *zoom*)
  existingTiles ← ∅
  for all *polygon* ∈ *polygons* **do**
    centroid ← CalculateCentroid(*polygon*)
    tile ← ConvertToTile(*centroid*, *zoom*)
    if *tile* ∉ *existingTiles* **then**
      image ← FetchTile(*tile*, *apiKey*)
      SaveTile(*image*)
      existingTiles ← *existingTiles* ∪ {*tile*}
    end if
  end for
end function
procedure DATASETPREPARATION(*builingGDB*, *apiKey*, *zoom*)
  polygons ← LoadShapefile(*buildingGDB*)
  FETCHANDRASTERISETILES(*polygons*, *apiKey*, *zoom*)
end procedure
DATASETPREPARATION(*buildingGDB*, *apiKey*, *zoom*)

Student's dataset (*S*) is developed for fine-tuning/distillation purposes by collecting images over the central business district (CBD) of Melbourne. The CBD consists mostly of high-rise and skyscraper buildings with complex structures, providing challenging urban environments. For the images, both off-nadir images and true orthophotos are collected from Nearmap. The orthophoto is separately downloaded and tiled, and the off-nadir image tiles are collected from the API service. City of Melbourne's building roof polygons are overlaid over the true orthophoto image, and for the off-nadir image tiles, the buildings are manually annotated (by hand). Therefore, unlike *T*, this dataset provides 'clean' image–label pairs.

The evaluation dataset (*Ev*) is curated as a dataset to evaluate the fine-tuned/distilled networks on the manually annotated labels for image scenes of the four building types. This dataset does not have training samples (only validation). A subset of 55 image tiles of *T* is taken and manually annotated (by hand). The building types are separated according to their heights as defined by the Australian Bureau of Statistics (ABS). See: https://www.abs.gov.au/ausstats/abs@.nsf/Lookup/8752.0Feature+Article1Dec%202018 (accessed on 15 April 2024), Characteristics of Apartment Building Heights by ABS.

Table 1 summarizes the datasets, where (i) *T* contains misaligned image–label pairs for training and validation, (ii) *S* includes both off-nadir and true orthophoto images with clean labels, and (iii) *Ev* features off-nadir images from *T* with clean labels. "Clean labels" refer to overlaying labels without misalignment. Both *S* and *Ev* consist of complex high-rise buildings from the CBD, making them valuable for evaluating segmentation performance under challenging conditions. All datasets have an image size of 256 × 256 px. Figure 3 presents the study area and dataset samples.

Table 1. Summary of the datasets prepared for this study.

Dataset	Label Quality	Image Res. (cm/px)	Train-Test	Building Type Majority
Benchmark Dataset	As provided	100	800-260	Low-rise
Teacher's Dataset (*T*)	Noisy/misaligned	30, 60, 120	6626-643	Low/mid/high-rise and skyscrapers
Student's Dataset (*S*)	Clean (manually labeled)	30, 60, 120	746-243	High-rise and skyscraper
Evaluation Dataset (*Ev*)	Clean (manually labeled)	30, 60, 120	0-55	Low/mid/high-rise and skyscrapers

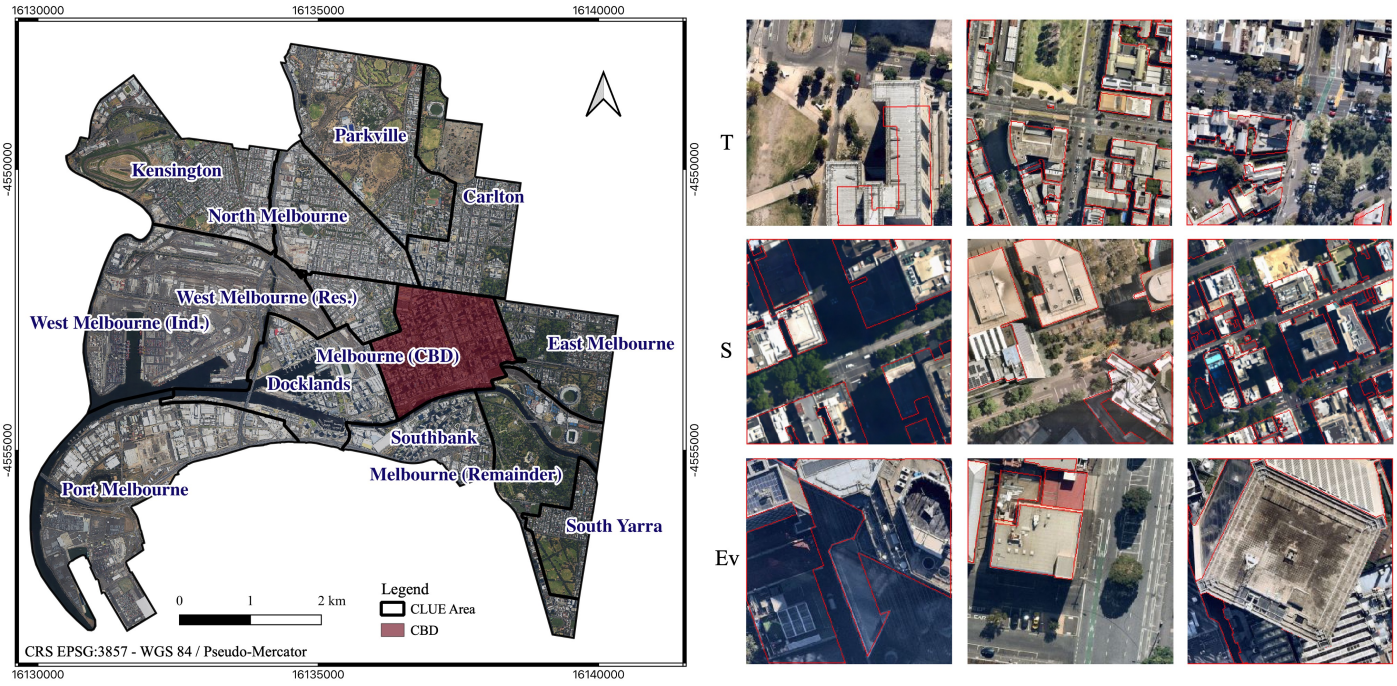

Figure 3. Study area and sample image–label pairs of the teacher's dataset (T), student's dataset (S), and Evaluation dataset (Ev). (**a**) City of Melbourne's Census of Land Use and Employment (CLUE) boundaries and CBD area. (**b**) Sample image–label pairs from T, S, and Ev datasets, with red polygons as building labels.

2.2. Selection of Representative Teacher and Student Networks

EDNs enable flexible integration of backbone CNNs (encoders) for multi-scale feature extraction in semantic segmentation, with U-Net being a widely used architecture for building segmentation [5]. The encoder processes input images, while the decoder upsamples learned features to generate the segmented output, with skip connections preserving spatial details. An illustrative example of a U-Net EDN and two CNN examples—EfficientNetv2 and MobileViT—are provided in Figure 4.

EfficientNetv2 [33] is a lightweight CNN from Google Brain. It uses Fused-MBConv convolutional blocks alongside the NAS component, and like its predecessor v1, it offers multiple scaled versions. The network uses different variations of Mobile Inverted Bottleneck Convolution (MBConv) blocks: MBConv1, MBConv4, and MBConv6. These blocks are a type of inverted residual block central to the network architecture, providing efficient and flexible building blocks for CNNs. The blocks consist of the EdgeResidual layer to improve the flow of gradients during backpropagation, the non-linear activation

function SiLU to introduce non-linearity, and the Squeeze-and-Excitation layer to improve the representational power of a network by explicitly modeling the interdependencies between the channels of convolutional features.

MobileViT [34] is a compact vision transformer developed for mobile devices. These models use MobileNetv2 (MV2) blocks to leverage their efficiency and lightweight nature for mobile and edge devices. MV2 blocks consist of an expansion layer, depthwise convolution, ReLU6 activation for non-linearity, pointwise convolution to project features back to the desired output dimension, and residual connection between input and output. It uses an additional MobileViT block, which is specialized to integrate the strengths of CNNs and transformers. It consists of the convolutional layer that extracts local features, the Transformer layer to capture global context and long-range dependencies, and the Reconstruction layer that reshapes and passes the output of the Transformer layer through another convolutional layer (to combine the local and global information). MobileViT is available in several smaller versions (s, xs, and xxs).

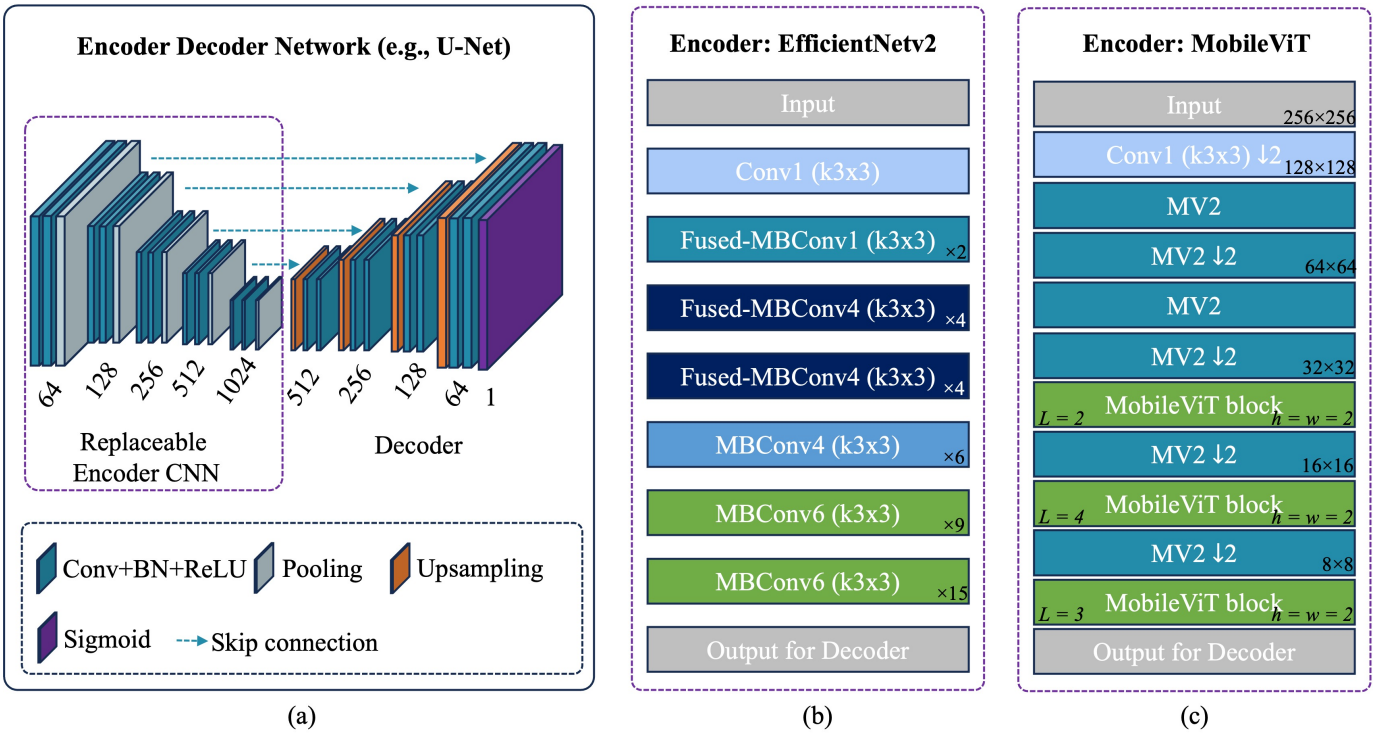

Figure 4. An illustrative example of EDN formulated with different options of encoder. (**a**) Network architecture of U-Net with replaceable encoder, skip connections, and decoder. (**b**) An example of EfficientNetv2 (adapted and modified from [33]) as the encoder. (**c**) An example of MobileViT (adapted and modified from [34]) as the encoder.

Optimal performance in EDNs is achieved by tuning hyperparameters, including the loss function ($\mathcal{L}$), which quantifies prediction errors (e.g., dice loss, Jaccard loss), the optimizer (e.g., Adam, SGD) for weight updates, the learning rate (η) controlling step size during training, and the batch size, affecting memory use and convergence stability. The best-performing model is selected by systematically searching for hyperparameter configurations, training until validation loss stabilizes, and evaluating metrics like IoU and F1 score to ensure high segmentation accuracy and generalization. Therefore, before applying the proposed FTL and comparing it to other transfer learning methods, we carry out a systematic search of the (i) hyperparameters, (ii) EDNs, (iii) student models, and (iv) teacher models with an extensive benchmark study. The detailed benchmark is provided in Appendix A; however, its summary can be categorized into four steps:

Step 1. Hyperparameter search: A CNN encoder (MobileOne-s1) is randomly chosen and integrated into a U-Net EDN to explore various hyperparameters, including learning rate, converging epoch, loss functions, and optimizers. Five optimizers and nine loss functions are tested, with learning rate scheduling and early stopping at the converging epoch. A combination of the RMSProp optimizer, dice loss, and 0.0001 learning rate provided the highest scores in most experiments.

Step 2. EDN Comparison: The best hyperparameters from Step 1 are applied to eight state-of-the-art EDNs—U-Net, U-Net++ [35], U-Net3+ [36], LinkNet [37], PSPNet [38], FPN [39], DeepLabv3+ [40], and MANet [41]—to identify an architecture that balances model size and performance for the teacher and student networks. From the benchmarks, U-Net provided the best trade-off between the number of parameters, network complexity, and evaluation scores. It is therefore the best EDN to configure student and teacher models for the proposed FTL.

Step 3. Student search: 43 lightweight CNNs (e.g., ResNeSt, GERNet, MobileOne, TinyNet, MobileViT, MobileNet, EfficientNet) (The explanation of the CNN along with their references is provided at https://github.com/bipulneupane/TeacherStudentBuildingExtractor (accessed on 30 March 2025)). They are benchmarked as potential students. These CNNs are integrated into the EDN from Step 2 (U-Net) and trained with the hyperparameters from Step 1. For the integration, we remove their final layers of 1x1 convolution and fully connected layers. The remaining layers are then connected to the decoder that comprises 2D convolution, batch normalization (BN), ReLU, and attention layers. The top five best student models are selected based on a trade-off between network parameters, loss, training time, and evaluation scores. The top five student models were U-EfficientNetv2B3 (high scorer), U-EfficientNet-lite0 (best trade-off between evaluation scores and network parameters), U-MNASNet-s, U-TinyNet-e, and U-MobileViT-xxs (with the smallest size, suitable to configure student models).

Step 4. Teacher search: For FTL, the teacher and student share the same EDN. However, to facilitate KD and DML for comparison, the teacher is a larger, static model distilling knowledge to smaller students. We evaluate a larger version of the student's CNN and U-VGG19, given its strong benchmark performance [5]. U-VGG19 was identified as the best teacher with the highest F1 score and lower network parameters when compared with U-EfficientNetv2L (the largest version of the high-scorer EfficientNetv2B3 from Step 3).

2.3. Proposed Fine-Tuning-Based Transfer Learning (FTL)

The proposed FTL fine-tunes a model pre-trained on a source dataset by updating all layers on a target dataset. The process begins with pre-training an EDN on a large–noisy dataset, T (source domain = teacher's dataset), containing misaligned labels. The model, denoted as f_θ, learns broad feature representations. The pre-trained model (teacher) is then fine-tuned on a clean dataset S (target domain = student's dataset). The goal is to adapt the model beyond the constraints of the teacher while retaining generalizable features learned from T. Fine-tuning consists of two key phases:

2.3.1. Pre-Training on Noisy Labels

The EDN, parameterised by θ, is trained on the noisy T dataset:

$$T = (x_T^i, y_T^i)_{i=1}^{N_T}, \quad (1)$$

where $x_T^i \in \mathbb{R}^{H\times W\times C}$ represents the input image of height H, width W, and number of channels C. Similarly, $y_T^i \in \{0,1\}^{H\times W}$ is the corresponding noisy building label, and N_T is the number of samples in T. The training minimizes a segmentation loss, such as dice loss:

$$\mathcal{L}_{\text{Dice}}(f\theta(x), y) = 1 - \frac{2\sum f_\theta(x)\cdot y + \epsilon}{\sum f_\theta(x) + \sum y + \epsilon} \tag{2}$$

where $f_\theta(x)$ is the predicted segmentation mask, y is the input label mask, and ϵ prevents division by zero. The pre-training loss across T is

$$\mathcal{L}_T(\theta) = \frac{1}{N_T}\sum_{i=1}^{N_T}\mathcal{L}_{\text{Dice}}(f\theta(x_T^i), y_T^i) \tag{3}$$

After minimizing $\mathcal{L}_T$, we obtain the pre-trained model parameters, θ_T.

2.3.2. Fine-Tuning on Clean Labels

The pre-trained model is fine-tuned on the student dataset:

$$S = (x_S^j, y_S^j)_{j=1}^{N_S}, \tag{4}$$

where $x_S^j \in \mathbb{R}^{H\times W\times C}$ is the clean input image, $y_S^j \in \{0,1\}^{H\times W}$ is its input label, and N_S is the number of samples in S. Fine-tuning minimizes dice loss over S:

$$\mathcal{L}_S(\theta) = \frac{1}{N_S}\sum_{j=1}^{N_S}\mathcal{L}_{\text{Dice}}(f\theta(x_S^j), y_S^j) \tag{5}$$

The model parameters are updated using gradient descent:

$$\theta_S = \theta_T - \eta\nabla_\theta\mathcal{L}_S(\theta) \tag{6}$$

where θ_S are the fine-tuned parameters, η is the learning rate, and $\nabla_\theta\mathcal{L}_S$ is the gradient of the loss. Since all layers are fine-tuned, the encoder and decoder adapt to S, improving segmentation precision.

2.4. Comparison Methods

The proposed FTL method is compared with existing teacher–student learning methods: knowledge distillation (KD) and deep mutual learning (DML). KD [20] distills a student on a small–clean dataset while being supervised by a teacher trained on a large–noisy set. DML [25] extends this by enabling multiple students to collaboratively learn from each other under teacher supervision. The two methods are established for the other applications of supervision from label noise [21,22,42,43]. Xu et al. [24] applied both methods to the label displacement problem we address, though with limited accuracy and unresolved knowledge gaps mentioned before. As KD and DML are the only teacher–student approaches introduced for this problem, we adopt them in our experimental setup with their respective loss functions for a fair, state-of-the-art comparison with FTL.

2.4.1. Knowledge Distillation

KD transfers knowledge from a large teacher model trained on noisy data (T) to a smaller student model trained on clean data (S) while minimizing performance loss. The student is optimized using a total loss:

$$\mathcal{L}_{\text{Total}} = \alpha\mathcal{L}_{\text{Dice}} + (1-\alpha)\mathcal{L}_{\text{Distil}} \tag{7}$$

where α balances segmentation accuracy ($\mathcal{L}_{\text{Dice}}$) and feature alignment ($\mathcal{L}_{\text{Distil}}$). The distillation loss aligns multi-scale feature maps:

$$\mathcal{L}_{\text{Distil}} = \mathcal{L}_{\text{MSE}}\left(\sigma\left(p_j^t/\tau\right), \sigma\left(p_j^s/\tau\right)\right) \tag{8}$$

where p_j^t and p_j^s are the predictions from the teacher and student for pixel j, σ is the sigmoid activation, and τ controls the softness of the predictions. The Mean Squared Error (MSE) loss is

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(y_j - p_j)^2 \tag{9}$$

where y_j and p_j are the GT and prediction for pixel j, and $\frac{1}{N}$ normalises the loss.

By optimizing $\mathcal{L}_{\text{Total}}$, KD ensures that the student retains the teacher's knowledge while refining its segmentation accuracy. However, the student model is inherently constrained by the teacher's performance, limiting its potential.

2.4.2. Deep Mutual Learning

DML enables multiple models to learn collaboratively rather than through a one-way teacher-to-student transfer. In this setup, one teacher and two students train jointly, benefiting from label supervision and mutual distillation. This encourages knowledge exchange, likely enhancing generalization and robustness, particularly in data-scarce settings like *S*.

Each model M_k (teacher or student) is trained with a segmentation loss and a mutual distillation loss:

$$\mathcal{L}_{\text{Total}}^{(k)} = \alpha\mathcal{L}_{\text{Dice}}^{(k)} + \frac{1-\alpha}{K}\sum_{i=1, i\neq k}^{K}\mathcal{L}_{\text{Distil}}^{(k,i)} \tag{10}$$

where $\mathcal{L}_{\text{Dice}}^{(k)}$ is the segmentation loss for M_k, $\mathcal{L}_{\text{Distil}}^{(k,i)}$ is the mutual distillation loss between M_k and M_i, and α balances segmentation accuracy and mutual learning.

The mutual distillation loss encourages agreement between models:

$$\mathcal{L}_{\text{Distil}}^{(k,i)} = \mathcal{L}_{\text{MSE}}\left(\sigma\left(p_j^k/\tau\right), \sigma\left(p_j^i/\tau\right)\right) \tag{11}$$

where p_j^k and p_j^i are the predictions for pixel j from models M_k and M_i, σ is the sigmoid activation function, and τ controls distribution softening.

By aligning their predictions, students develop a consensus on segmentation accuracy, improving generalization. DML is particularly effective when labeled data in *S* is limited, as models benefit from mutual knowledge exchange.

2.5. Experimental Setup

The experimental setup with training details, accuracy metrics, train-val settings, and baseline scores for comparison is provided below.

2.5.1. Training Details and Accuracy Metrics

All networks are wrapped in the PyTorch (version 2.2.2) using the Segmentation Models Pytorch library (version 0.3.4) [44] and trained on an Nvidia A100 GPU (Nvidia, Santa Clara, CA, USA) (with 80 GB RAM). The teacher and students are trained up to 50 and 200 epochs, respectively. The learning rate is reduced upon a plateau of intersection over union (IoU) metric by a factor of 0.1 with a patience of 10 epochs. A sigmoid function classifies the final binary outputs. Four accuracy metrics are used for evaluation: precision (P), recall (R), IoU, and F1 score. The mathematical notation of the metrics is from [5].

2.5.2. Train-Val Settings

The experiments are performed in different experimental settings for FTL, KD, and DML. Depending on the training (train) and validation (val) subsets of T, S, and Ev datasets, five experimental settings are used for the comparison:

- T–T: training/fine-tuning/distilling on T; validation on T.
- S–S: training/fine-tuning/distilling on S; validation on S.
- T–S: training/fine-tuning/distilling on T; validation on S.
- T–Ev: training/fine-tuning/distilling on T; validation on Ev.
- S–Ev: training/fine-tuning/distilling on S; validation on Ev.

2.5.3. Baseline Scores for Comparison

With the teacher and students identified from the benchmarks, we trained them on the T and S datasets alone. The trained models were evaluated on Ev to produce the baseline scores for the comparison of the transfer learning methods.

Teacher's baseline scores were produced in T–T, T–S, and T–Ev settings, as reported in Table 2. From the difference in T–T and T–S settings, it was found that the teacher's ability to generalize the validation samples from S was influenced by the domain shift between T and S. This domain shift can be attributed to two primary factors: the presence of unfamiliar, complex urban high-rise buildings in S and the contrast between the precision of ortho-rectification between the samples of T and S. However, in the T–Ev setting, the teacher demonstrated its capacity to generalize complex high-rise buildings in Ev, even though it was trained on misaligned labels in T. The complexity of the clean data S and Ev was prominent in the experimental results.

Table 2. Teacher's baseline score obtained from the performance comparison of teacher U-VGG19 trained on T and validated on three datasets (T, S, and Ev).

Setting	Loss	P	R	IoU	F1	ms/it
T–T	0.080	0.929	0.917	0.860	0.920	235
T–S	0.186	0.770	0.894	0.713	0.814	40
T–Ev	0.169	0.877	0.814	0.746	0.831	40

Student's baseline scores (without transfer learning) were studied in S–S and S–Ev settings. Table 3 reports their baseline scores along with the performance of the three knowledge transfer methods. U-VGG19 (0.893 F1) and U-EfficientNetv2B3 (0.819 F1) score the highest when validated on S and Ev, respectively. The performance of students after FTL and other knowledge transfer methods will be evaluated against these baseline scores.

Table 3. Student's baseline scores obtained from the performance comparison of the selected students in settings S–S and S–Ev. S–S: Trained and validated on S; S–Ev: Trained on S and validated on Ev. Boldface: best performance.

Network	Loss	P	R	IoU	F1	ms/it	Loss	P	R	IoU	F1	ms/it
	S–S						S–Ev					
U-VGG19	**0.107**	**0.890**	**0.913**	**0.824**	**0.893**	224	0.215	**0.866**	0.741	0.676	0.785	**41**
U-Eff.Net-lite0	0.155	0.838	0.877	0.754	0.845	190	0.195	0.816	0.814	0.687	0.805	75
U-Eff.Netv2B3	0.145	0.840	0.901	0.769	0.855	392	**0.181**	0.838	**0.819**	**0.708**	**0.819**	96
U-MNASNet-s	0.184	0.797	0.872	0.710	0.816	219	0.235	0.784	0.773	0.639	0.765	56
U-MobileViT-xxs	0.181	0.803	0.872	0.717	0.819	296	0.233	0.814	0.748	0.634	0.767	154
U-TinyNet-e	0.174	0.795	0.896	0.727	0.826	**181**	0.221	0.808	0.776	0.648	0.779	56

Having thus established our experimental setup, we now present the comparative evaluation of FTL against KD and DML across different types of urban buildings.

3. Results

This section presents the results from the proposed FTL method and compares FTL with KD and DML. The individual performances are first reported for the three methods, and then the performance, training times, and computational efficiency are compared. In addition, an ablation of FTL vs. without FTL is reported at the end of this section.

3.1. Individual Performances of FTL, KD, and DML

First, the results from the proposed FTL method and the KD and DML comparison methods are presented individually.

3.1.1. Results from FTL

FTL consistently improved the performance of all student models, demonstrating its effectiveness over standalone training (baseline). The performance of FTL under S–S and S–Ev settings is reported in Table 4. In both S–S and S–Ev, all students achieved higher scores than their baseline counterparts (Table 3), except for U-EfficientNet-lite0 under S–Ev. For example, the IoU of U-EfficientNetv2b3 improved by 3.6% after FTL (0.805 vs. 0.769) in the S–S setting and by 4.4% (0.752 vs. 0.708) in the S–Ev setting. Additionally, all students trained with FTL outperformed the teacher model (T–S setting from Table 2) when evaluated on the S dataset, indicating that fine-tuning provided better adaptation to the target domain. For example, U-VGG19 after FTL provided 16.3% higher IoU (0.832 vs. 0.713) compared with the teacher's baseline score (before FTL).

Table 4. Results from the proposed FTL method. S–S: FTL from T to S, and validated on S; S–Ev: FTL from T to S, and validated on Ev; 'ms/it': training time in milliseconds per iteration; Boldface: best performance.

Network	Loss	P	R	IoU	F1	ms/it	Loss	P	R	IoU	F1	ms/it
	S–S						S–Ev					
U-VGG19	**0.098**	**0.906**	**0.914**	**0.838**	**0.903**	231	0.173	**0.917**	0.784	0.737	0.827	48
U-Eff.Net-lite0	0.138	0.859	0.890	0.778	0.862	194	0.197	0.873	0.763	0.688	0.803	103
U-Eff.Netv2B3	0.120	0.869	**0.914**	0.805	0.880	386	**0.153**	0.874	**0.837**	**0.752**	**0.847**	95
U-MNASNet-s	0.151	0.828	0.900	0.759	0.849	215	0.205	0.844	0.772	0.690	0.795	**55**
U-MobileViT-xxs	0.145	0.846	0.890	0.769	0.855	294	0.191	0.848	0.793	0.694	0.809	149
U-TinyNet-e	0.160	0.831	0.882	0.748	0.840	**181**	0.222	0.844	0.741	0.653	0.779	56

Among the students, U-VGG19 achieved the highest F1 score (0.903) in the S–S setting, while U-EfficientNetv2B3 performed best in the S–Ev setting (0.847 F1). This suggests that U-EfficientNetv2B3 is the most suitable student for FTL when generalizing to the Ev dataset, making it the preferred choice for enhanced segmentation performance in more complex environments.

3.1.2. Results from KD

The performance of KD under S–S and S–Ev settings is reported in Table 5. In comparison to the students' baseline scores, most distilled networks (with KD) underperformed, including a decline in F1 scores. KD applied to U-VGG19 with self-distillation (same teacher and student model) showed some improvement in both S–S and S–Ev settings, but it still failed to surpass FTL and did not achieve model compression. U-MNASNet-s was the only exception, producing higher scores in S–Ev, indicating that certain architectures may benefit more from distillation than others.

Among the smaller student models, U-EfficientNetv2B3 achieved the highest F1 score (0.851) in the S–S setting, while U-EfficientNet-lite0 performed best in the S–Ev setting (0.784 F1). This suggests that U-EfficientNet-lite0 is the most effective student for KD in the Ev dataset, offering a significant reduction in network parameters—from 29.1 M (U-VGG19) to 5.6 M. This is an 82% decrease, demonstrating the potential of KD for lightweight model compression.

Table 5. Results from KD. S–S: Distilled from T to S and validated on S; S–Ev: Distilled from T to S and validated on Ev; 'ms/it': training time in milliseconds per iteration; 'Par. Red. (%)': percentage of the reduced number of network parameters; Boldface: best performance.

Network	S–S						S–Ev						Par. Red.(%)
	Loss	P	R	IoU	F1	ms/it	Loss	P	R	IoU	F1	ms/it	
U-VGG19	**0.099**	0.907	**0.895**	**0.820**	**0.901**	410	**0.184**	**0.877**	0.772	**0.707**	**0.812**	40	0
U-Eff.Net-lite0	0.265	**0.944**	0.602	0.581	0.735	306	0.232	0.794	0.802	0.656	0.784	43	82
U-Eff.Netv2B3	0.149	0.895	0.810	0.740	0.851	467	0.219	0.844	0.735	0.650	0.774	77	50
U-MNASNet-s	0.199	0.869	0.744	0.669	0.801	278	0.223	0.756	**0.842**	0.654	0.781	52	93
U-MobileViT-xxs	0.214	0.811	0.763	0.648	0.786	346	0.320	0.840	0.610	0.538	0.688	93	89
U-TinyNet-e	0.179	0.860	0.785	0.696	0.821	**315**	0.294	0.826	0.638	0.554	0.702	**46**	92

3.1.3. Results from DML

Similar to KD, experiments were conducted between the teacher (U-VGG19) and the selected students, with DML performed between six pairs of students. Due to the high computational cost of loading multiple EDNs into memory, only two students were distilled at a time. Notably, U-MobileViT-xxs was excluded from DML experiments due to its poor performance with KD.

The results under S–S and S–Ev settings are reported in Table 6. In the S–S, all students performed worse than their baseline scores, with most student pairs yielding lower scores. However, DML showed some improvement in the S–Ev, although it also reduced performance in at least one student from each pair. Two pairs, (U-VGG19, U-TinyNet-e) and (U-EfficientNetv2B3, U-MNASNet-s), showed improvements, with both students in each pair achieving higher F1 scores than their baseline models. The pair of (U-VGG19, U-EfficientNetv2B3) produced the highest average scores across both settings, suggesting that DML can benefit certain student pairs, but its impact is inconsistent and highly dependent on the models involved.

Table 6. Results from DML. S–S: Distilled from T to S and validated on S; S–Ev: Distilled from T to S and validated on Ev; 'ms/it': training time in milliseconds per iteration; 'Par. Red. (%)': percentage of the reduced number of network parameters; Boldface: best performance.

Network	S–S						S–Ev						Par. Red.(%)
	Loss	P	R	IoU	F1	ms/it	Loss	P	R	IoU	F1	ms/it	
U-VGG19	0.118	0.912	0.873	0.806	0.882	862	0.201	0.888	0.739	0.685	0.793	**41**	0
U-Eff.Netv2B3	0.168	0.881	0.819	0.731	0.832		**0.192**	0.829	0.810	**0.696**	**0.810**	78	82
U-VGG19	0.121	**0.913**	0.869	0.802	0.879	699	0.196	**0.889**	0.743	0.689	0.796	**41**	0
U-TinyNet-e	0.201	0.883	0.768	0.685	0.799		0.221	0.775	0.808	0.658	0.781	44	92
U-VGG19	**0.117**	0.907	**0.882**	**0.808**	**0.883**	**690**	0.204	0.883	0.731	0.678	0.784	47	0
U-MNASNet-s	0.191	0.880	0.785	0.701	0.809		0.253	0.805	0.722	0.615	0.748	50	93
U-Eff.Netv2B3	0.172	0.879	0.817	0.726	0.828	775	0.232	0.846	0.718	0.630	0.761	75	50
U-Eff.Net-lite0	0.179	0.886	0.798	0.717	0.821		0.254	0.611	**0.999**	0.610	0.746	47	82
U-Eff.Netv2B3	0.171	0.900	0.801	0.730	0.829	840	0.313	0.864	0.597	0.541	0.685	77	50
U-TinyNet-e	0.196	0.881	0.776	0.693	0.804		0.221	0.792	0.801	0.656	0.782	47	92
U-Eff.Netv2B3	0.169	0.884	0.816	0.732	0.831	800	0.210	0.845	0.747	0.658	0.782	76	50
U-MNASNet-s	0.203	0.884	0.763	0.684	0.797		0.218	0.767	0.838	0.665	0.790	50	93

3.2. Comparison of FTL with Baseline, KD, and DML

After reporting the individual performances of FTL, KD, and DML, the three methods are compared in terms of performance, training time, and computational efficiency.

3.2.1. Performance Comparison

The performance of FTL against standalone models (baseline), KD, and DML is summarized for the two top-performing students in S–S and S–Ev settings: U-VGG19 and U-EfficientNetv2B3. The comparison, illustrated with radar charts in Figure 5, highlights that FTL consistently outperformed baseline scores and other transfer learning methods across both settings.

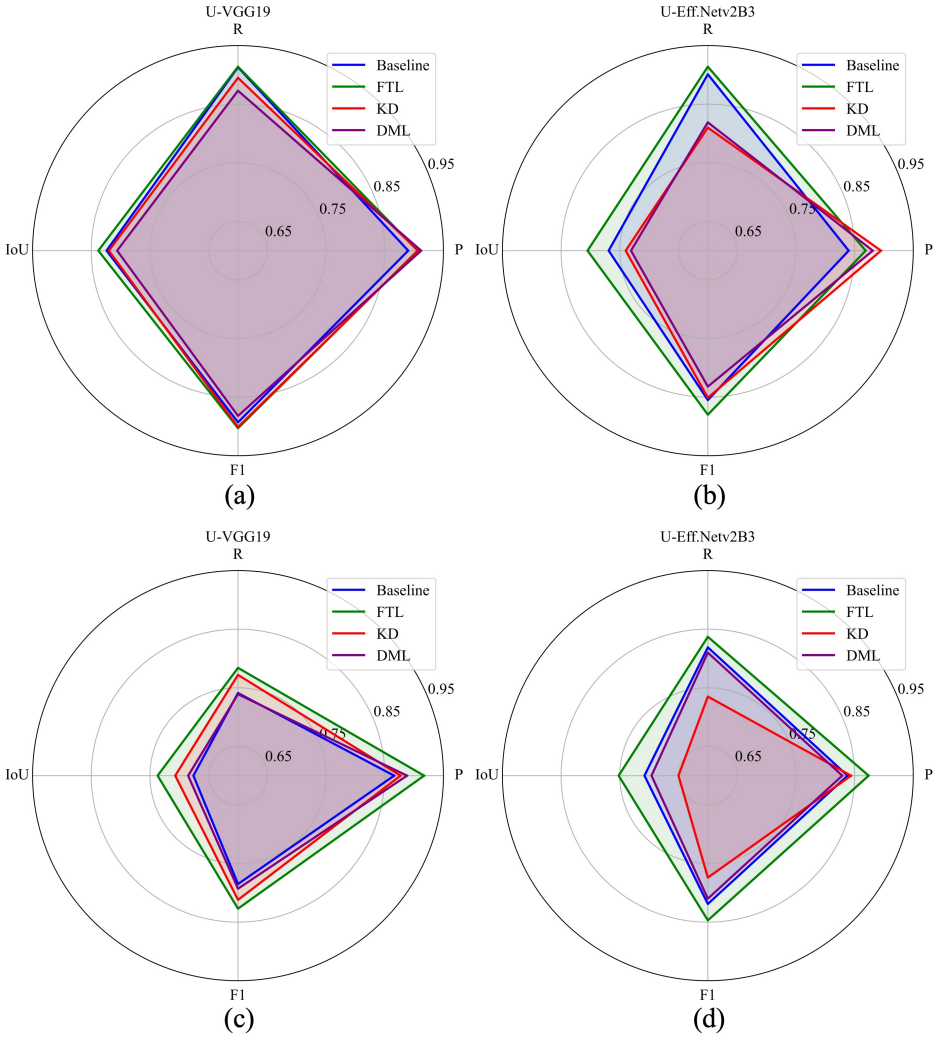

Figure 5. Summary with the two best-performing students (U-VGG19 and U-EfficientNetv2B3) across standalone training (baseline), FTL, KD, and DML. (**a**) U-VGG19 in S–S setting (FTL results in highest F1 of 0.903). (**b**) U-EfficientNetv2B3 in S–S setting. (**c**) U-VGG19 in S–Ev setting. (**d**) U-EfficientNetv2B3 in S–Ev setting (FTL results in highest F1 of 0.819).

In the S–S setting, FTL with U-VGG19 achieved the highest F1 score of 0.903, followed by KD with U-VGG19 (0.901 F1). In the S–Ev setting, FTL with U-EfficientNetv2B3 obtained the highest F1 score of 0.847, followed by its standalone training (baseline) at 0.819 F1. KD provided a slight improvement over baseline scores with U-VGG19 as the student, but since it is a self-distillation, it did not offer any model compression. In contrast, KD with U-EfficientNetv2B3 provided a 50% reduction in network parameters (29.1 M vs. 14.6 M), but

the performance trade-off was minimal (F1: 0.851 vs. 0.855 for KD vs. Baseline). Similarly, DML with U-EfficientNetv2B3 showed only a marginal difference from the baseline (F1: 0.832 vs. 0.855).

In the S–Ev setting, FTL remained the best-performing approach, outperforming the baseline and both KD and DML for U-VGG19 and U-EfficientNetv2B3. While all three methods outperformed baseline scores with U-VGG19, none provided model compression. With U-EfficientNetv2B3, KD led to a 4.5% F1 reduction (0.774 vs. 0.819), while DML resulted in a smaller 0.9% drop (0.810 vs. 0.819).

Overall, U-EfficientNetv2B3 was the best student for FTL, though it did not achieve model compression. KD and DML enabled compression but at the cost of reduced performance. U-EfficientNet-lite0 proved to be the best student for KD, reducing network parameters by 82% (from 29.1 M to 5.6 M) compared with the teacher (U-VGG19). With DML, only a few student pairs improved both models' scores. These findings indicate that FTL is the most effective method for mitigating the impact of misaligned building labels, significantly outperforming KD and DML in both accuracy and consistency.

3.2.2. Analysis of Training Time and Computational Efficiency

The training times (ms/it) for different student models under FTL, standalone training, and KD are shown in Figure 6, while DML is excluded due to its training time dependence on the number and size of distilled students. KD incurred the highest training time across all models due to the additional overhead of aligning student predictions with the teacher's soft labels, significantly increasing processing time per iteration. In contrast, standalone training was the fastest, as it trained models from scratch without prior knowledge transfer. FTL remained computationally efficient, achieving training times comparable to standalone training while leveraging pre-trained weights. The computational time could be reduced if fewer layers were unfrozen during the fine-tuning. The difference between standalone and FTL was minimal, as seen with U-TinyNet-e (181 ms/it in both) and U-MobileViT-xxs (294 ms in FTL vs. 296 ms in standalone), indicating fine-tuning introduced negligible overhead while retaining transfer learning benefits.

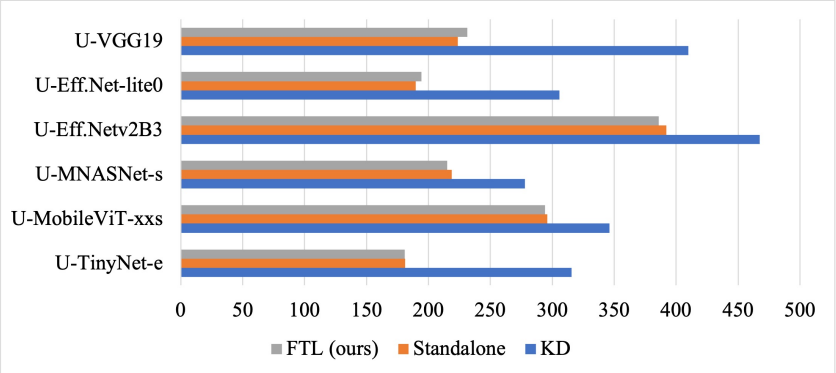

Figure 6. Analysis of training time and computational efficiency for FTL, standalone training, and KD across different student models.

DML required significantly longer training times than all other methods, as shown in Table 6. The computational costs in DML increase based on the number of students and model sizes. Unlike KD and FTL, which involve only a single teacher–student relationship, DML's complexity scaled quadratically (compared with the standalone training and FTL), making it the least efficient for applications where training time is a constraint. It also required more memory and GPU resources for training, as it had to load all students, the teacher, and the training data simultaneously.

Therefore, FTL is the most computationally efficient transfer learning method among the compared, maintaining near-standalone training speeds while benefiting from pre-trained models. KD incurred significant computational costs, while DML was the least efficient due to its model interactions. FTL offered the best balance between efficiency and transfer learning benefits, making it ideal for lightweight U-Net segmentation models in resource-constrained settings.

3.3. Ablation Study: With FTL vs. Without FTL

The performance gains from FTL compared with the baseline (without fine-tuning) are presented in Table 7, showing improvements in IoU and F1 scores across all students. FTL consistently improved IoU (up to 6.1%) and F1 scores (up to 4.2%) for all models in the Ev dataset. U-VGG19 and U-EfficientNetv2B3 produced the highest scores in S–S and S–Ev settings, respectively, showing the effectiveness of fine-tuning.

Table 7. Improvements with FTL vs. without FTL for the students with different encoders on S and Ev datasets. The gains after FTL are shown in percentage, and the highest scores are highlighted in bold.

Student	S–S		S–Ev	
	IoU	**F1**	**IoU**	**F1**
U-VGG19	**0.838 (+1.4%)**	**0.903 (+1.0%)**	0.737 (+6.1%)	0.827 (+4.2%)
U-Eff.Net-lite0	0.778 (+2.4%)	0.862 (+1.7%)	0.688 (+0.1%)	0.803 (−0.2%)
U-Eff.Netv2B3	0.805 (+3.6%)	0.880 (+2.5%)	**0.752 (+4.4%)**	**0.847 (+2.8%)**
U-MNASNet-s	0.759 (+4.9%)	0.849 (+3.3%)	0.690 (+5.1%)	0.795 (+3.0%)
U-MobileViT-xxs	0.769 (+5.2%)	0.855 (+3.6%)	0.694 (+6.0%)	0.809 (+4.2%)
U-TinyNet-e	0.748 (+2.1%)	0.840 (+1.4%)	0.653 (+0.5%)	0.779 (+0.0%)

A visual comparison of segmentation outputs before and after FTL (Figure 7) further highlights the advantages of fine-tuning. FTL reduced over-segmentation and minimized outliers, leading to cleaner building extractions and improved alignment with input labels. These findings confirm that fine-tuning significantly improves segmentation accuracy compared with standalone training.

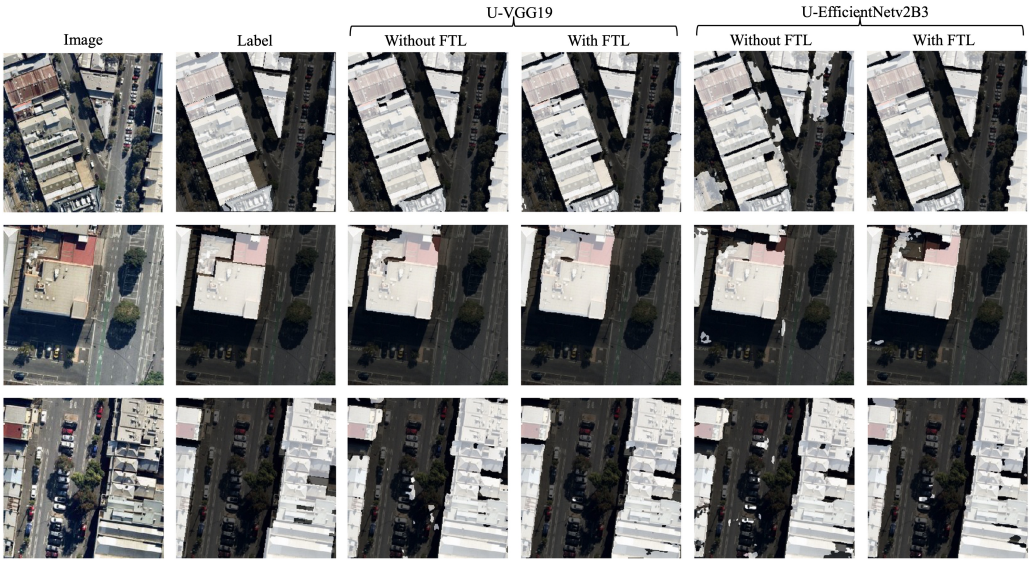

Figure 7. The segmented outputs of the U-VGG19 and U-EfficientNetv2B3 without FTL and with FTL on the data samples of the Ev dataset.

4. Discussion

This section further dissects the results from the previous section and provides a discussion on how the proposed method compares to KD and DML for different building heights and different spatial resolutions of remote sensing images. Furthermore, the limitations and opportunities are discussed.

4.1. Investigation of Different Building Heights

The effects of building heights were studied by separating the image scenes of Ev into four categories of building types: low-rise, mid-rise, high-rise, and skyscrapers. The chart in Figure 8 shows the performance of FTL as compared with the teacher, KD, and DML. FTL outperformed the teacher, KD, and DML, with the highest scores for low-rise (0.943 F1), mid-rise (0.868 F1), and skyscrapers (0.697 F1). Both student and teacher networks yielded F1 and average scores above 90% for low-rise buildings. The performance degraded with the increased height of the buildings.

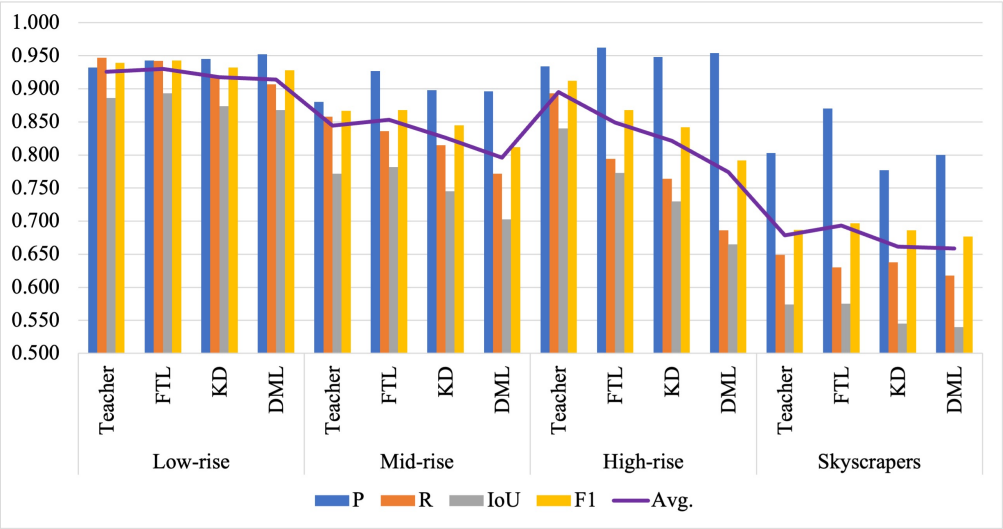

Figure 8. The segmentation performance of the teacher, FTL, KD, and DML using the U-VGG19 network on four types of buildings: low-rise, mid-rise, high-rise, and skyscrapers. The y-axis shows the scores P, R, IoU, F1, and their average (denoted by 'Avg.').

A mean difference (MD) of the four performance scores (difference in 'Avg.' from Figure 8) is further examined to observe the clear differences between our FTL versus the teacher, KD, and DML. The MD is plotted in Figure 9. It indicates that FTL outperforms KD and DML across all building types, with the most significant improvements seen in mid-rise and skyscraper buildings. For low-rise structures, FTL showed only a marginal improvement (0.4–1.7%) over other methods, suggesting that fine-tuning was not impactful for simpler buildings. In mid-rise buildings, FTL provided a moderate gain over KD (2.8%) and a notable advantage over DML (5.8%), making it a better alternative in moderately complex urban settings. For high-rise buildings, the teacher performed better than the FTL (−4.6%), meaning fine-tuning did not enhance performance for these buildings. However, FTL still outperformed KD (2.8%) and DML (7.5%), confirming its effectiveness compared with KD and DML. The greatest improvements are seen in skyscrapers, where FTL significantly outperformed all methods, with gains over teacher (1.5%), KD (3.2%), and DML (3.4%), showing its robustness in handling extreme building heights and off-nadir distortions. These results suggest that FTL is most beneficial for complex urban environments, particularly for taller buildings.

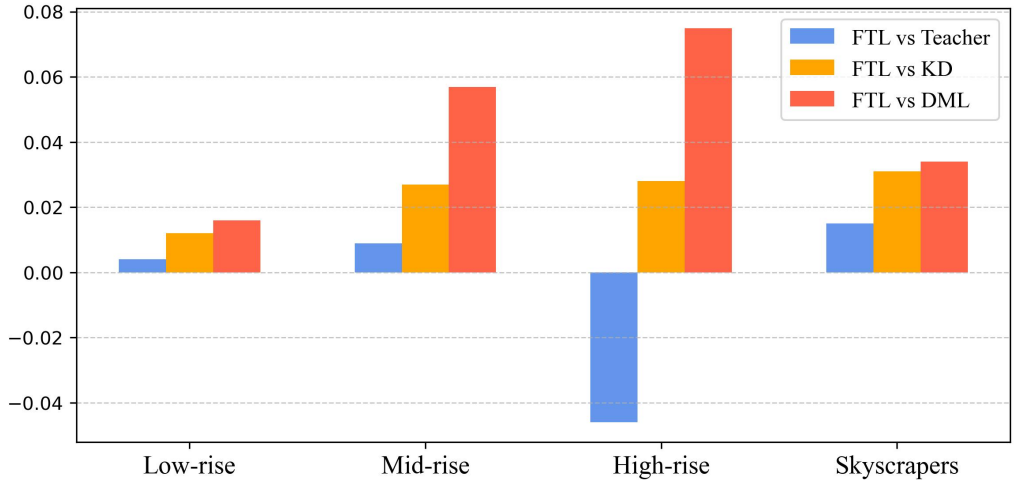

Figure 9. Mean difference of performance score ('Avg.') to examine the differences between FTL, teacher (standalone), KD, and DML for different building types.

4.2. Investigation of Different Spatial Resolutions

The study was further dissected in terms of the spatial resolution of the images. Figure 10 shows sample output from each method with images of three spatial resolutions for the four building types. For the low-rise buildings, FTL increased the generalization of the buildings in all three resolutions and achieved the highest F1 of 0.943. The green boxes highlight the network's ability in instance segmentation of buildings. The blue boxes in the 30 cm/px images show that FTL could separate the roofs from the facades. KD and DML over-segmented the images. For mid-rise, the performance was lower than low-rise, with FTL showing some improvements. The blue boxes in the 60 cm/px image sample show FTL's ability to segment out the facades and the gaps between the buildings. For the high-rise, the teacher performed with the highest evaluation scores despite the noisy training data. However, it was seen that FTL could separate roofs from the facades with better segmentation. For skyscrapers (outlined with orange boxes), the segmentation was poor. FTL produced the highest scores but with increased noise. The 30 cm/px sample in Figure 10 shows the segmentation of the tallest skyscraper in Melbourne (319 m high).

The spatial resolution of the images affected the segmentation of the four building types differently. From Figure 10, the students performed poorly on the highest resolution (30 cm/px) images. Low-rise buildings were too small to segment accurately, and high-rise buildings and skyscrapers were too large to fit into the image tiles. The best visual results were achieved at the resolution of 60 cm/px, except for skyscrapers, for which the lowest resolution of 120 cm/px provided the best segmentation.

4.3. Limitations and Opportunities

FTL proved most beneficial for complex urban environments as it outperformed other transfer learning methods. Figure 11 (sample on top) shows the considerably accurate segmentation output from the two best students using the FTL approach. The students could produce relatively precise building segments despite the presence of highly dense and connected buildings. Moreover, they also precisely segmented buildings with multiple roofs and smaller objects.

However, segmentation challenges arose in taller buildings (high-rises and skyscrapers) due to shadows, occluded buildings, reflections from glass facades, rooftop objects, and label misalignment. This limitation is shown in the bottom half of Figure 11, where even the top-performing models struggled to segment skyscrapers. While FTL reduced

over-segmentation, further improvements could be achieved by integrating LiDAR or by learning multiple features of oblique-view images, such as footprints, roofs, and facades.

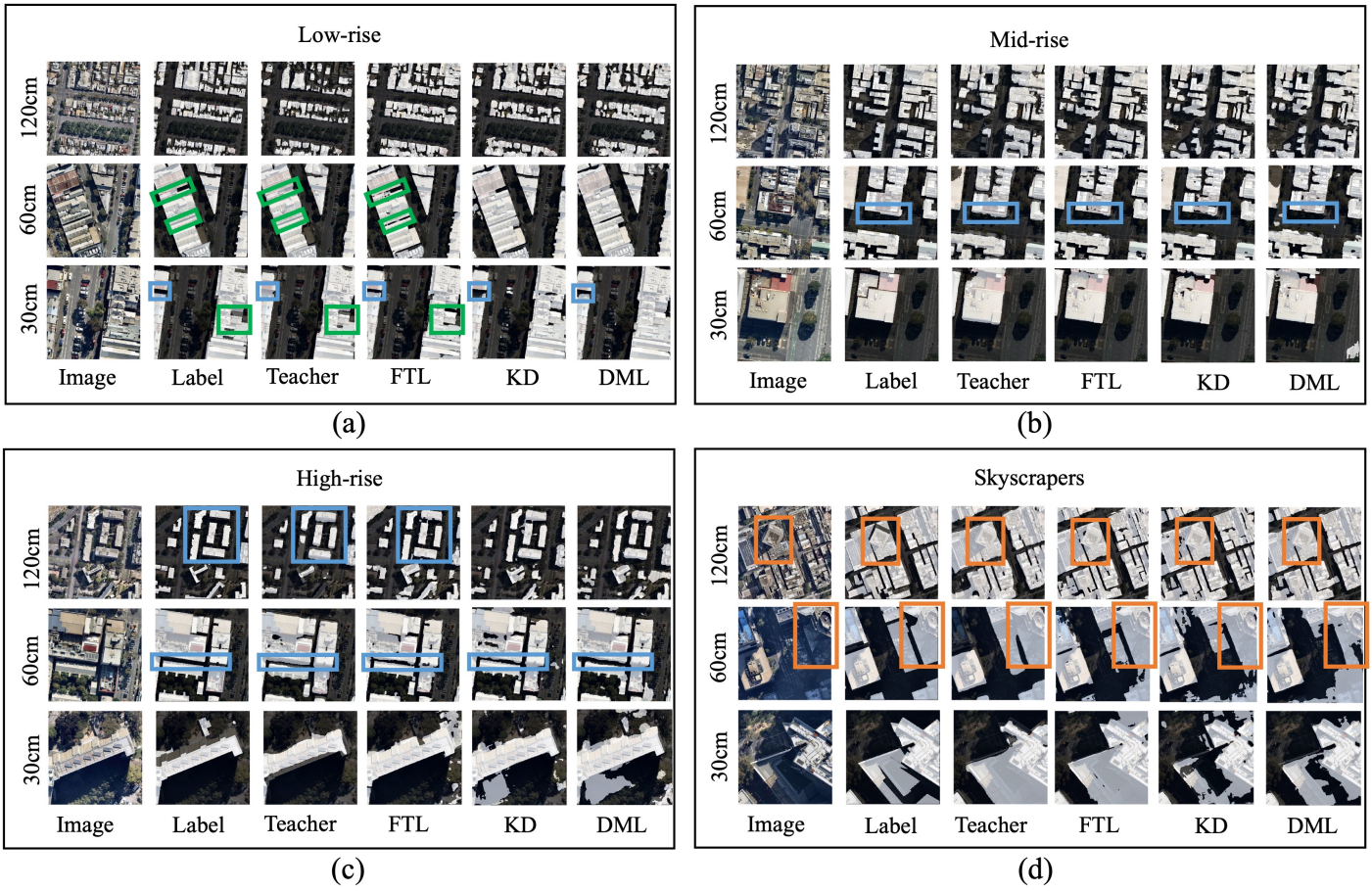

Figure 10. Segmentation results for four building types: (**a**) low-rise, (**b**) mid-rise, (**c**) high-rise, and (**d**) skyscrapers, with further analysis across three spatial resolutions (120, 60, and 30 cm/px). Green rectangles outline the student networks' ability to distinguish individual buildings. Blue rectangles outline instances where the models attempt to exclude building facades. Orange rectangles mark skyscrapers, with each 30, 60, and 120 cm/px image featuring buildings of heights 177 m, 251 m, and 319 m (the tallest in Melbourne), respectively.

The limitations of the compared methods, KD and DML, were more than the inherent inability of student models to surpass the teacher. They were less effective in tackling the label displacement error, and implementing them required storing multiple networks in memory, posing deployment challenges. FTL was more memory-efficient and easier to integrate into operational systems. The limitation of FTL in terms of implementation is that it depends on a small but high-quality clean dataset, which may not always be available. Most benchmark datasets contain noise and label displacement errors [45]. This can lead to model overfitting, as observed in our benchmarks with FPN-VGG19, PSPNet-VGG19, and FPN-EfficientNetv2L. Expanding benchmarks is crucial to identifying the best trade-off between lightweight and high-performance models. Furthermore, FTL requires testing on diverse geographic datasets with varying architectural styles, climate conditions, and sensor data sources (e.g., New York, Tokyo, and rural areas) to assess its generalizability. Beyond building extraction, FTL could be adapted for flood damage assessment, post-earthquake building extraction, and other remote sensing applications in resource-constrained environments, enhancing its practical impact.

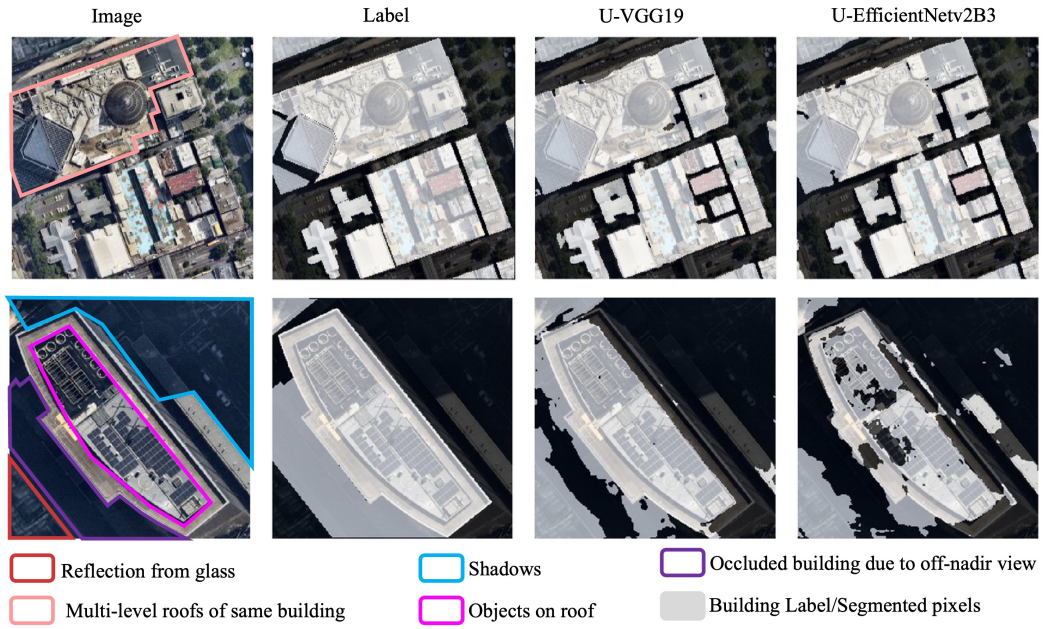

Figure 11. Example illustrating the performance of the best student models (U-VGG19 and U-EfficientNetv2B3) in segmenting buildings in complex urban environments of the Ev dataset. The image–label pair and segmented results highlight the accuracy of segmentation in a complex urban scene with buildings having multiple roofs (Melbourne Central Plaza in this example). The image–label pair and segmented results at the bottom highlight the segmentation challenges due to skyscrapers.

5. Conclusions

This study proposed fine-tuning-based transfer learning as an effective method for building extraction from remote sensing imagery, specifically addressing the misalignment problem (label displacement errors) caused by off-nadir images. Unlike the existing methods of KD and DML, which primarily focus on model compression, FTL leverages the teacher–student learning framework to fine-tune a pre-trained model on a smaller, high-quality dataset. This approach allows better adaptation to the target domain without compromising segmentation accuracy, making it a more robust alternative to the previous transfer learning methods.

A structured experimental framework was designed with three new urban building datasets that incorporate both large–noisy (misaligned) and small–clean (manually annotated) image–label pairs. This setup enabled a controlled evaluation of FTL, KD, and DML across different building types and spatial resolutions. Additionally, an extensive benchmark study was conducted, evaluating 43 representative lightweight CNNs, five optimizers, nine loss functions, and seven encoder–decoder networks (EDNs) to identify the optimal balance between lightweight efficiency and high segmentation accuracy. The benchmark found that U-VGG19 and U-EfficientNetv2B3 are the best (in terms of accuracy, training time, and model size) teacher and student models.

The results demonstrated that FTL consistently outperformed standalone training (baseline), KD, and DML, achieving higher F1 and IoU scores across different building types and spatial resolutions. Among the student models, U-VGG19 achieved the highest F1 (0.903) in the S–S setting, while U-EfficientNetv2B3 performed best in the S–Ev setting (0.847 F1), suggesting its suitability for generalizing to unseen datasets. KD and DML provided some improvements in model efficiency but at the cost of reduced segmentation accuracy. Additionally, FTL proved computationally efficient, introducing minimal overhead compared with standalone training, while KD incurred significant computational

costs, and DML was the least efficient due to its quadratic increase in complexity when distilling multiple students.

Further investigations revealed that FTL was particularly effective in complex urban environments, especially for skyscrapers, where it significantly outperformed KD and DML in handling off-nadir distortions and misaligned building labels. However, high-rise buildings remained challenging, with the teacher sometimes performing better than fine-tuned models. The analysis of spatial resolution further highlighted that 60 cm/px imagery provided the best balance for building extraction, and 120 cm/px images were more suitable for skyscrapers. The 30 cm/px images introduced intricate noise (over-segmentation and outliers) for small buildings as they were too small for accurate segmentation, and the high-rises and skyscrapers were too large to fit in a single image tile.

These findings highlight the potential of FTL as a robust method for accurate building extraction, particularly in off-nadir images of urban environments. The approach can be extended to other remote sensing applications, such as flood damage assessment, post-disaster building extraction, and large-scale urban monitoring, where accurate segmentation under variable conditions is needed. Future research could explore self-supervised learning to reduce reliance on clean labels, test FTL across diverse geographic regions, and investigate multi-feature learning for enhanced segmentation of oblique-view images.

Author Contributions: Conceptualization, B.N.; methodology, B.N.; software, B.N.; validation, B.N. and J.A.; formal analysis, B.N.; investigation, B.N.; resources, B.N.; data curation, B.N.; writing—original draft preparation, B.N.; writing—review and editing, B.N., J.A. and A.R.; visualization, B.N.; supervision, J.A. and A.R.; project administration, J.A.; funding acquisition, B.N. and J.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The codes to access and reproduce the datasets are available at: https://github.com/bipulneupane/TeacherStudentBuildingExtractor (accessed on 30 March 2025).

Acknowledgments: The first author (B.N.) is supported by the University of Melbourne for his PhD research and is awarded the Melbourne Research Scholarship. The experimental results were produced using the LIEF HPC-GPGPU Facility hosted at the University of Melbourne, which was established with the assistance of LIEF Grant LE170100200. The authors would like to thank Nearmap for providing the API service to collect the image data for the experiments.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Benchmarking Network Architectures for Teacher and Student Models

Appendix A.1. Hyperparameter Search

The optimal hyperparameters were searched with a randomly selected MobileOne-s1 CNN encoder for U-Net EDN (abbr. U-MobileOne-s1). Then, using the loss function of total loss, five optimizers—Adam, stochastic gradient descent (SGD), RMSprop, Adadelta, and Nadam—were compared with find the optimizer that yields the highest evaluation scores. The total loss was formulated as the sum of dice loss and binary focal loss. The RMSProp optimizer produced the highest scores for our experimental settings with a trade-off between loss and faster-converging epochs. Then, using RMSProp, nine loss functions—binary cross entropy (BCE), BCE dice loss, BCE Jaccard loss, binary focal dice loss, binary focal Jaccard loss, Jaccard loss, dice loss, binary focal loss, and total loss—were further compared. A combination of RMSProp optimizer and dice loss yielded the highest scores, as shown in Table A1. Adam was marginally close to RMSProp optimizer. Jaccard loss produced the same IoU and F1 with a faster-converging epoch compared with dice loss but yielded a lower loss. The learning rate of 0.0001 produced the highest scores in

most experiments. Therefore, the selected hyperparameters are RMSProp optimizer, dice loss, and 0.0001 learning rate.

Table A1. Hyperparameter search using a U-Net with randomly chosen lightweight CNN of MobileOne-s1. Five optimizers are tested with a total loss (dice loss + binary focal loss). The optimizer with the highest F1 score is then used to test the nine loss functions. Columns 'ms/it' (millisecond/iteration) and 'Ep' report training time and converging epoch. Boldface: best performance.

	Loss	P	R	IoU	F1	ms/it	Ep.
Optimizers (with Total Loss)							
Adam	**0.177**	**0.959**	0.963	0.926	0.960	376	26
SGD	0.338	0.889	**0.990**	0.882	0.935	**200**	28
RMSProp	**0.177**	0.958	0.964	**0.927**	**0.961**	294	22
Adadelta	0.444	0.927	0.784	0.744	0.845	420	29
Nadam	0.179	0.957	0.965	**0.927**	**0.961**	395	**20**
Loss Functions (with RMSProp optimizer)							
BCE	0.200	0.950	0.973	0.927	0.961	658	26
BCE Dice	0.243	0.949	0.975	**0.928**	0.961	658	15
BCE Jaccard	0.285	0.946	**0.978**	0.927	0.961	312	14
Binary Focal Dice	0.191	0.951	0.966	0.922	0.958	324	**7**
Binary Focal Jaccard	0.217	0.955	0.967	0.926	0.960	292	26
Jaccard	0.072	0.953	0.972	**0.928**	**0.962**	**282**	18
Dice	**0.039**	0.948	0.977	**0.928**	**0.962**	314	20
Binary Focal	0.120	**0.975**	0.896	0.876	0.932	292	30
Total Loss	0.177	0.958	0.964	0.927	0.961	294	22

Appendix A.2. EDN Comparison

With the best hyperparameters found, we further used the U-MobileOne-s1 with RMSProp optimizer and dice loss to investigate the other EDNs. The results are shown in the first half of Table A2. Results indicated that U-Net++ achieves the highest scores (0.964 F1), closely followed by U-Net, which offered faster training and lower network parameters. Compared with U-Net++, U-Net provided the best trade-off between network parameters (9.1 M vs. 12.3 M) and evaluation scores with a marginally lower F1 of 0.962 and less complexity in network architecture. Opting for U-Net's slightly lower evaluation scores, we prioritized lower parameters and faster training for small deployable students. Fewer parameters mean a smaller model size and faster training time, which is essential for students in knowledge transfer. Among the other EDNs, MANet matched U-Net in IoU and F1 but with nearly 6x more parameters. LinkNet and DeepLabv3+ were lightweight but yielded lower scores. FPN and PSPNet, despite faster training, overfit on the dataset. U-Net is, therefore, the best EDN configuration for students.

Appendix A.3. Student Search

After identifying U-Net as the best-performing EDN, 43 representative lightweight CNNs identified from the computer vision domain were integrated into U-Net for the student search. Thus developed U-Net configurations of the 43 CNNs are shown in the second half of Table A2. The high scorer was U-EfficientNetv2B3 (0.965 F1), U-EfficientNet-lite0 provided the best trade-off between evaluation scores and network parameters, and the lightest students were U-MNASNet-s (2.2M), U-TinyNet-e (2.3M), and U-MobileViT-xxs (3.1M). These EDNs were, therefore, selected as the top five students for the rest of the study. Figure A1 shows the segmentation outputs from these five students on two random samples from the validation subset of the Massachusetts building dataset. The trend showed that the segmentation performance decreases with the number of network parameters.

Table A2. Search of EDN and CNN encoders for student networks. Eight EDNs are first tested with a randomly chosen MobileOne-s1 CNN to identify the EDN with the best trade-off between network parameters and evaluation scores. The best performance scores among the EDNs are highlighted in bold. This EDN is then used to test and benchmark the 43 identified lightweight CNNs. Network parameters (Par.) are shown in Millions. The top three scores for the lightweight CNNs are underlined and ranked in order from one to three with a postscript.

	Par.	Loss	P	R	IoU	F1	ms/it
EDN with MobileOne-s1 encoder							
U-Net	9.1	0.039	0.948	0.977	0.928	0.962	314
DeepLabv3+	5.7	0.044	0.937	**0.979**	0.919	0.956	287
LinkNet	6.2	0.042	0.945	0.973	0.922	0.958	326
MANet	53.3	0.039	0.948	0.976	0.928	0.961	376
U-Net++	12.3	**0.036**	0.954	0.975	**0.933**	**0.964**	395
FPN	5.7	0.062	0.887	1.000	0.887	0.938	309
PSPNet	**3.8**	0.062	0.887	1.000	0.887	0.938	**168**
U-Net (best trade-off of Par. and F1) with lightweight CNNs as encoder							
ResNet-18	14.3	0.038	0.949	0.977^{3}	0.929	0.962	109
DenseNet-121	13.6	0.037^{3}	0.950	0.977^{3}	0.931^{3}	0.963^{3}	629
SE-ResNet-18	14.4	0.039	0.950	0.974	0.927	0.961	151^{1}
MobileNet-v2	6.6	0.039	0.951	0.972	0.927	0.961	201
MobileNetv3-s	3.6	0.044	0.940	0.974	0.918	0.956	281
MobileNetv3-l	6.7	0.043	0.946	0.970	0.921	0.957	316
Eff.NetB0	6.3	0.038	0.952^{3}	0.973	0.929	0.962	304
Eff.NetB1	8.8	0.038	0.950	0.976	0.930	0.962	461
Eff.NetB2	10.0	0.037^{3}	0.956^{1}	0.971	0.931^{3}	0.963^{3}	459
Eff.NetB3	13.2	0.037^{3}	0.951	0.977^{3}	0.930	0.963^{3}	485
Eff.Net-lite0	5.6	0.039	0.950	0.974	0.928	0.962	187
Eff.Net-lite1	6.4	0.039	0.948	0.976	0.928	0.961	233
Eff.Net-lite2	7.2	0.039	0.948	0.975	0.927	0.961	229
Eff.Net-lite3	9.4	0.038	0.948	0.978^{2}	0.929	0.962	258
Eff.Net-lite4	14.4	0.040	0.945	0.978^{2}	0.926	0.960	313
Eff.Netv2B0	7.6	0.038	0.951	0.975	0.929	0.962	284
Eff.Netv2B1	8.6	0.037^{3}	0.953^{2}	0.975	0.931^{3}	0.963^{3}	334
Eff.Netv2B2	10.4	0.037^{3}	0.951	0.977^{3}	0.931^{3}	0.963^{3}	347
Eff.Netv2B3	14.6	0.035^{1}	0.956^{1}	0.975	0.934^{1}	0.965^{1}	408
SK-ResNet-18	14.6	0.038	0.948	0.978^{2}	0.929	0.962	201
DPN-68	17.0	0.041	0.945	0.975	0.924	0.959	493
ResNeSt-14	17.6	0.036^{2}	0.952^{3}	0.977^{3}	0.932^{2}	0.964^{2}	159^{2}
GERNet-s	12.8	0.040	0.947	0.974	0.925	0.960	180^{3}
MobileOne-s0	8.6	0.038	0.948	0.979^{1}	0.930	0.963	610
MobileOne-s1	9.1	0.039	0.950	0.975	0.928	0.962	287
MobileOne-s2	13.6	0.039	0.949	0.975	0.928	0.961	296
MobileOne-s3	16.2	0.039	0.950	0.973	0.927	0.961	293
HRNet-18	16.1	0.038	0.952^{3}	0.974	0.929	0.962	870
MNASNet-s	2.2^{1}	0.043	0.938	0.978^{2}	0.920	0.957	214
MobileViT-s	8.0	0.040	0.946	0.975	0.925	0.960	289
MobileViT-xs	4.3	0.040	0.947	0.975	0.926	0.960	294
MobileViT-xxs	3.1^{3}	0.042	0.943	0.975	0.922	0.958	299
FBNet-c100	5.2	0.041	0.947	0.973	0.924	0.959	229
FBNet-v3b	8.6	0.042	0.943	0.974	0.921	0.958	690
FBNet-v3d	10.3	0.041	0.943	0.978^{2}	0.925	0.960	769
FBNet-v3g	16.8	0.040	0.945	0.976	0.925	0.960	917
HardCoRe-NAS-a	6.5	0.042	0.940	0.979^{1}	0.922	0.958	283
HardCoRe-NAS-f	9.4	0.043	0.937	0.980	0.920	0.957	552
MixNet-s	4.3	0.044	0.943	0.970	0.918	0.956	429
MixNet-m	5.2	0.043	0.943	0.974	0.921	0.957	498
MixNet-l	7.6	0.041	0.948	0.972	0.924	0.959	532
TinyNet-a	6.7	0.042	0.944	0.975	0.923	0.959	296
TinyNet-e	2.3^{2}	0.045	0.934	0.979^{1}	0.916	0.955	182

Appendix A.4. Teacher Search

For FTL, the teacher and student constitute the same EDN network. For KD and DML, the best teacher was identified from the comparison of VGG19 (top scorer from [5]) and EfficientNetv2L (largest version of the best-performing CNN of EfficientNetv2B3) with eight EDNs. The heatmaps and the parallel coordinate plots in Figure A2 provide minute

details to identify the most optimal trade-off from the networks. With VGG-19, three EDNs of U-Net, LinkNet, and U-Net++ achieved the highest average of the four accuracy metrics (0.960) and an F1 of 0.967. However, with EfficientNetv2L, only U-Net yielded the highest F1 score, higher than DeepLabv3+, LinkNet, MANet, and U-Net++. FPN and PSPNet provided overfitting results (with recall up to 1.000). LinkNet provided the best trade-off with VGG-19, while U-Net provided that with the highest F1 and lower parameters with both CNNs. Therefore, U-VGG19 is the teacher for our transfer learning study.

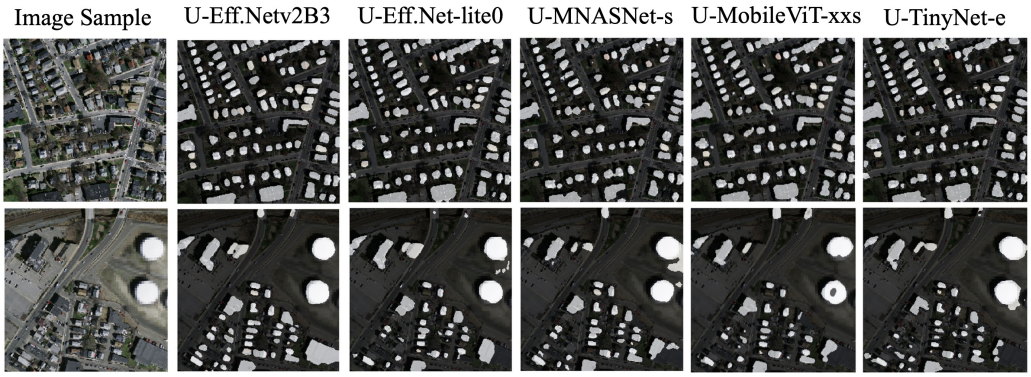

Figure A1. Segmentation results of the selected lightweight students on the subset of Massachusetts building dataset. U-EfficientNetv2B3 produced the highest F1 score of 0.965. Other students produced lower scores but with trade-offs between network parameters, training time, loss, and evaluation scores as seen in Table A2.

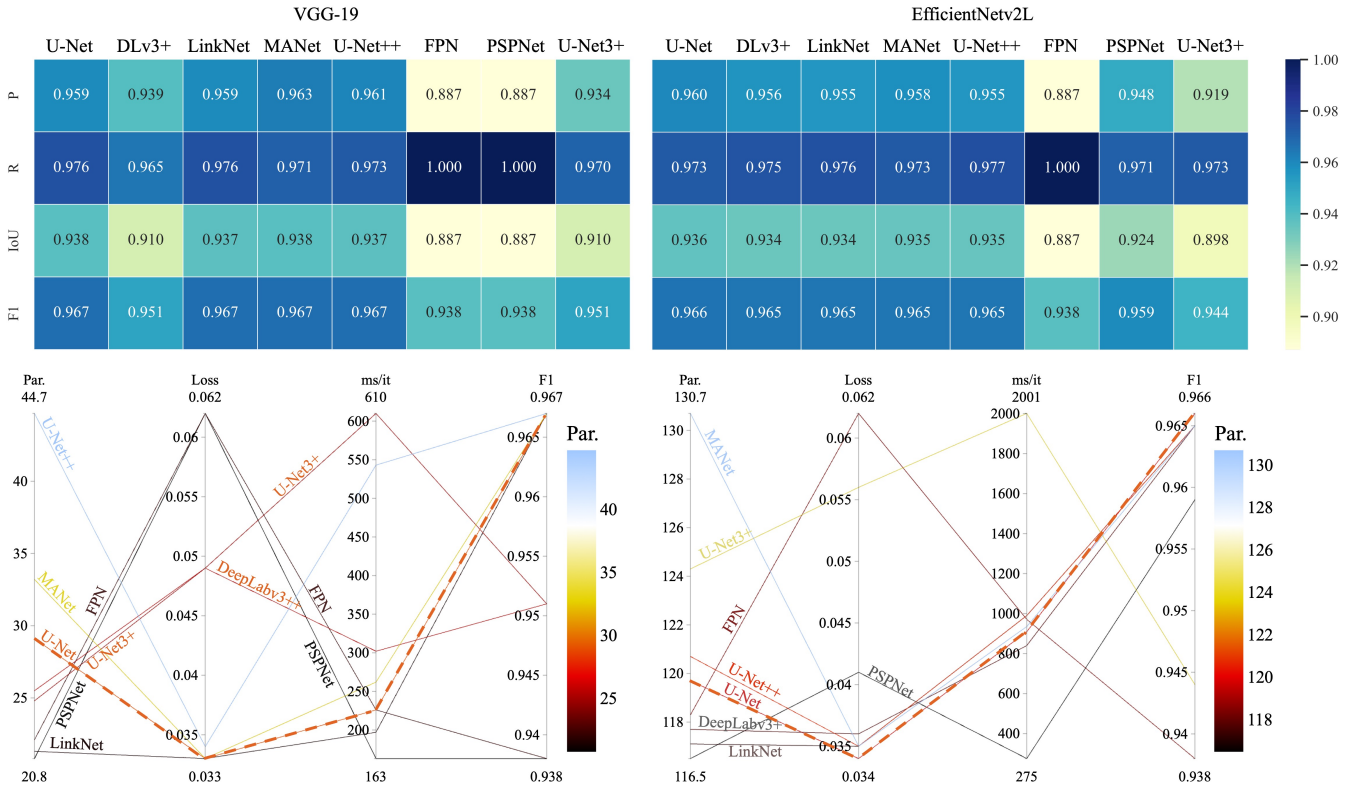

Figure A2. Teacher search with VGG-19 (left) and EfficientNetv2B3 (right) as CNNs for the state-of-the-art EDNs. The heat maps on top compare the evaluation scores of the EDNs. The parallel coordinate plots on the bottom show the trade-off between network parameters (Par.), loss, training time (ms/it), and F1 score. The network with the dashed line (U-Net) provides the best trade-off among the variables.

References

1. Chauhan, A.; Wasim, M.; Mohanty, S.; Pandey, P.C.; Pandey, M.; Maurya, N.K.; Rankavat, S.; Dubey, S.B. Earth observation applications for urban mapping and monitoring: Research prospects, opportunities and challenges. *Earth Obs. Urban Monit.* **2024**, 197 –229. [CrossRef]
2. Camacho, R.; Aryal, J.; Rajabifard, A. How do disasters disrupt the spatial growth of informal settlements? A multi-temporal remote sensing approach–The case study of Mocoa, Colombia. *Habitat Int.* **2025**, *156*, 103272.
3. Herfort, B.; Lautenbach, S.; Porto de Albuquerque, J.; Anderson, J.; Zipf, A. The evolution of humanitarian mapping within the OpenStreetMap community. *Sci. Rep.* **2021**, *11*, 3037.
4. Li, J.; Huang, X.; Tu, L.; Zhang, T.; Wang, L. A review of building detection from very high resolution optical remote sensing images. *Gisci. Remote Sens.* **2022**, *59*, 1199–1225.
5. Neupane, B.; Aryal, J.; Rajabifard, A. CNNs for remote extraction of urban features: A survey-driven benchmarking. *Expert Syst. Appl.* **2024**, *255*, 124751.
6. Amirgan, B.; Erener, A. Semantic segmentation of satellite images with different building types using deep learning methods. *Remote Sens. Appl. Soc. Environ.* **2024**, *34*, 101176.
7. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Proceedings of the Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, 5–9 October 2015; Proceedings, Part III 18; Springer: Berlin/Heidelberg, Germany, 2015; pp. 234–241.
8. Huang, X.; Wang, C.; Li, Z.; Ning, H. A 100 m population grid in the CONUS by disaggregating census data with open-source Microsoft building footprints. *Big Earth Data* **2021**, *5*, 112–133. [CrossRef]
9. Sirko, W.; Kashubin, S.; Ritter, M.; Annkah, A.; Bouchareb, Y.S.E.; Dauphin, Y.; Keysers, D.; Neumann, M.; Cisse, M.; Quinn, J. Continental-Scale Building Detection from High Resolution Satellite Imagery. *arXiv* **2021**, arXiv:2107.12283. [CrossRef]
10. Richards, J.A.; Richards, J.A. *Remote Sensing Digital Image Analysis*; Springer: Berlin/Heidelberg, Germany, 2022; Volume 5, pp. 256–258.
11. Weir, N.; Lindenbaum, D.; Bastidas, A.; Etten, A.V.; McPherson, S.; Shermeyer, J.; Kumar, V.; Tang, H. Spacenet mvoi: A multi-view overhead imagery dataset. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 992–1001.
12. Chen, Q.; Zhang, Y.; Li, X.; Tao, P. Extracting Rectified Building Footprints from Traditional Orthophotos: A New Workflow. *Sensors* **2021**, *22*, 207. [CrossRef]
13. Li, B.; Gao, J.; Chen, S.; Lim, S.; Jiang, H. POI detection of high-rise buildings using remote sensing images: A semantic segmentation method based on multitask attention Res-U-Net. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 1–16.
14. Neupane, B.; Aryal, J.; Rajabifard, A.; Pelizari, P.A.; Geiß, C. Multi-label Learning with ViT for Building Footprint Extraction from Off-Nadir Aerial Images. *IEEE Geosci. Remote Sens. Lett.* **2025**, *22*, 1–5. [CrossRef]
15. Zhu, Q.; Huang, S.; Hu, H.; Li, H.; Chen, M.; Zhong, R. Depth-enhanced feature pyramid network for occlusion-aware verification of buildings from oblique images. *ISPRS J. Photogramm. Remote Sens.* **2021**, *174*, 105–116. [CrossRef]
16. Yan, Y.; Qi, Y.; Xu, C.; Su, N.; Yang, L. Building Extraction at Amodal-Instance-Segmentation Level: Datasets and Framework. *IEEE Trans. Geosci. Remote Sens.* **2023**, *62*, 1–18. [CrossRef]
17. Hu, H.; Tan, Q.; Kang, R.; Wu, Y.; Liu, H.; Wang, B. Building extraction from oblique photogrammetry point clouds based on PointNet++ with attention mechanism. *Photogramm. Rec.* **2024**, *39*, 141–156. [CrossRef]
18. Chen, Q.; Gan, W.; Tao, P.; Zhang, P.; Huang, R.; Wang, L. End-to-end multiview fusion for building mapping from aerial images. *Inf. Fusion* **2024**, *111*, 102498. [CrossRef]
19. Wang, J.; Meng, L.; Li, W.; Yang, W.; Yu, L.; Xia, G.S. Learning to Extract Building Footprints from Off-Nadir Aerial Images. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *45*, 1294–1301. [CrossRef]
20. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, arXiv:1503.02531.
21. Zhang, Z.; Zhang, H.; Arik, S.O.; Lee, H.; Pfister, T. Distilling effective supervision from severe label noise. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 9294–9303.
22. Chen, H.; Shah, A.; Wang, J.; Tao, R.; Wang, Y.; Li, X.; Xie, X.; Sugiyama, M.; Singh, R.; Raj, B. Imprecise label learning: A unified framework for learning with various imprecise label configurations. *Adv. Neural Inf. Process. Syst.* **2024**, *37*, 59621–59654.
23. Himeur, Y.; Aburaed, N.; Elharrouss, O.; Varlamis, I.; Atalla, S.; Mansoor, W.; Al Ahmad, H. Applications of knowledge distillation in remote sensing: A survey. *Inf. Fusion* **2024**, *115*, 102742. [CrossRef]
24. Xu, G.; Deng, M.; Sun, G.; Guo, Y.; Chen, J. Improving Building Extraction by Using Knowledge Distillation to Reduce the Impact of Label Noise. *Remote Sens.* **2022**, *14*, 5645. [CrossRef]
25. Zhang, Y.; Xiang, T.; Hospedales, T.M.; Lu, H. Deep mutual learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4320–4328.

26. Neupane, B.; Aryal, J.; Rajabifard, A. Knowledge Transfer and Model Compression for Misaligned Building Labels. In Proceedings of the IGARSS 2023—2023 IEEE International Geoscience and Remote Sensing Symposium, Pasadena, CA, USA, 16–21 July 2023; pp. 3632–3635. [CrossRef]
27. Aryal, J.; Neupane, B. Multi-Scale Feature Map Aggregation and Supervised Domain Adaptation of Fully Convolutional Networks for Urban Building Footprint Extraction. *Remote Sens.* **2023**, *15*, 488. [CrossRef]
28. Alexander, C.; McKay, A.; Bhatt, K.; da Costa Lourenço, A.L.R.; Kaplan, B.; Krishnan, R.S.S.G. Pre-trained regional models for extracting buildings from high resolution satellite imagery to support public health initiatives. *Remote Sens. Appl. Soc. Environ.* **2024**, *36*, 101270.
29. Veit, A.; Alldrin, N.; Chechik, G.; Krasin, I.; Gupta, A.; Belongie, S. Learning from noisy large-scale datasets with minimal supervision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 839–847.
30. Mirikharaji, Z.; Yan, Y.; Hamarneh, G. Learning to segment skin lesions from noisy annotations. In Proceedings of the Domain Adaptation and Representation Transfer and Medical Image Learning with Less Labels and Imperfect Data: First MICCAI Workshop, DART 2019, and First International Workshop, MIL3ID 2019, Shenzhen, Held in Conjunction with MICCAI 2019, Shenzhen, China, 13–17 October 2019; Proceedings 1; Springer: Berlin/Heidelberg, Germany, 2019; pp. 207–215.
31. Ahn, S.; Kim, S.; Ko, J.; Yun, S.Y. Fine-tuning Pre-trained Models for Robustness under Noisy Labels. In Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24, Main Track, Jeju, Republic of Korea, 3–9 August 2024; Larson, K., Ed.; International Joint Conferences on Artificial Intelligence Organization: San Francisco, CA, USA, 2024; pp. 3643–3651. [CrossRef]
32. Mnih, V. Machine Learning for Aerial Image Labeling. Ph.D. Thesis, University of Toronto, Toronto, ON, Canada, 2013. Available online: https://www.cs.toronto.edu/~vmnih/docs/Mnih_Volodymyr_PhD_Thesis.pdf (accessed on 18 March 2025).
33. Tan, M.; Le, Q. Efficientnetv2: Smaller models and faster training. In Proceedings of the International conference on machine learning. PMLR, Online, 18–24 July 2021; pp. 10096–10106.
34. Mehta, S.; Rastegari, M. Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer. *arXiv* **2021**, arXiv:2110.02178.
35. Zhou, Z.; Siddiquee, M.M.R.; Tajbakhsh, N.; Liang, J. Unet++: Redesigning skip connections to exploit multiscale features in image segmentation. *IEEE Trans. Med. Imaging* **2019**, *39*, 1856–1867.
36. Huang, H.; Lin, L.; Tong, R.; Hu, H.; Zhang, Q.; Iwamoto, Y.; Han, X.; Chen, Y.W.; Wu, J. Unet 3+: A full-scale connected unet for medical image segmentation. In Proceedings of the ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; IEEE: New York, NY, USA, 2020; pp. 1055–1059.
37. Chaurasia, A.; Culurciello, E. Linknet: Exploiting encoder representations for efficient semantic segmentation. In Proceedings of the 2017 IEEE visual communications and image processing (VCIP), St. Petersburg, FL, USA, 10–13 December 2017; IEEE: New York, NY, USA, 2017; pp. 1–4.
38. Zhao, H.; Shi, J.; Qi, X.; Wang, X.; Jia, J. Pyramid scene parsing network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2881–2890.
39. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125.
40. Chen, L.C.; Zhu, Y.; Papandreou, G.; Schroff, F.; Adam, H. Encoder-decoder with atrous separable convolution for semantic image segmentation. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 801–818.
41. Fan, T.; Wang, G.; Li, Y.; Wang, H. Ma-net: A multi-scale attention network for liver and tumor segmentation. *IEEE Access* **2020**, *8*, 179656–179665.
42. Liu, J.; Li, R.; Sun, C. Co-correcting: Noise-tolerant medical image classification via mutual label correction. *IEEE Trans. Med. Imaging* **2021**, *40*, 3580–3592.
43. Liu, L.; Zhang, H.; Wang, Y. Contrastive mutual learning with pseudo-label smoothing for hyperspectral image classification. *IEEE Trans. Instrum. Meas.* **2024**, *73*, 1–14.
44. Iakubovskii, P. Segmentation Models Pytorch. GitHub Repository, 2019. Available online: https://github.com/qubvel/segmentation_models.pytorch (accessed on 18 March 2025).
45. Bakirman, T.; Komurcu, I.; Sertel, E. Comparative analysis of deep learning based building extraction methods with the new VHR Istanbul dataset. *Expert Syst. Appl.* **2022**, *202*, 117346.