Expert Systems With Applications 209 (2022) 118380

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Semantic segmentation of clouds in satellite images based on U-Net++ architecture and attention mechanism

Preetpal Kaur Buttar *, Manoj Kumar Sachan

Department of Computer Science and Engineering, Sant Longowal Institute of Engineering and Technology, Longowal, Punjab, India

ARTICLE INFO

Keywords:
U-Net++
ResNet
95-Cloud
Semantic segmentation
Cloud segmentation
Multispectral satellite data

ABSTRACT

The presence of clouds in satellite imagery may pose hindrances to the accurate and reliable analysis of the objects present on the land. Therefore, automatic cloud detection is a vital pre-processing step before lending the satellite images to any further analysis. This is a challenging task due to the varying thickness and densities of clouds. It is also very difficult to distinguish clouds from certain terrains such as snow and white sandy beaches. This paper proposes a deep learning based algorithm to solve the problem of cloud segmentation on the Landsat 8 multispectral dataset, 95-Cloud: SEUNet++. Specifically, the proposed model consists of a U-Net++ semantic segmentation model with a lightweight channel attention mechanism. We also experimented with using different encoder backbones in the U-Net++ encoder-decoder architecture such as ResNet (Residual Neural Networks) variants including ResNet-18, ResNet-34, ResNet-50, and ResNet-101, DenseNet-264, CSPNet (Cross Stage Partial Network), and EfficientNet-B8 and compared their performance. The experimental results show that the proposed architecture achieves an IoU (Intersection over Union) score of 91.8 %, improving the state-of-the-art on the task by 0.23 %. It also boosts the accuracy, precision, and recall values creating crisp cloud boundaries and detecting even thin layers of clouds. We also experimented using transfer learning and found that it has a positive impact on the cloud segmentation task. The proposed model also beats the original U-Net++ architecture in terms of various evaluation metrics such as the IoU score, accuracy, precision, and recall. The experimental results thereby demonstrate that our model is computationally efficient and achieves precise segmentation results.

1. Introduction

The earth observation satellites revolving around the Earth produce hundreds of terabytes of image data every day. Automated methods are needed to preprocess such abundant amounts of data for detecting certain kinds of anomalies such as the presence of clouds before they can be used for any further analysis. Cloud detection is one of the important preprocessing steps in satellite image processing. The presence of clouds in the satellite images poses obstacles to accurate analysis of remotely sensed satellite images as it hides the ground objects from the view of the satellite. Thus, precise detection and removal of clouds from satellite images is an important task. This is a complex task as clouds are present in a variety of thicknesses and densities. Sometimes, the clouds look very similar to the background objects on the land such as snow or sandy, white beaches. Moreover, to serve a useful purpose, the task requires demarcation of clouds at pixel-level rather than simply labeling an image as containing a cloud or not.

Existing cloud segmentation algorithms are very complex and computationally expensive. They also do not achieve a satisfactory level of performance, especially in the presence of snow and haze (Mohajerani & Saeedi, 2021).

This paper proposes a U-Net++ (Zhou, Rahman Siddiquee, Tajbakhsh, & Liang, 2018) based semantic segmentation model, SEUNet++, with a lightweight channel-based attention mechanism for segmenting the clouds from multispectral satellite images. U-Net++ is a variant of U-Net (Ronneberger, Fischer, & Brox, 2015), a U-shaped CNN (Convolutional Neural Network), which was originally designed for the medical image segmentation domain.

The last decade has witnessed enormous success in image processing tasks due to advancements in deep learning based CNNs (Krizhevsky, Sutskever, & Hinton, 2017). CNNs have now become a prominent approach and state-of-the-art to image classification (He, Zhang, Ren, & Sun, 2016). CNNs employ convolutional filters which move across the image from left to right and top to bottom to learn patterns in the

* Corresponding author.
E-mail address: preetpal@sliet.ac.in (P. Kaur Buttar).

https://doi.org/10.1016/j.eswa.2022.118380
Received 2 February 2022; Received in revised form 27 June 2022; Accepted 1 August 2022
Available online 5 August 2022

images. Learning is performed layer-wise through weight-sharing in an end-to-end manner. The initial layers learn fine features such as lines, curves, edges, etc., while the later layers learn more abstract features from the image.

As with image classification, CNN-based approaches have become dominant approaches to the task of semantic segmentation (Chen, Papandreou, Kokkinos, Murphy, & Yuille, 2015). Before CNN-based approaches to semantic segmentation, this task relied on spatial feature extraction and texture of the images (Shotton, Johnson, & Cipolla, 2008). The limitation of this approach was that it was unable to extract enough information for the segmentation of remote-sensing images and performed poorly on difficult segmentation tasks (Fraz, Javed, & Basit, 2008; Fukushima, 1980). Processing satellite images is a challenging task due to various factors such as uneven texture, uneven illumination across an image, cloud shadows, etc. (W. Zhao, Du, Wang, & Emery, 2017). Therefore, it requires the application of more robust and efficient CNN-based approaches to the task of semantic segmentation of satellite images.

There are various CNN-based approaches for the task of semantic segmentation which are based on encoder-decoder architecture, such as fully convolutional networks (FCNs) (Long, Shelhamer, & Darrell, 2015), SegNet (Badrinarayanan, Kendall, & Cipolla, 2017), Deconvolution Network (DeconvNet) (Noh, Hong, & Han, 2015), U-Net (Ronneberger et al., 2015), DeepLab (Chen et al., 2015; Chen, Papandreou, Kokkinos, Murphy, & Yuille, 2018), RefineNet (Lin, Milan, Shen, & Reid, 2017), Pyramid Scene Parsing Network (PSPNet) (Zhao, Shi, Qi, Wang, & Jia, 2017), Global Convolutional Network (GCN) (Peng, Zhang, Yu, Luo, & Sun, 2017), FastFCN (Wu, Zhang, Huang, Liang, & Yu, 2019), Kernel-Sharing Atrous Convolution (KSAC) (Huang et al., 2021), etc.

Among these, the U-Net architecture turned out to be a revolution in the field of image semantic segmentation. It won the International Symposium on Biomedical Imaging (ISBI) cell tracking challenge, in 2015 in various categories by a great edge (Bharath, 2021). All of the top entries in the 2017 "DSTL Satellite Imagery Feature Detection" challenge on Kaggle employed some variant of U-Net due to its ability to combine low-level feature maps with higher-level ones, enabling precise localization (Iglovikov, Mushinskiy, & Osin, 2017). U-Net and its variants have been applied in numerous tasks related to satellite imagery such as identifying landslide scars (Bragagnolo, Rezende, da Silva, & Grzybowski, 2021), building extraction (Li et al., 2019; Yi et al., 2019), forest cover mapping (Bragagnolo, da Silva, & Grzybowski, 2021), land cover mapping (Neves et al., 2020; Rakhlin, Davydow, & Nikolenko, 2018; Ulmas & Liiv, 2020), cloud detection (Guo, Cao, Liu, & Gao, 2020), natural disaster damage mapping (Bai, Mas, & Koshimura, 2018), etc.

The original U-Net architecture suffers from the problem of loss of localization precision. To counter this, several modifications to the basic U-Net architecture have been proposed in the literature. For example, (Zhou et al., 2018) proposed U-Net++ where the encoder and decoder sub-networks are connected through a series of nested, dense skip pathways. The authors improved upon the original U-Net by making these additions: redesigned nested and dense skip pathways having convolution layers, dense skip connections, and deep supervision, explained in more detail in Section 3.2. The redesigned skip pathways resulted in bridging the semantic gap between the feature maps of the encoder and the decoder paths as the fine-grained feature maps from the encoder sub-network are gradually enriched before blending them with the corresponding feature maps from the decoder sub-network. The dense skip connections were used in U-Net++ to improve the gradient flow which is inspired by DenseNet (Huang, Liu, Van Der Maaten, & Weinberger, 2017). The authors compared the performance of U-Net++ with the original U-Net architecture across several medical image segmentation tasks and found that U-Net++ improves the average IoU score with a gain of 3.9 points over U-Net.

Attention in neural networks is a mechanism which allows a model to focus on the most relevant features (Hu, Shen, Albanie, Sun, & Wu, 2020; Woo, Park, Lee, & Kweon, 2018). It was first proposed by (Bahdanau, Cho, & Bengio, 2014) for a natural language processing task of machine translation. This idea was later borrowed for improving the representational power of CNNs (Gregor, Danihelka, Graves, Rezende, & Wierstra, 2015; Jaderberg, Simonyan, Zisserman, & kavukcuoglu, 2015; Xu et al., 2015). In the proposed architecture, we incorporated an attention module after the convolution operations in a U-Net++ decoder block to compute the attention maps which allows to focus on the most important convolutional layer activations and ignoring the irrelevant ones.

CNN architectures such as AlexNet (Krizhevsky, Sutskever, & Hinton, 2012), VGGNet (Simonyan & Zisserman, 2015), ResNet (He et al., 2016), MobileNet (Howard et al., 2017), EfficientNet (Tan & Le, 2021), etc., pretrained on large image datasets are usually employed as encoder backbones in semantic segmentation networks. Our proposed model consists of using ResNet (He et al., 2016) (ResNet-18, ResNet-34, ResNet-50, and ResNet-101), DenseNet-264 (Huang et al., 2017), CSPNet (Wang et al., 2020), and EfficientNet-B8 (Tan & Le, 2019) as the encoder in the encoder-decoder structure of U-Net++ to perform the cloud segmentation task. We compared the performance of the proposed architecture employing these encoder backbones and found ResNet-50 giving the best performance. The proposed model architecture has been trained on Landsat 8 satellite imagery dataset, 95-Cloud. The experimental results show that our proposed architecture achieves an IoU score of 90.17 %, without using the attention modules in the decoder blocks. The application of attention modules in the U-Net++ decoder blocks further improved the results by 1.63 %, thus achieving an IoU score of 91.8 %, which is 0.23 % higher than that of the best performing model on this task and it also boosts the accuracy, precision, and recall values creating crisp cloud boundaries and detecting even thin layers of clouds.

We also experimented with using transfer learning on the cloud segmentation task. Transfer learning allows using a model already pretrained on a vast and diverse dataset on a new learning task without requiring to train the model from scratch. This helps to improve the model's performance, reduces the training time, and can be used to train models on small datasets efficiently. Our experiments found that transfer learning has a positive impact on the cloud segmentation task. The proposed model also beats the original U-Net++ architecture in terms of various evaluation metrics such as the IoU score, accuracy, precision, and recall. The outcomes in terms of the IoU score, accuracy, precision, and recall show that the proposed model attained reliable and robust results and can be efficiently used to segment the clouds in a precise manner.

Thus, the contributions of our work can be summarized as under:

- We compared the performance of different pretrained CNN models as encoder in the U-Net++ encoder-decoder architecture on the cloud segmentation task and found that ResNet-50 as encoder gives the best performance among others.
- We examined the effect of transfer learning on the cloud segmentation task and found that the models pretrained on 3-channel (red, green, and blue) images of everyday objects are able to perform well and improve the segmentation results on 4-channel (red, green, blue, and near infra-red (NIR)) images of clouds.
- We incorporated a lightweight channel attention mechanism in the decoder blocks of U-Net++ based on squeeze-and-excitation (SE) attention modules to enable the model to focus on most relevant features to extract. The addition of attention modules resulted in improving the performance by lifting the IoU score by 1.63 %.

The rest of this article has been organized as follows: section-2 discusses the related literature for cloud detection/segmentation methods in satellite images, section-3 presents the background of the work, section-4 describes the proposed architecture for cloud segmentation and its working, section-5 puts forward the experimental settings,

section-6 presents the experimental results and discussion, and section-7 concludes the research work presented in the article.

2. Related work

A lot of solutions have been proposed in the past to perform cloud detection, which can be mainly divided into threshold-based methods, handcrafted methods, and deep learning-based methods.

Threshold-based methods use a threshold value based on one or more spectral bands to decide to label a pixel as cloud or non-cloud. Among the threshold-based methods for cloud detection, the function of mask (FMask) (Qiu, Zhu, & He, 2019; Zhu, Wang, & Woodcock, 2015; Zhu & Woodcock, 2012) and automated cloud cover assessment (ACCA) (Irish, Barker, Goward, & Arvidson, 2006) are the most popular. The first version of FMask (Zhu & Woodcock, 2012) classified each pixel of an image into land, water, cloud, shadow, or snow using seven bands of Landsat satellite images. FMask v3 (Zhu et al., 2015) distinguished between cirrus clouds and low altitude clouds utilizing the cirrus band. The last version of FMask (Fmask v4) (Qiu et al., 2019) used digital elevation maps (DEMs), their derivatives, and global surface water occurrences (GSWOs) in addition to other usual bands for improving the performance on the water, high altitude regions, and Sentinel-2 images.

(Mateo-García, Gómez-Chova, Amorós-López, Muñoz-Marí, & Camps-Valls, 2018) used Landsat-8 cloud-free images to detect potential clouds. The final cloud masks were generated using clustering and some threshold-based postprocessing. (Zi, Xie, & Jiang, 2018) segmented the image superpixels and classified them as cloud, potential cloud, or non-cloud by combining a threshold-based method with a classical machine learning approach.

Handcrafted methods such as (Chen et al., 2016; Zhang, Guindon, & Li, 2014) exploit the relationship between red and blue spectral bands to detect haze and thick clouds. (Xu, Wong, & Clausi, 2017) employed multiple spectral, temporal, and spatial features for cloud detection using a Bayesian probabilistic model.

Traditional approaches to semantic segmentation fail to achieve satisfactory results when images contain complex scenarios with fragmented clouds mixed up with land objects.

Deep learning being the state-of-the-art for many computer vision-related tasks is widely being used by the research community for satellite image processing. (Xie, Shi, Shi, Yin, & Zhao, 2017) used a CNN for the classification of image patches into one of the classes: thin cloud, thick cloud, or clear. (Yang et al., 2019) employed an FCN to detect clouds in ZY-3 thumbnail satellite images. To reduce post-processing, they incorporated a built-in boundary refinement approach into their CSnetV1 model. In the second version of their network (CDnetV2) (Guo et al., 2021), they incorporated two new feature fusion and information guidance modules to extract cloud features more accurately. (Chai, Newsam, Zhang, Qiu, & Huang, 2019) proposed a modified version of SegNet and showed promising results on Landsat cloud and shadow detection datasets. (Jeppesen, Jacobsen, Inceoglu, & Skjødeberg, 2019) proposed RS-Net which was inspired by U-Net for cloud detection in Landsat 8 images. (Hughes & Kennedy, 2019) proposed an FCN with a pretrained VGG-16 backbone to distinguish between clouds, shadow, water, snow, and clear regions in Landsat 8 images. In (Jiao, Huo, Hu, & Tang, 2020), the authors proposed RefUNetv1 based on U-Net for segmenting coarse clouds and cloud shadows in Landsat 8 images whose boundaries were thereafter refined using a dense conditional random field (CRF) as a post-processing step. CRFs are one of the statistical modeling methods which consider the relationships among neighboring pixels before assigning a semantic label. The second version, RefUNetv2 (Jiao, Huo, Hu, & Tang, 2020), used a simultaneous joint pipeline with U-Net for identifying coarse masks and guided Gaussian filter-based CRF to refine the boundaries in an end-to-end manner. (Guo et al., 2020) used attention-based U-Net model called Cloud-AttU for cloud detection on 38-Cloud Landsat 8 cloud dataset. (Francis, Sidiropoulos, & Muller, 2019) proposed CloudFCN, a model based on U-Net using inception modules between the convolution blocks in the contracting and expanding paths. They also estimated cloud coverage in images with the help of a regression-friendly loss function and showed that their method is robust to the presence of white noise. CloudNet+ (Mohajerani & Saeedi, 2021) is a modification of Cloud-Net (Mohajerani & Saeedi, 2019) which uses novel filtered Jaccard loss and a sunlight direction-aware data augmentation technique.

The accuracy of deep learning models can further be increased using image augmentation techniques. In the remote sensing field, various generative adversarial network (GAN) based data augmentation techniques (Howe, Pula, & Reite, 2019; Ma, Tang, & Zhao, 2018; Zheng, Wei, Sun, Anas, & Li, 2019) have been proposed. For cloud segmentation, (Mohajerani et al., 2019) proposed CloudMaskGAN for transformations between snowy and non-snowy scenes on Landsat 8 images. The major challenge in using deep learning-based methods for cloud detection is the lack of accurate ground truth segmentation masks for training the models. As deep learning-based models rely on very large datasets, creating segmentation masks for a large number of training samples is a dreary and laborious task (Francis et al., 2019). Although the existing solutions have yielded promising results, generating accurate segmentation masks for images with bright/cold non-cloud regions along with the presence of clouds is still a major challenge (Ji, Dai, Lu, & Zhang, 2021; Zhu et al., 2015).

Apart from cloud semantic segmentation, deep learning has its applications to solve many real-life problems. For example, (Banan, Nasiri, & Taheri-Garavand, 2020) employed a CNN for automatic identification of carp species with 100 % accuracy. Similarly, (Chen et al., 2022) used a long short-term memory model (LSTM) for forecasting monthly rainfall. The authors found this deep learning based method better than a random forest model on various performance metrics which can be used globally for rainfall forecasting without any regard to climatic conditions. (Afan et al., 2021) used an ensemble of deep learning techniques to model the fluctuations of groundwater level for five wells in Malaysia and achieved promising results. Besides this, deep learning is being increasingly used in other domains such as managing wind and solar energy resources (Shamshirband, Rabczuk, & Chau, 2019). Deep learning can also be used in combination with other soft computing techniques, for example, (Fan, Xu, Wu, Zheng, & Tao, 2020) proposed a spatiotemporal model based on the Karhunen-Loève (KL) decomposition, the multilayer perceptron (MLP), and the long short-term memory (LSTM) network for estimating temperature distributions with a three-step procedure.

Our proposed model consisting of a pretrained CNN encoder backbone and a U-Net++ decoder with attention module makes the most of transfer learning by using the pretrained weights instead of training the model from scratch. This helps saving the training time as the model converges within lesser number of epochs. For example, the Cloud-Net (Mohajerani & Saeedi, 2019) model took 600 epochs to train on this task while our proposed model converged in 100 epochs, which is 83 % lesser. Moreover, we also exploited the use of attention mechanism in U-Net++ architecture by inserting SE channel attention layer after the convolutional layers in a decoder block, enabling the model to better attend to the features that are helpful for the cloud segmentation task. This helped the proposed model to achieve better segmentation results. As compared to Attention U-Net++ (Li et al., 2020) where every skip connected includes a heavy attention block, our proposed architecture utilizes a very lightweight attention that does not pose any extra training overhead.

3. Background

In this work, we employed a CNN for semantic segmentation of clouds from satellite images based on U-Net++ architecture that takes an image as input and outputs a label for each pixel. U-Net++ is a variant of the original U-Net (Ronneberger et al., 2015) (Fig. 1), which is one of the popular encoder-decoder based architectures for image

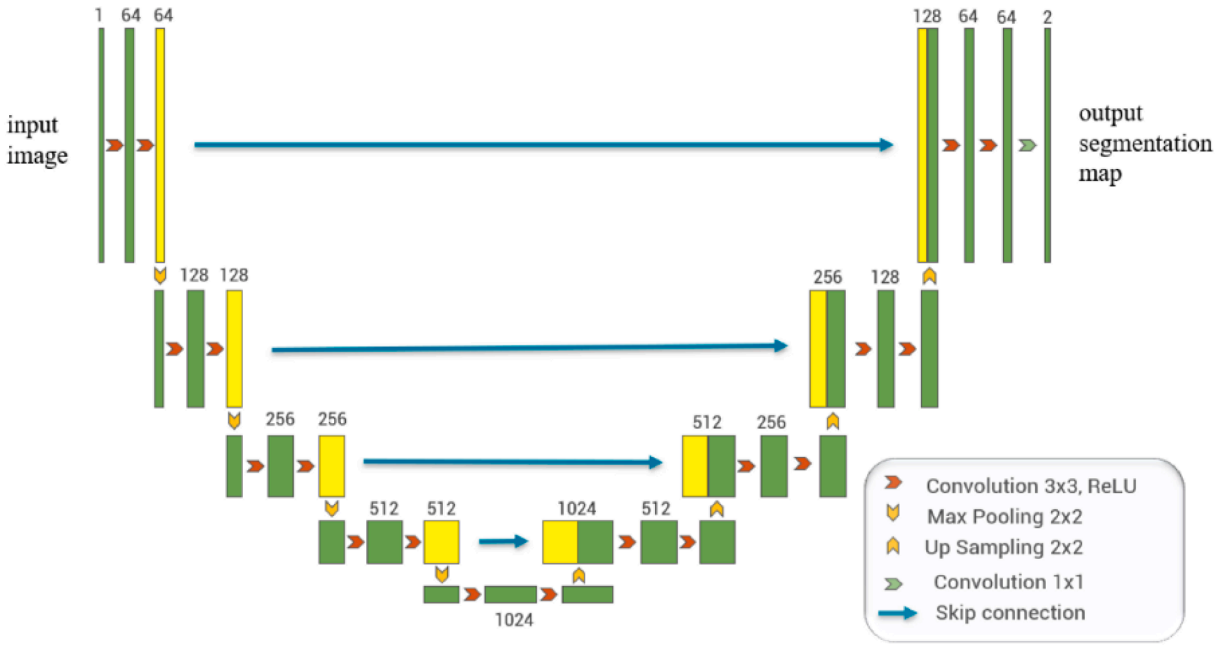

Fig. 1. The encoder-decoder based standard U-Net architecture (Ronneberger et al., 2015).

segmentation. In this section, we present a brief explanation of U-Net and U-Net++ architectures upon which the underlying work is based.

3.1. *The U-Net: Architecture*

Compared with other CNNs, U-Net requires a smaller training dataset and provides higher segmentation accuracy (Hou, Liu, Zhang, & Li, 2021). It was originally developed for the medical domain, which yields more accurate segmentation masks with much less training data. This encoder-decoder architecture captures higher-dimensional image representations to make a dense prediction by processing information layer by layer. The architecture can localize and distinguish borders by classifying every pixel so that the input and output share the same size. Also, due to the absence of any dense layers and the whole architecture consisting of only convolutional layers, U-Net can accept an image of any size.

There are two symmetrical paths in the U-Net architecture, i.e., a contractive path (encoder) and an expansive path (decoder), each consisting of the same number of layers. The encoder is like any standard CNN and is used to capture the context in the image. It extracts a meaningful feature map from an input image, doubles the number of channels at every step, and halves the spatial dimension. By contrast, the decoder is used to upsample the feature maps, where at every step it doubles the spatial dimension and halves the number of channels. It is used to enable precise localization using transposed 2D convolutions along with regular convolutions, thereby increasing the resolution of the output.

The encoder structure follows the traditional stack of convolutional and maxpooling layers to increase the receptive field as it goes through the layers. It gradually downsamples the spatial resolution at each layer by employing a sequence of two 3×3 filters followed by a 2×2 max-pooling layer. This process is done four times, with each downsampling step doubling the number of feature channels, thereby, allowing the network to learn the "WHAT" information contained in the image. However, in this process, the "WHERE" information is lost; here comes the function of the symmetric decoder part of the architecture. The decoder structure utilizes transposed convolution layers for upsampling so that the end dimensions are close to that of the input images. It has a similar structure to that of an encoder except that the maxpooling layers are replaced by transposed convolutions. The corresponding layers convolution and transposed convolution layers in the encoder and decoder respectively are connected using skip connections.

The advantage of U-Net lies in its addition of symmetric skip connections between the encoder and the decoder layers which enable the concatenation of the corresponding feature maps of the encoder and the decoder that helps improve the accuracy of pixel-level localization, thereby supporting segmentation and providing precise detailed location information. After every concatenation, two regular convolutions are applied, allowing the model to learn how to generate a more accurate output. Thus, we get a symmetric U-shaped architecture, hence, the name U-Net.

3.2. *U-Net++*

U-Net++ proposed in (Zhou et al., 2018) improved upon the original U-Net by making these additions: redesigned nested and dense skip pathways having convolution layers (shown in green) and dense skip connections (shown in blue), and deep supervision (shown in red) as shown in Fig. 2.

The authors argue that the redesigned skip pathways resulted in bridging the semantic gap between the feature maps of the encoder and the decoder paths as the fine-grained feature maps from the encoder sub-network are gradually enriched before blending them with the corresponding feature maps from the decoder sub-network. The dense skip connections were used in U-Net++ to improve the gradient flow which is inspired by DenseNet (Huang et al., 2017).

The enrichment of the feature maps from the encoder sub-network occurs in a dense convolution block that consists of several convolution layers. For example, the skip pathway between the nodes $X^{1,0}$ and $X^{1,3}$ consists of a dense convolution block with 2 convolution layers. The input to a convolution layer is the output of the previous convolution layer of the same dense block concatenated with the corresponding up-sampled output of the lower dense block.

More formally, the authors formulated the skip pathway as follows: let $x^{i,j}$ denote the output of the node $X^{i,j}$. Here, i indexes the encoder contracting path, and j indexes the convolution layer in the dense block along the skip pathway. Then, $x^{i,j}$ is computed as:

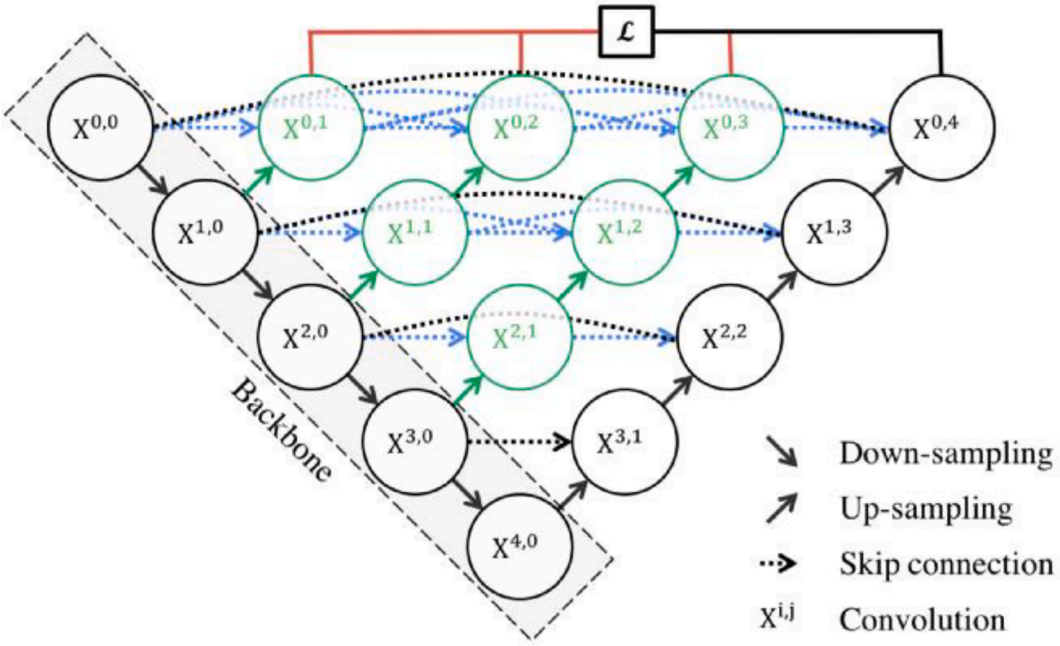

Fig. 2. The U-Net++ architecture (Zhou et al., 2018).

$$x^{i,j} = \begin{cases} H\left(x^{i-1},j\right), & j=0 \\ H\left(\left[\left[x^{i,k}\right]_{k=0}^{j-1}\right), U(x^{i+1,j-1})\right]\right), & j>0 \end{cases} \quad (1)$$

where $H(.)$ means a convolution operation followed by an activation function, $U(.)$ denotes up-sampling, and [] is the concatenation operation. At $j = 0$, the nodes get input only from the previous encoder layer. For all levels with $j > 1$, the nodes receive inputs from the previous convolution layers in the same dense convolution block coupled with the up-sampled feature map of the lower dense block. This results in an accumulation of all the previous feature maps at the current node thus making a dense convolution block along a skip pathway. The dense skip connections help gather all the previous feature maps along a skip pathway made of dense convolutional blocks.

Multiple full resolution feature maps are generated at different semantic levels, $\{x^{0,j}, j \in \{1,2,3,4\}\}$ and then the final segmentation map is generated through deep supervision. Deep supervision allows to adjust between the inference time and performance of the model by operating in two modes: the accurate mode, wherein the feature maps of all the segmentation branches are averaged, and the fast mode, in which one feature map is selected from the feature maps at the segmentation branches.

4. The proposed architecture

The proposed architecture, SEUNet++ consists of a U-Net++ based network with a pretrained CNN model as the encoder backbone and the decoder blocks comprising of SE attention modules.

4.1. SEUNet++ encoder

Let the input to the proposed architecture be an image of size $k \times m \times n$. This input image is first passed to the encoder sub-network, which consists of 5 encoder layers. We experimented with using 7 different pretrained models as encoders including 4 ResNet (He et al., 2016) variants, namely, ResNet-18, ResNet-34, ResNet-50, and ResNet-101, DenseNet-264 (Huang et al., 2017), CSPNet (Wang et al., 2020), and EfficientNet-B8 (Tan & Le, 2019). The feature maps at each encoder layer are extracted and the outputs of these 5 nodes, termed as, x_{00}, x_{10}, x_{20}, x_{30}, and x_{40} are of sizes $a \times \frac{m}{2} \times \frac{n}{2}$, $b \times \frac{m}{4} \times \frac{n}{4}$, $c \times \frac{m}{8} \times \frac{n}{8}$, $d \times \frac{m}{16} \times \frac{n}{16}$, and $e \times \frac{m}{32} \times \frac{n}{32}$, respectively. Here, a, b, c, d, and e are the number of filters in the feature maps produced at the respective layers. For example, in DenseNet-264, the values of a, b, c, d, and e are 96, 384, 768, 3456, and 4032, respectively, while, for ResNet-50 and ResNet-101, these values are 64, 256, 512, 1024, and 2048, respectively. For CSPResNet-50, the values of a, b, c, d, and e are 64, 128, 256, 512, and 1024, respectively, while for EfficientNet-B8, they are 32, 56, 88, 248, and 704, respectively.

4.1.1. ResNet variants as encoders

Building sufficiently deeper neural networks result in problems such as vanishing gradients, the curse of dimensionality, and the degradation problem where the accuracy improvement comes to a halt at a certain point and eventually starts degrading. The residual networks in ResNet are made up of residual blocks (Fig. 3) that consist of skip connections which enable the layers to learn the residual between the input and the output, instead of trying to learn the true input. ResNet consists of a series of residual blocks which are made up of layers such as batch normalization, ReLU activation, convolution layers stacked on top of each other. In ResNets, we either train the layers in residual blocks or skip their training using skip connections. So, different parts of networks will be trained at different rates for different training data points based on how the error flows backward in the network. This can be thought of

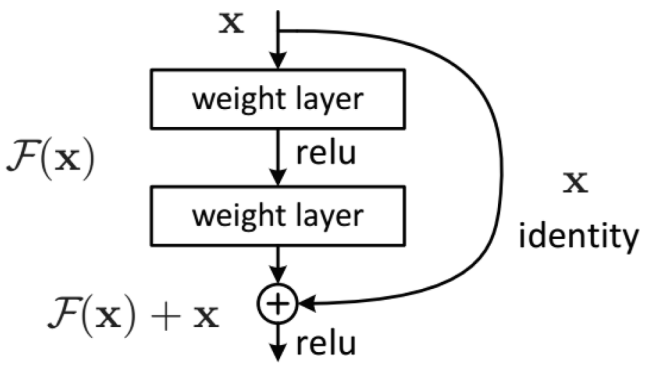

Fig. 3. A single residual block in ResNet architecture (He et al., 2016).

as training an ensemble of different models on the dataset and getting the best possible accuracy. Due to skip connections, larger gradients get passed to the initial layers which facilitate faster learning similar to that of the last layers. This allows for the training of deeper networks.

4.1.2. DenseNet-264 as encoder

DenseNet (Huang et al., 2017) architecture is based on densely connected convolutional layers where each layer gets as input the feature maps from all the previous layers (Fig. 4). It then concatenates the input feature maps with its own feature map channel-wise and forwards the concatenated feature map of all the succeeding layers. Each such collection of layers densely connected to each other form a Dense Block and the DenseNet architecture consists of several such Dense Blocks. As all the layers are directly connected with each other, error signal propagation from the final layer to initial layers is direct and strong. There are various DenseNet models available in the Torch library such as DenseNet-121, DenseNet-169, DenseNet-201, DenseNet-264, etc. We selected DenseNet-264 for our experiments as it has the best top-1 accuracy as well as top-5 accuracy on the ImageNet dataset among other variants.

4.1.3. CSPNet as encoder

The densely connected structure of DenseNet results in duplication of gradient information (Liu & Zeng, 2018; Wang et al., 2020). CSPNet (Wang et al., 2020) is designed to solve this problem of DenseNet by clipping the gradient flow. CSPNet applied to DenseNet, CSPDenseNet keeps the advantages benefits of DenseNet's reuse qualities while also preventing an unnecessarily excessive amount of duplicate gradient information. CSPNet can also be applied to other CNN architectures such as, ResNet and ResNeXt. We applied CSPNet to our best performing encoder backbone, i.e., ResNet.

4.1.4. EfficientNet as encoder backbone

EfficientNet (Tan & Le, 2019) is designed of efficiently scale the CNN's depth, width, and resolution evenly on the basis of a compound coefficient. The intuition behind the compound scaling is that the number of layers and the number of channels needed by a CNN are proportional to the images size, such that a bigger images requires more layers and channels and vice-versa. The EfficientNet series consists of 9 models from EfficientNet-B0 to EfficientNet-B8. For our experiments, we selected EfficientNet-B8 which is having the highest accuracy among others.

4.2. SEUNet++ decoder

The detailed architecture of the proposed model is shown in Fig. 5. The five encoder layers are shown on the left size inside a grey rectangle. The rest of the figure on the right side represents the decoder.

Each decoder node consists of three layers including two convolutional layers (ConvLayer) followed by an attention layer, except the nodes at topmost level which apply a 1×1 2D convolution to generate the final segmented map consisting of a single channel representing the binary mask for cloud/no-cloud classes. Here, each ConvLayer performs a sequence of convolution, ReLU activation, and batch normalization operations on the input.

SE attention module: Our proposed architecture utilizes SE attention module, which is a lightweight channel attention module proposed by (Hu et al., 2020). Here, 'S' refers to the squeeze operation that performs a channel-wise global average pooling over the entire feature map and 'E' refers to the excitation operation which performs the activation using a couple of fully connected layers and an activation function.

Let $X \in \mathbb{R}^{H \times W \times C}$ be a feature map generated after a convolution operation, where H, W, and C are the height, width, and number of channels in the feature map. Then, the squeeze operation performs a channel-wise global average pooling over the spatial dimensions of the feature map to produce an output, $S \in \mathbb{R}^{1 \times 1 \times C}$. This is then passed through the excitation operation which generates channel-wise modulation weights, $E \in \mathbb{R}^{1 \times 1 \times C}$, by employing two fully connected layers and a ReLU (Rectified Linear Unit) activation. E is then applied to the feature map X to generate attention-based feature maps to be fed to subsequent layers of the neural network. The detailed architecture of an SE attention module is shown in Fig. 6.

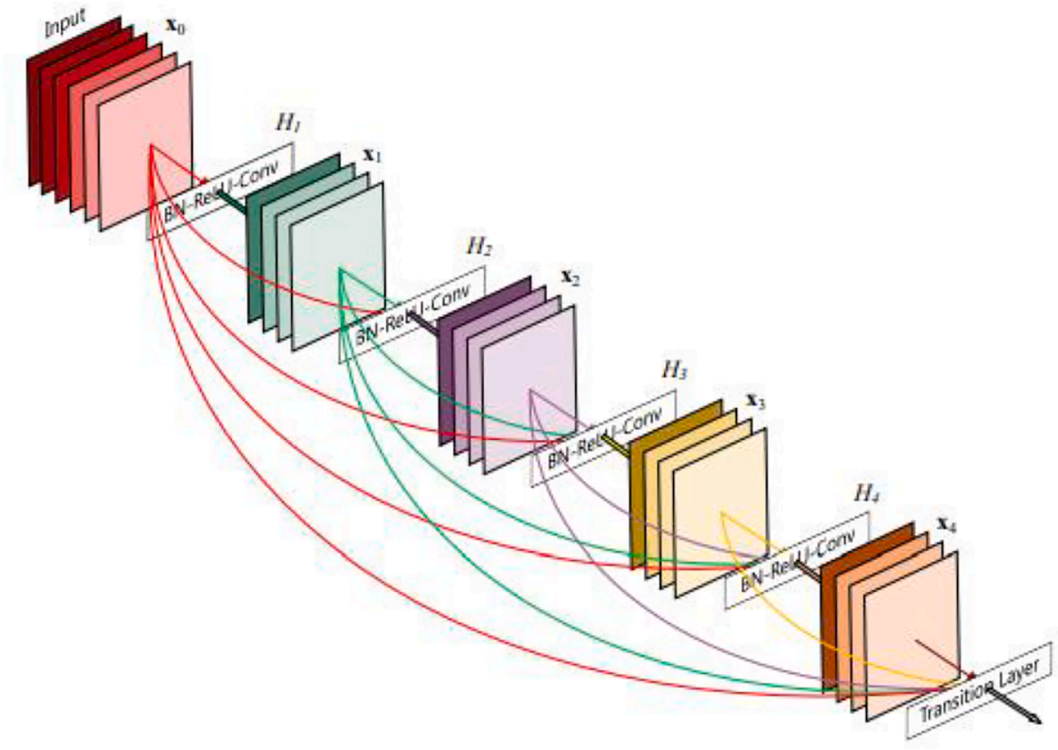

Fig. 4. A Dense Block in DenseNet architecture (Huang et al., 2017).

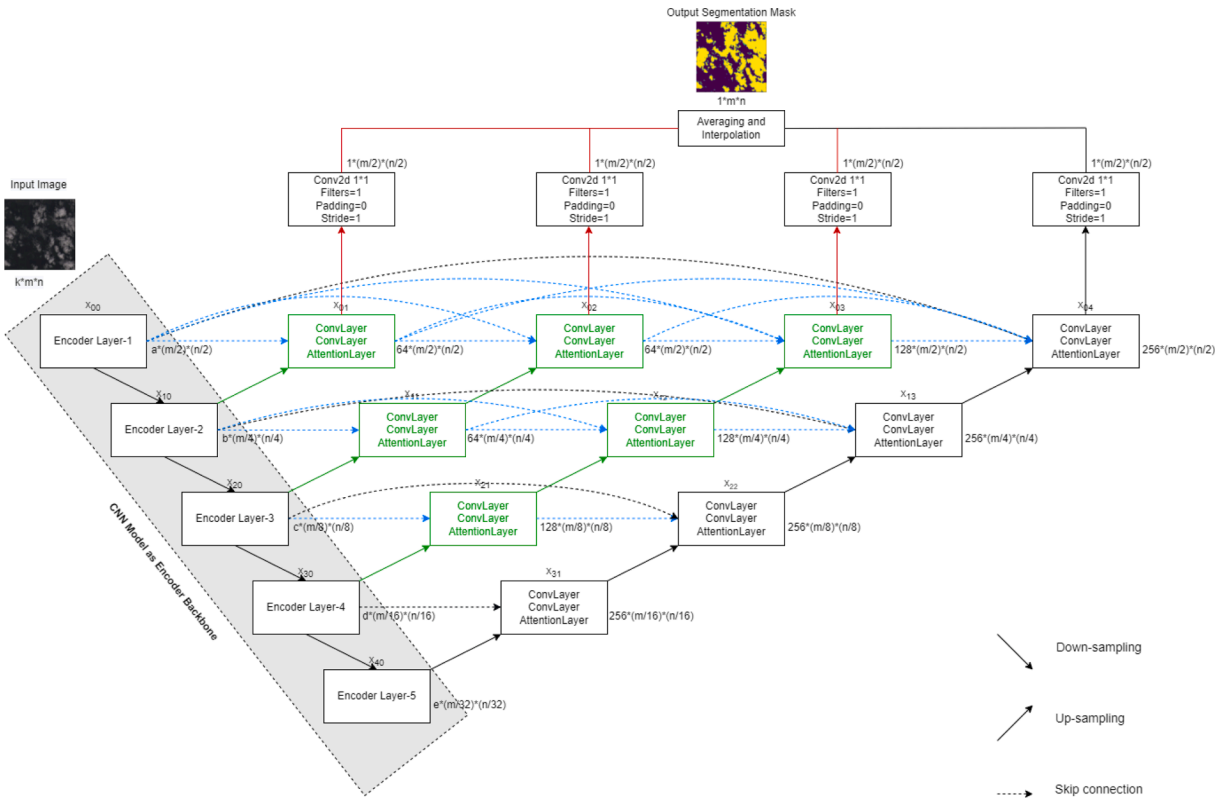

Fig. 5. The detailed architecture of the proposed SEUNet++.

In the proposed architecture, the attention layer at each decoder node applies SE attention on the input received from the second ConvLayer.

The decoder nodes can be viewed as belonging to 4 different levels, $j = 1,2,3,4$. The first level, $j = 1$ produces four outputs x_{01}, x_{11}, x_{21}, and x_{31}, as shown in Fig. 6. Similarly, the second level $j = 2$ consists of nodes x_{02}, x_{12}, and x_{22} and the third level $j = 3$ consists of nodes x_{03} and x_{13}, and the fourth level $j = 4$ consists of a single node x_{04}.

In Fig. 6, the green decoder nodes represent the dense skip pathways along the horizontal axis. For example, from the encoder node x_{00} to the decoder node x_{04} through x_{01}, x_{02}, and x_{03} is a dense skip pathway. The nodes along a skip pathway form a dense convolution block. For example, x_{01}, x_{02}, and x_{03} form one dense convolution block.

There are skip connections (shown in black) connecting each encoder node with the corresponding last decoder node, horizontally. For example, the encoder node x_{00} is connected to the decoder node x_{04} through a skip connection. The dense skip connections are shown in blue dotted arrows connect either an encoder node to a green decoder node, i. e., a node in a dense convolution block, or the nodes within a dense convolution block, or the node from a dense convolution block to the last decoder node on that horizontal path.

Each decoder node receives as input the feature maps from the corresponding encoder layer and from all the previous decoder nodes in that node's dense convolution block along the horizontal axis through dense skip connections as well as the upsampled output from the previous decoder node at a lower-level dense convolution block.

For example, at node x_{01}, the output of skip connected x_{00} and the upsampled output of previous x_{10} are concatenated and then passed through the two convolutional layers and an attention layer to produce a $64 \times \frac{m}{2} \times \frac{n}{2}$ output. Similarly, at node x_{11}, the output of skip connected x_{10} and the upsampled output of previous x_{20} are concatenated and processed to produce a $64 \times \frac{m}{4} \times \frac{n}{4}$ output. The same pattern is repeated for x_{21} and x_{31} nodes to produce outputs of sizes $128 \times \frac{m}{8} \times \frac{n}{8}$ and $256 \times \frac{m}{16} \times \frac{n}{16}$, respectively.

The next level $j = 2$ produces three outputs x_{02}, x_{12}, and x_{22} by concatenating the output of the previous two convolution layers along the same skip pathway through skip connections with the upsampled output of the previous lower-level dense convolution block. For example, the inputs to x_{02} are the outputs of x_{00} and x_{01} through skip connections and the upsampled output of x_{11}. The sizes of the feature maps produced for x_{02}, x_{12}, and x_{22} are $64 \times \frac{m}{2} \times \frac{n}{2}$, $128 \times \frac{m}{4} \times \frac{n}{4}$, and $256 \times \frac{m}{8} \times \frac{n}{8}$, respectively.

The level $j = 3$ produces two outputs x_{03} and x_{13} by concatenating the output of the previous three convolution layers in the same dense convolution block with the upsampled output of the previous layer. For example, the inputs to x_{03} are the outputs of x_{00}, x_{01}, and x_{02} through skip connections and the upsampled output of x_{12}. The feature maps produced for x_{03} and x_{13} are of sizes $128 \times \frac{m}{2} \times \frac{n}{2}$ and $256 \times \frac{m}{4} \times \frac{n}{4}$, respectively.

The final level $j = 4$ produces an output feature map of size $256 \times \frac{m}{2} \times \frac{n}{2}$ by concatenating the feature maps of the convolution layers x_{00}, x_{01}, x_{02}, and x_{03} through skip connections with the upsampled output of lower dense block from node x_{13}.

The feature maps of the nodes x_{01}, x_{02}, x_{03}, and x_{04} are then again passed through a 2-D convolution to reduce the respective number of filters in each feature map to 1. Thus, in the end, the size of the final feature maps at multiple semantic levels is $1 \times \frac{m}{2} \times \frac{n}{2}$. All these feature maps are then averaged to produce a final feature map. The final feature map is then interpolated to produce a $1 \times m \times n$ size full-resolution segmentation map corresponding to a $k \times m \times n$ input image.

Table 1 provides a detailed layer-wise description of the proposed model. Here, *Decoder Block* represents the operations applied on the input feature map in a decoder node including the application of two convolutional layers and an attention layer, *Concat[]* refers to the concatenation of the feature maps mentioned inside the square brackets, *upsample* is the upsampling operation on a given feature map to increase its height and width.

The improvements to the architecture as compared to the original

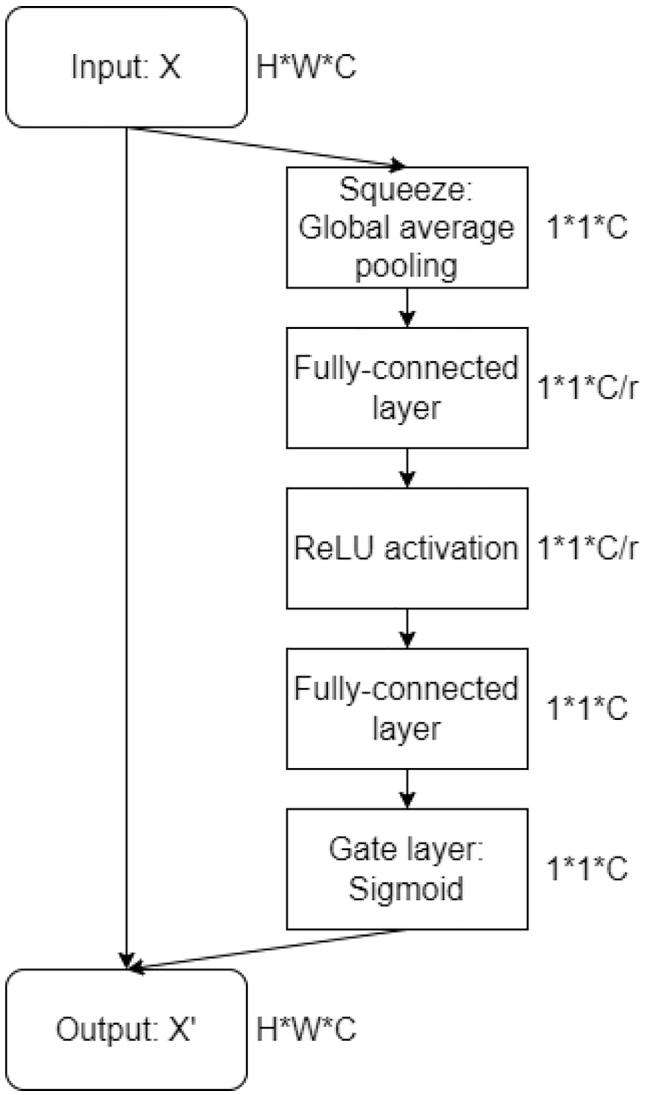

Fig. 6. An SE attention module.

architecture are:

- The use of lightweight SE channel attention helps the neural network to focus on relevant features in the feature map and ignore the irrelevant ones, which helps to improve the segmentation performance of the proposed architecture. Futhermore, being a lightweight attention module does not add any overwhelming computational cost for training the network, unlike Attention U-Net++ (C. Li et al., 2020) which used heavy-weight attention.
- Kernels of size 3×3 with uniform padding of 1 are used in all the convolution layers of the dense convolution blocks that keep the height and the width of the feature map uniform across each skip pathway.

Typical values for kernel size include 1×1, 3×3, 5×5, and 7×7. Choosing a kernel size greater than 3×3 enables learning larger spatial filters and helps reduce the volume size, but it does not capture the fine details such as straight edges and sharp tips (Öztürk, Özkaya, Akdemir, & Seyfi, 2018). Using a 1×1 filter, on the other hand, implies capturing very low-level features on a per-pixel basis without capturing any neighboring context of the pixels at all. As this research work is concerned with the task of cloud segmentation, where images exhibit very complex textures with different shapes, densities, and thicknesses of clouds, using a 3×3 kernel was the optimal choice that enabled capturing fine details in the images together with overall context.

The shape of the output feature map of a convolutional layer depends on the input shape and the kernel size. If the input shape is $m \times n$, where m and n are the height and the width of the input feature map, respectively, and the kernel shape is $k_h \times k_w$, where k_h and k_w are the kernel height and width, respectively, then the output shape will be: $(m - k_h + 1) \times (n - k_w + 1)$. As the kernels generally have height and width greater than 1, applying successive convolutions will result in generating output significantly smaller than the input, losing useful boundary information of the original image (Dwarampudi & Reddy, 2019). Padding is a solution for tackling this issue which adds extra filler pixels at the boundary of the image and thus increases its effective size. As the semantic segmentation task requires that the spatial dimensions of the input image and that of the output feature map should be the same, we used uniform padding of 1 to preserve the spatial dimensions.

As an example, in our proposed architecture, along the skip pathway from x_{00} to x_{04}, the size of the feature maps is $\frac{m}{2} \times \frac{n}{2}$. This permits for the boundary information to be preserved and adds more convolutions.

- As the size of the feature map produced by the convolution layers inside a dense convolution block is uniform, there is no need to crop the feature map before feeding it to any succeeding convolution layer in a dense convolution block through skip connections.

These improvements make the proposed architecture perform efficiently and accurately on the semantic segmentation task.

5. Experimental setup

5.1. The dataset

There are several publicly available datasets for training deep neural networks for the task of cloud semantic segmentation, such as 38-Cloud (Mohajerani & Saeedi, 2019), 95-Cloud (Mohajerani & Saeedi, 2021), Biome 8 (Hughes & Hayes, 2014), SPARCS dataset (Hughes & Kennedy, 2019), etc. The 38-Cloud dataset is a subset of the 95-Cloud dataset containing 18 scenes obtained from Landsat 8 satellite as the training set, which are divided into 8400 non-overlapping 384×384 size patches. The 95-Cloud dataset consists of 57 scenes extracted from Landsat 8 divided into patches of the same size as that of the 38-Cloud dataset. The Biome dataset consists of 96 scenes from Landsat 8 with manually generated ground truth segmentation maps, where each pixel is classified into one of the five classes: cloud, thin cloud, clear, cloud shadow, and empty. The SPARCS dataset consists of 80 patches extracted from Landsat 8 scenes with each pixel classified into one of the categories as cloud, shadow, snow/ice, water, land, or flooded.

As the research work is concerned with binary segmentation of clouds by classifying each pixel in a satellite image as 'cloud' or 'no cloud', we discarded working with the Biome dataset and the SPACS dataset, which are concerned with multi-class semantic segmentation. Moreover, the SPARCS dataset is very small for a deep neural network to be effectively trained. More the number of training samples, the better the deep neural network trains. As the 38-Cloud dataset is a subset of the 95-Cloud dataset, we decided to work with 95-Cloud dataset with a greater number of training instances to achieve better results.

This is a benchmark dataset for cloud segmentation, publicly available at the Kaggle platform (https://www.kaggle.com). The 95-Cloud dataset consists of 57 scenes extracted from the Landsat-8 earth observation satellite mission. These scenes are divided into 384×384 size patches. This results in a total of 26,301 image patches. The patches consist of four bands – red, green, blue, and NIR. Additionally, there is a ground truth patch to mark the clouds. The dataset keeps each band in a separate folder. Each band is a matrix with values between 0 and 255. The ground truth images are the binarized masks with pixel values 1 for cloud or 0 for non-cloud.

Table 1
A detailed layer-wise description of the proposed model SEUNet++.

	Level	Node	Operation	Kernel size	Filters	Padding	Stride	Output size
Input	$4 \times 192 \times 192$							
Encoder		x_{00}	Encoder Layer-1					$a \times 96 \times 96$
		x_{10}	Encoder Layer-2					$b \times 48 \times 48$
		x_{20}	Encoder Layer-3					$c \times 24 \times 24$
		x_{30}	Encoder Layer-4					$d \times 12 \times 12$
		x_{40}	Encoder Layer-5					$e \times 6 \times 6$
Decoder	$j = 1$	x_{01}	Concat$[x_{00}, upsample(x_{10})]$					$[a + b] \times 96 \times 96$
			Decoder Block	3×3	64	1	1	$64 \times 96 \times 96$
		x_{11}	Concat$[x_{10}, upsample(x_{20})]$					$[b + c] \times 48 \times 48$
			Decoder Block	3×3	64	1	1	$64 \times 48 \times 48$
		x_{21}	Concat$[x_{20}, upsample(x_{30})]$					$[c + d] \times 24 \times 24$
			Decoder Block	3×3	128	1	1	$128 \times 24 \times 24$
		x_{31}	Concat$[x_{30}, upsample(x_{40})]$					$[d + e] \times 12 \times 12$
			Decoder Block	3×3	256	1	1	$256 \times 12 \times 12$
	$j = 2$	x_{02}	Concat$[x_{00}, x_{01}, upsample(x_{11})]$					$[a + x_{01} + x_{11}] \times 96 \times 96$
			Decoder Block	3×3	64	1	1	$64 \times 96 \times 96$
		x_{12}	Concat$[x_{10}, x_{11}, upsample(x_{21})]$					$[b + x_{11} + x_{21}] \times 48 \times 48$
			Decoder Block	3×3	128	1	1	$128 \times 48 \times 48$
		x_{22}	Concat$[x_{20}, x_{21}, upsample(x_{31})]$					$[c + x_{21} + x_{31}] \times 24 \times 24$
			Decoder Block	3×3	256	1	1	$256 \times 24 \times 24$
	$j = 3$	x_{03}	Concat$[x_{00}, x_{01}, x_{02}, upsample(x_{12})]$					$[a + x_{01} + x_{02} + x_{12}] \times 96 \times 96$
			Decoder Block	3×3	128	1	1	$128 \times 96 \times 96$
		x_{13}	Concat$[x_{10}, x_{11}, x_{12}, upsample(x_{22})]$					$[b + x_{11} + x_{12} + x_{22}] \times 48 \times 48$
			Decoder Block	3×3	256	1	1	$256 \times 48 \times 48$
	$j = 4$	x_{04}	Concat$[x_{00}, x_{01}, x_{02}, x_{03}, upsample(x_{13})]$					$[a + x_{01} + x_{02} + x_{03} + x_{13}] \times 96 \times 96$
			Decoder Block	3×3	256	1	1	$256 \times 96 \times 96$
Output			1×1 2D Convolution on $x_{01}, x_{02}, x_{03}, x_{04}$, separately	1×1	1	0	1	$1 \times 96 \times 96$
			Average$(x_{01}, x_{02}, x_{03}, x_{04})$					$1 \times 96 \times 96$
			Interpolation					$1 \times 192 \times 192$

5.2. Implementation details

5.2.1. Machine

The proposed architecture for cloud segmentation was implemented in Pytorch version 1.10.0 on the Google Colaboratory cloud platform using Tesla P100 GPU.

5.2.2. Dataset split

70 % of the dataset was kept for training, while 15 % was used as validation data, and another 15 % was kept for testing purposes.

5.2.3. Image size

The original image size of the dataset is 384×384, which was resized to 192×192 for training and validation datasets, while for the test dataset, it was kept as original 384×384. Thus, we trained the model with input image size 192×192, with 4 channels per image. All the 4 channels, Red, Green, Blue, and NIR were stacked up in that order to create $4 \times 192 \times 192$ input.

5.2.4. Data augmentation

Image augmentation helps the model avoid overfitting as well as limits the need for huge training datasets by augmenting the images and thereby increasing the size of the dataset. This helps the network to learn on different versions of the same images, thus increasing the capability of the network to generalize well on unseen data. There are several data augmentation libraries such as Albumentations, Augmentor, imgaug, etc. These libraries provide various image augmentation techniques such as flipping, rotation, cropping, brightness, scaling, noise addition, etc. Albumentations library is an easy-to-use, flexible, open-source, and faster image augmentation library as compared to other image augmentation libraries (Buslaev et al., 2020). It was used by the topper at various deep learning competitions such as Kaggle, CVPR (Conference on Computer Vision and Pattern Recognition), MICCAI (Medical Image Computing and Computer Assisted Intervention Society), etc. Therefore, we adopted the Albumentations library for data augmentation on the training set. Particularly, we applied image resizing, shift, scale, rotate, brightness, horizontal flip, and vertical flip transformations.

5.2.5. Loss function

Selection of the appropriate loss function is one of the important hyperparameters in any CNN especially in the datasets with high class imbalance as ours. Various alternatives can be adopted as loss functions such as cross-entropy loss, IoU loss, Dice loss, focal loss, Tversky loss, etc. The decision regarding which loss function to use usually depends on the properties of the dataset used for training (Jadon, 2020). We did some preliminary experimentation with various loss functions on a simple U-Net model and compared the results. Dice loss proved to be the best loss function.

Dice loss works better as compared to other loss functions in case of class imbalance problems in the dataset (Jadon, 2020) and also provides a crisper detection of the boundaries of objects (Du, 2020). This research work is concerned with the semantic segmentation of clouds in images. As with most of the other semantic segmentation datasets, the 95-Cloud dataset used in this study for training the deep neural network also suffers from the class imbalance problem with most of the pixels in an image belonging to the 'non-cloud' class and few belonging to the 'cloud' class. Also, the cloud boundaries are very difficult to extract due to their fuzzy appearance most of the time. Thus, based on our preliminary experimentation result, the class imbalance problem in our training dataset, and the requirement of accurately detecting the complex cloud boundaries, we decided to use the Dice loss as the loss function while training the network.

Dice loss (Milletari, Navab, & Ahmadi, 2016) is based on Sørensen–Dice coefficient, a statistic developed in 1940. It is useful in high class imbalance semantic segmentation problems which calculates the overlap between the predicted class (p) and the ground truth class (g) as:

$$D = \frac{2\sum_i^N p_i g_i}{\sum_i^N p_i + \sum_i^N g_i} \tag{2}$$

Here, p_i and g_i are the pixel values of the predicted semantic segmentation mask and the ground truth mask, respectively.

5.2.6. The training loop

Finding an optimal learning rate for training a deep neural network is a big challenge and it is often tackled with a trial-and-error approach. Instead of using a fixed learning rate, we employed the strategy of varying the learning rate after every epoch of training using a 'one cycle learning rate policy' (Smith & Topin, 2017). It allows faster convergence of the training loop and is considered as a state-of-the-art for training deep neural networks. It starts the training using a low learning rate and gradually increases it after every epoch thereby using a high learning rate for about 30 % of the epochs, and then decreases the learning rate gradually for the rest of the epochs. The learning rate versus batch curve is shown in Fig. 7. Starting with a lower learning rate helps the training loop to warm up. It then slowly increases the learning rate as the training loop reaches the middle part where the learning rate works as a regularization method and prevents the network from overfitting by avoiding steep areas of the loss function and landing better flatter minima. Higher learning rates also help the network to learn fast. In the last part of the training, the learning rate decreases gradually to allow the training to cool down and converge at a steeper local minimum. Using this technique, (Smith, 2017) was able to train a ResNet-56 network to achieve an accuracy of 92.3 % on the CIFAR-10 image dataset in just 50 epochs.

5.2.7. Regularization techniques used

Several regularization techniques are used in the training of deep neural networks to keep the models from overfitting the training data and enhancing their generalization capability on the unseen data, thus increasing their prediction accuracy. Weight decay, also known as L_2 regularization works by adding a penalty on the L_2 norm of the weights in the neural network, thereby preventing the weights to explode and encouraging smaller weights. We employed the weight decay regularization technique to prevent model overfitting and to prevent the model weights from becoming large and thus avoid the problem of exploding gradients by adding an extra term to the loss function. Another policy of gradient clipping was also employed to prevent the problem of exploding gradients by restricting the values of gradients to a small range. Both weight decay and gradient clipping prevent the model from overfitting the training data by keeping the weights to smaller values. Image augmentation is another regularization technique used in our experiments, as explained in Section 5.2.4 earlier.

5.2.8. Other hyperparameters for training

The optimization algorithm: We used Adam as the optimization algorithm for training. The role of an optimization algorithm in training is to search for such values for the deep neural network's parameters which minimize the difference between the ground truth and the output produced by the deep neural network, i.e., minimize the error. Thus, the optimization algorithms have a deep impact on the accuracy of the model, and they also affect the speed of training. There are various optimization algorithms such as Stochastic Gradient Descent (SGD), AdaGrad, RMSProp, AdaDelta, Adam, etc. SGD takes a considerable amount of computation time, thus making the training slow. Also, it is not very effective at dealing with the saddling points in the training curve well (Ruder, 2016). Adam (Kingma & Ba, 2014) takes the best of both AdaGrad and RMSProp. It is a fast optimization algorithm that works well in practice and provides better results in fewer epochs than any other optimization algorithm. It is also considered robust to the choice of training hyperparameters (Goodfellow and Yoshua Bengio, 2016). Adam has become more or less a default choice for the optimization algorithm in most deep learning experiments.

Maximum learning rate: In the training loop, we chose the maximum learning rate to be 0.001. Learning rate is an important hyperparameter that determines the scale by which the model's parameters should be updated after every epoch during the training. The learning rates are usually in the range [0.1, 0.00001]. We are using the 'One cycle learning rate policy' (Smith & Topin, 2017), as explained in Section 5.2.6. It starts the training using a low learning rate and gradually increases it after every epoch thereby using a high learning rate for about 30 % of the epochs, and then decreases the learning rate gradually for the rest of the epochs. As shown in Fig. 5, the training starts with a very small learning rate and gradually increases reaching the maximum value of 0.001 somewhere in the middle. Starting with a lower learning rate helps the training loop to warm up. In the middle part, the learning rate works as a regularization method and prevents the network from overfitting by avoiding steep areas of the loss function and landing better flatter minima, also helping the network to learn fast. Then, the learning rate starts decreasing gradually till the end of the training. It allows the training to cool down and converge at a steeper local minimum. Choosing a smaller learning rate slows down the training process and makes the model converge slowly. On the other hand, choosing a larger learning rate makes the model oscillate (Goodfellow and Yoshua Bengio, 2016). Setting the maximum learning rate to be 0.001 allowed our model to converge faster (Izmailov, Podoprikhin, Garipov, Vetrov, & Wilson, 2018).

The number of epochs: In our experiments, we used 100 training epochs. Choosing a value for the number of epochs depends on the dataset and the time the model takes to converge. As shown in Fig. 8, the curves of the training and validation losses while training using the proposed architecture are decreasing steadily and become stable by the end of 100 epochs. This confirms that the model is trained with a good fit.

Batch size: Batch size is another important hyperparameter in

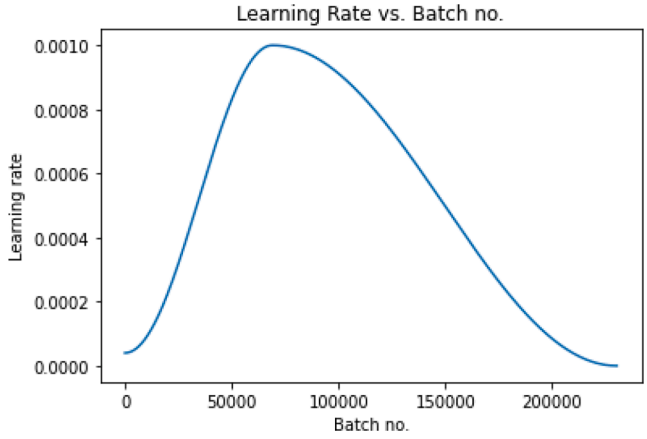

Fig. 7. The learning rate curve of the proposed model during training.

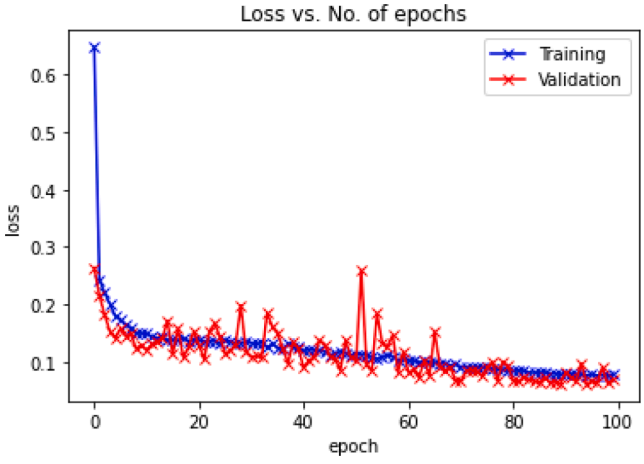

Fig. 8. The loss curve on the training and validation datasets after every epoch while training on SEUNet++.

training a deep neural network. The value of this hyperparameter can be anywhere between 1 and the size of the entire training dataset. In our experiments, we used a small batch size of 12. Smaller batch sizes are preferred as compared to larger ones as using a large batch size reduces the model's ability to generalize (Goodfellow and Yoshua Bengio, 2016). Using a smaller batch size, on the other hand, exhibits a regularizing effect and prevents a model from being overfitting to the training dataset (Smith, 2018).

The various hyperparameters used for training are listed in Table 2.

5.2.9. Evaluation metrics

After training the model, it was evaluated through predictions over the unseen test dataset. The cloud masks generated by the model were compared against the corresponding ground truth masks. Several typical statistical metrics commonly used in the state-of-the-art image segmentation architectures were selected to evaluate the results, which are IoU score, accuracy, precision, and recall.

Accuracy is a standard metric that is used to determine the preciseness of the results predicted by a model. Thus, we used accuracy as one of the evaluation metrics in this work. It is defined as:

$$\text{Accuracy} = \frac{\sum_{i=1}^{M}(tp_i + tn_i)}{\sum_{i=1}^{M}(tp_i + tn_i + fp_i + fn_i)} \tag{3}$$

where t_p, t_n, f_p, and f_n are the numbers of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) pixels for each class in each scene, respectively. Here, TP means that a pixel belonging to a positive class is also predicted as belonging to the positive class by the model. TN means that a pixel belonging to a negative class is correctly predicted as belonging to the negative class by the model. FP means that a pixel belonging to a negative class is wrongly predicted as belonging to the positive class by the model. Similarly, FN means that a pixel belonging to a positive class is wrongly predicted as belonging to the negative class by the model. M is the total number of scenes in the dataset.

As our dataset suffers from a class imbalance between the 'cloud' and 'non-cloud' classes, accuracy may not reflect the true performance of the model (Gonzales & Sakla, 2019). This is because most of the pixels in the images belong to the 'non-cloud' class. For example, let us assume that 70 % of the pixels in an image belong to the 'non-cloud' class and 30 % of the pixels belong to the 'cloud' class. In this case, even if our model wrongly predicts every pixel in an image as belonging to the 'non-cloud' class, still the model will yield an accuracy of 70 %.

Therefore, other metrics are required to effectively evaluate the performance of the model. IoU score is the industry-stand metric used for semantic segmentation (Gonzales & Sakla, 2019). It represents the overlap between the predicted segmentation mask and the ground truth mask which is divided by the total number of pixels in the predicted segmentation mask and the ground truth mask. The value of the IoU score is in the range [0, 1], where 0 represents no overlap between the predicted mask and the ground truth mask and 1 represents a complete overlap. IoU score is defined as:

$$\text{IoU score} = \frac{\sum_{i=1}^{M} tp_i}{\sum_{i=1}^{M}(tp_i + fp_i + fn_i)} \tag{4}$$

Two other metrics, precision, and recall are also used for shedding more light on the performance of the model. Precision is a measure of the quality of predictions made by a model. It measures how many pixels out of all the pixels predicted as belonging to the positive class actually belong to that class. Precision is defined as:

$$\text{Precision} = \frac{\sum_{i=1}^{M} tp_i}{\sum_{i=1}^{M}(tp_i + fp_i)} \tag{5}$$

Recall measures the completeness of the predictions made by the model, i.e., how many predictions have been made correctly. The recall is defined as:

$$\text{Recall} = \frac{\sum_{i=1}^{M} tp_i}{\sum_{i=1}^{M}(tp_i + fn_i)} \tag{6}$$

These metrics provide meaningful insights into understanding the efficacy and performance of the model. We also have used these metrics to compare the performance of our model will that of the state-of-the-art cloud detection models.

Table 2
Hyperparameters used for training.

Loss function	Dice loss
Optimization algorithm	Adam
Maximum learning rate	0.001
No. of epochs	100
Batch size	12

6. Experimental results and discussion

This section presents the results of cloud semantic segmentation obtained by the proposed architecture. To verify the efficacy of the proposed model, we compare it with the other state-of-the-art methods such as Fmask V3, CloudNet, and CloudNet +. For comparison, we have used four evaluation indices, namely, IoU score, accuracy, precision, and recall, as explained in Section 5.2.7. The higher the value of the evaluation indices, the better the model performs.

For the encoder backbone, we experimented with different CNN architectures including ResNet variants, DenseNet-264, CSPResNet-50, and EfficientNet-B8. We adopted four variants of ResNet, i.e., ResNet-18, ResNet-34, ResNet-50, ResNet-101 in our experiments. All these variants are based on the same concept, but with a different number of layers as indicated by the variant name. This choice was based primarily on the available computational resources. ResNet variants such as ResNet-110, ResNet-152, and higher have a greater number of layers and thus require more computational resources and time. Also, using a very large number of layers in the model increases its complexity as the number of parameters to be learned by the model also increases. The time required by the model to converge also increases with the increased number of layers.

We also studied the impact of transfer learning on cloud detection by setting the value of pretrained variable of the selected CNN architecture as encoder to true, that is, using a CNN architecture already trained on the ImageNet dataset. Then, we studied the effect of adding the SE attention module to the proposed architecture. Lastly, we compared the performance of the proposed model with the original U-Net++ model. All the results are obtained using an identical training set up as explained in Section 5.

Table 3 presents the results of using different CNN architectures as the encoder backbone with/without using transfer learning. These models do not have the SE attention module added to their architecture.

6.1. Impact of transfer learning on training

Transfer learning allows to use of a model already pretrained on a vast and diverse dataset on a new learning task without requiring to train the model from scratch (Zhuang et al., 2019). This helps to improve the model's performance, reduces the training time, and can be used to train models on small datasets efficiently as the model is already pretrained on a large dataset. Pretrained models such as AlexNet (Krizhevsky et al., 2012), VGGNet (Simonyan & Zisserman, 2015), ResNet (He et al., 2016), MobileNet (Howard et al., 2017), EfficientNet (Tan & Le, 2021), etc. are usually trained on the ImageNet dataset which contains images related to everyday objects. Satellite images, on the other hand, consist of aerial scenes instead of everyday objects. Our quest here is whether the features learned on everyday objects by a pretrained deep

Table 3
Performance of different CNN architectures as backbones in U-Net++ for cloud segmentation.

U-Net++ backbone	ResNet-18		ResNet-34		ResNet-50		ResNet-101		DenseNet-264		CSPResNet-50		EfficientNet-B8	
Pretrained (with transfer learning)	False	True	False	True	False	True	False	True	False	True	False	True	False	True
IoU score (%)	83.19	89.21	87.23	87.56	88.54	**90.17**	84.08	88.67	87.71	88.55	86.2	89.03	83.97	87.59
Accuracy (%)	92.38	95.48	94.73	94.94	95.08	**95.97**	92.77	95.03	94.91	95.12	94.02	95.52	92.84	94.62
Precision (%)	86.57	90.60	86.95	91.52	89.65	**93.57**	81.89	87.02	88.25	89.38	83.94	90.57	79.73	87.36
Recall (%)	80.19	92.60	93	88.35	92.75	91.42	94.09	96.24	92.72	92.92	95.08	92.24	**96.95**	93.96
Time (hours)	7.72	8.08	8.43	8.77	11.23	10.53	13.87	13.77	16.95	**17.10**	7.86	5.08	11.94	12.28

learning model generalize well to the satellite imagery cloud segmentation domain as well (Penatti, Nogueira, & dos Santos, 2015). Fig. 9 shows the bar charts for comparing the performance of pretrained vs non-pretrained CNN based encoders in U-Net++ based on four parameters, i.e., IoU score, accuracy, precision, and recall. As demonstrated in Fig. 9(a) and (d), pretrained ResNet-18 and ResNet-101 encoders outperformed the non-pretrained ones on all the four evaluation metrics. For example, in the case of ResNet-18 U-Net++, there is an improvement in the IoU score, accuracy, precision, and recall by 6.02 %, 3.1 %, 4.03 %, and 12.41 % respectively, which are quite large margins. Similarly, in the case of ResNet-101 U-Net++, there is an improvement in the IoU score, accuracy, precision, and recall by 4.59 %, 2.26 %, 5.13 %, and 2.15 %, respectively. Using ResNet-34 pretrained encoder also improved by results as compared to non-pretrained ResNet-34 encoder except the recall value, as shown in Fig. 9(b). Similar results were obtained with ResNet-50 pretrained encoder, except for a marginal decrease in the recall value, shown in Fig. 9(c). In the same manner, using pretrained DenseNet-264, CSPResNet-50, and EfficientNet-B8 yields better performance on the cloud segmentation task as compared to using non-pretrained encoder backbones, as shown in Fig. 9(e)-(g). This confirms that using transfer learning exhibits a positive impact on the training and the models pretrained on 3-channel RGB images containing everyday objects can be used as encoders for 4-channel satellite imagery domain, containing images of clouds.

6.2. Comparison of performance among different CNN architectures as encoders in U-Net++

The selected CNN architectures as encoders, i.e., ResNet-18, ResNet-34, ResNet-50, ResNet-101, DenseNet-264, CSPResNet-50, and EfficientNet-B8 used in U-Net++ are compared on the basis of the mentioned evaluation parameters. With the help of a line chart, Fig. 10 (a) compares the results obtained on these 7 CNN variants without using transfer learning and Fig. 10(b) compares the results using transfer learning. In both cases, ResNet-50 emerged out to be the best encoder for U-Net++. In the case of non-transfer learning, as shown in Fig. 10(a), ResNet-50 outperformed the other CNN architectures in terms of IoU score by 0.83 %, accuracy by 0.17 %, and precision by 1.4 % beating DenseNet-264, except the recall metric in which EfficientNet-B8 is the winner with recall value 96.95 % beating CSPResNet-50 by 1.87 %. DenseNet-264 also yielded a good performance next to ResNet-50, but this model is quite complex with a large number of dense connections making the model training process heavy and slow (Liu & Zeng, 2018). Similar yet better results were obtained by using the pretrained ResNet-50 encoder as the backbone in U-Net++. This combination yielded an IoU score of 90.17 %, an accuracy of 95.97 %, a precision of 93.57 %, and a recall of 91.42 %.

More the layers and connections in a model, the more time it takes to complete the execution as indicated in Table 3 with the maximum time taken by DenseNet-264 backbone. This also means that the model is very complex with a large number of parameters.

6.3. Performance of the proposed model with SE attention, i.e., SEUNet++

We selected the best model obtained so far, i.e., ResNet-50 encoder (with transfer learning) with U-Net++ decoder and applied the SE attention module to its decoder blocks to get SEUNet++. Table 4 below presents the results of segmentation performance of SEUNet++ and compares it with the corresponding model without SE attention module, i.e., ResNet-50 encoder with U-Net++ decoder. From the table, it can be seen that adding the attention module benefitted the model's performance and yielded improved results. As compared to the corresponding model without attention, SEUNet++'s IoU score registered an improvement of 1.63 %, which is considerable. Similarly, the accuracy, precision, and recall values improved by 0.98 %, 1.51 %, and 4.35 %, respectively.

6.4. Comparison of the performance of the original U-Net++ model with the proposed model

Table 5 presents the results obtained after training the original U-Net++ model and compares them with our proposed model, SEUNet++. As shown by the results, the proposed model achieved an IoU score of 91.8 % as compared to the IoU score obtained using the original U-Net++, which, is, 89.8 %, thus improving by 2 %. The proposed model also beats the U-Net++ model in terms of accuracy, precision as well as recall.

This means that using a ResNet backbone positively impacted the performance of the model on the cloud segmentation task. This can be credited to the presence of skip connections and residual blocks in ResNet, similar to the attention mechanism, which allows learning the residual between inputs and outputs and resulting in increased accuracy. Also, ResNet as backbone gives better performance in a fewer number of epochs as compared to other backbone networks due to the presence of skip connections which facilitate faster learning and training of deeper networks. Lastly, the addition of SE attention layers in the decoder blocks helped the proposed model to gain further improvements in its performance.

6.5. Performance of SEUNet++ model

Fig. 11(a)-(c) show the loss, IoU score, and accuracy curves generated while training SEUNet++. Fig. 11(a) shows both the training losses as well as the validation losses after every training epoch. Fig. 11(b) and (c) show the IoU score curve and accuracy curve on the validation dataset per epoch during training the architecture. The graphs of the losses are decreasing steadily and become stable by the end of 100 epochs. This confirms that the model is trained with a good fit. The IoU score and accuracy graphs also keep increasing until the end of the training process.

Fig. 12 displays some of the cloud masks produced by the proposed model, SEUNet++. After analyzing some of the cloud masks produced by the proposed model, it can be observed that it produces crisp cloud boundaries, segments even very thin layers of clouds in the satellite

Fig. 9. Bar charts for comparing the performance of pretrained vs non-pretrained CNN architectures as encoders in U-Net++ based on four parameters, i.e., IoU score, accuracy, precision, and recall.

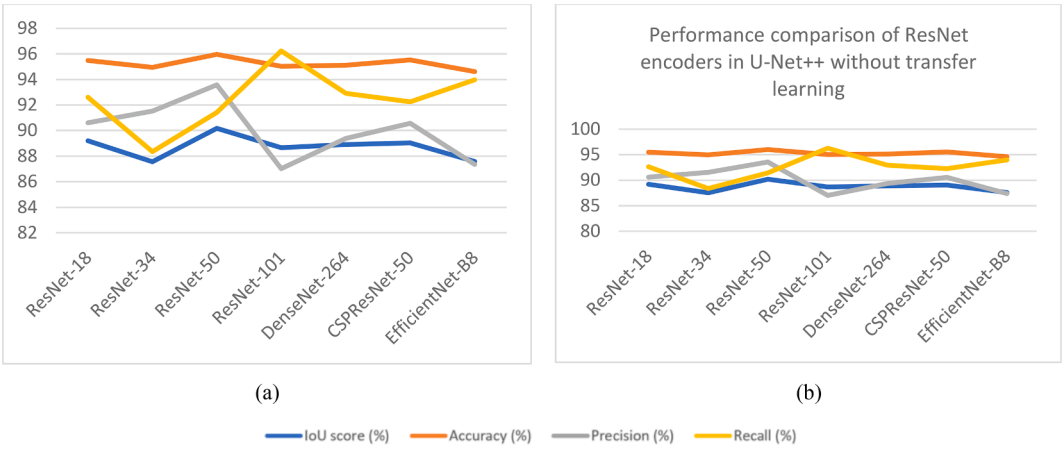

Fig. 10. Performance comparison of different CNN architectures as encoders in U-Net++ (a) without transfer learning and (b) with transfer learning, based on four parameters, i.e., IoU score, accuracy, precision, and recall.

Table 4
Performance of SEUNet++ and its comparison with the corresponding model without SE attention module.

Model	ResNet-50 encoder with U-Net++ decoder (without SE attention)	SEUNet++
IoU score (%)	90.17	91.8
Accuracy (%)	95.97	96.95
Precision (%)	93.57	95.08
Recall (%)	91.42	95.77
Time (hours)	10.53	9.83

Table 5
Comparison of the performance of the original U-Net++ with SEUNet++.

Model	U-Net++	SEUNet++
IoU score (%)	89.8	91.8
Accuracy (%)	95.91	96.95
Precision (%)	89.45	95.08
Recall (%)	94.62	95.77
Time (hours)	9.04	9.83

images and is capable of discriminating cloud shadows. The model is trying to create smooth boundaries of clouds as can be seen in Fig. 12(b), but clouds rarely exhibit smooth boundaries as shown in the true mask of Fig. 12(b).

6.6. Comparison of SEUNet++ with existing cloud segmentation approaches

Table 6 summarizes the comparison of the results of the SEUNet++ architecture with the state-of-the-art deep learning architectures all trained on Landsat 8 image dataset. We adopted Fmask V3 (Zhu et al., 2015), CloudNet (Mohajerani & Saeedi, 2019), and CloudNet+ (Mohajerani & Saeedi, 2021) as benchmarks for comparison. The reason for this is that they reported their results on the same '95-Cloud' dataset which is used in our proposed work. This allows us to benchmark our dataset against these state-of-the-art models and justifies the comparison. Also, CloudNet and CloudNet+ are based on deep learning architectures similar to our proposed model. Thus, we can compare the effectiveness of our proposed deep learning based model against existing deep learning based methods. Fmask V3 is based on thresholding methods and its comparison with the proposed work can provide insights into how much the performance of deep learning based methods differs from the thresholding methods.

From Table 6, it can be observed that our proposed model, SEUNet++ beats CloudNet+ in terms of the IoU score by a margin of 0.23 %. The accuracy, precision, and recall of SEUNet++ are also comparable to the state-of-the-art. This architecture can be trained on any other semantic segmentation task as well and obtain efficient performance. It is also worth mentioning that our proposed model reached the level of performance only in 100 epochs. The results reported by CloudNet used 600 epochs to train the model. This means that our model utilized an 83.33 % lesser number of epochs for training to converge.

When comparing with the thresholding-based method, i.e., FMask V3, the proposed model outperforms on IoU score by 5.89 %, which establishes the fact that deep learning based methods can yield better results on image processing tasks as compared to the traditional

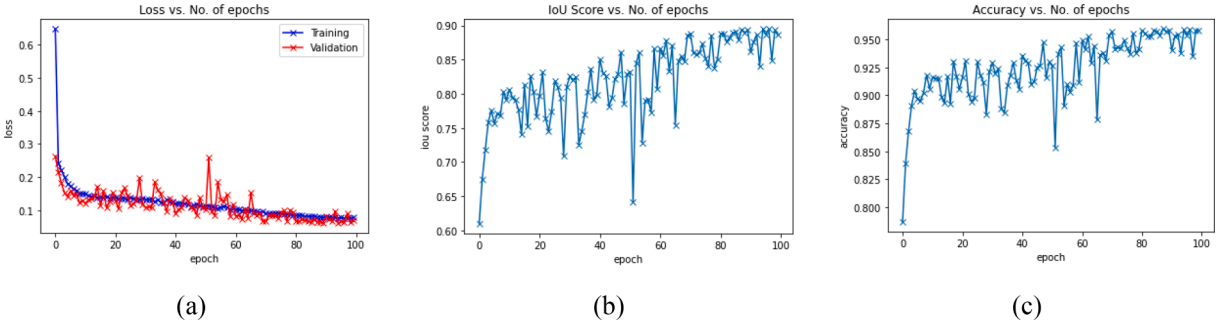

Fig. 11. (a) The loss curve on the training and validation datasets, (b) the IoU score curve on the validation dataset, and (c) the accuracy curve on the validation dataset after every epoch while training on SEUNet++.

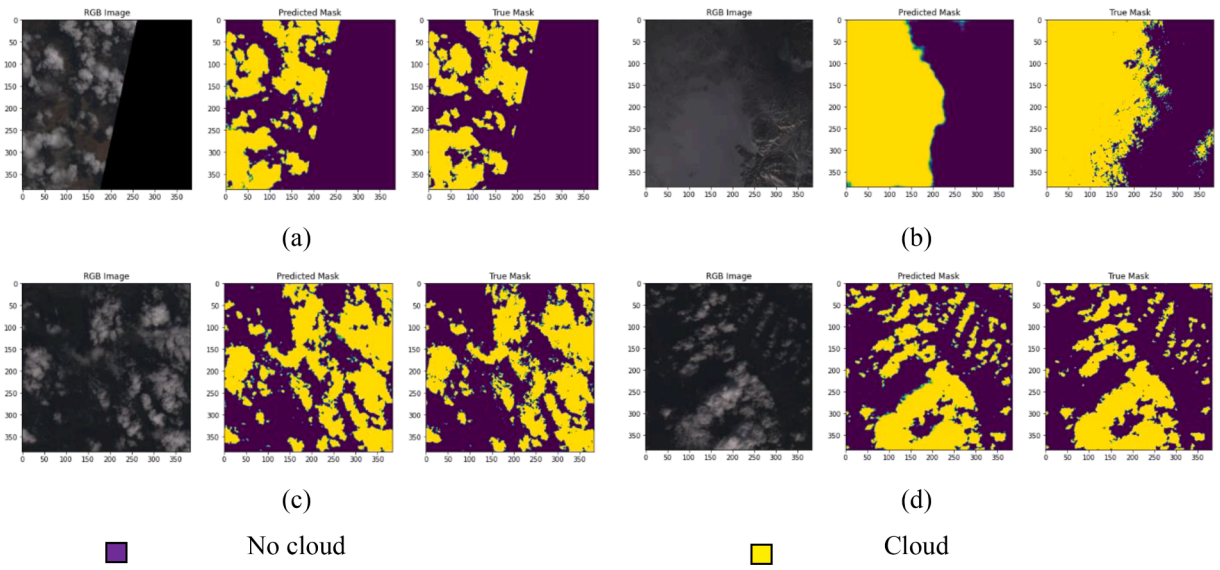

Fig. 12. Some examples of the cloud masks produced by the proposed model SEUNet++.

Table 6
Comparison of the proposed SEUNet++ semantic segmentation method with other methods.

Model	IoU score (%)	Accuracy (%)	Precision (%)	Recall (%)
Fmask V3 (Zhu et al., 2015)	85.91	96.94	88.65	**96.52**
CloudNet (Mohajerani & Saeedi, 2019)	90.83	97.00	**97.67**	92.84
CloudNet+ (Mohajerani & Saeedi, 2021)	91.57	**97.23**	96.94	94.28
Ours (SEUNet++)	**91.8**	96.95	95.08	95.77

methods.

Thus, from the results, it can be determined that our proposed model produces convincing cloud segmentation results, including a significant increase in the IoU score.

7. Conclusions

This paper proposed a deep learning based algorithm to solve the problem of cloud segmentation on the Landsat 8 multispectral dataset, 95-Cloud. Specifically, the proposed model consists of a U-Net++ semantic segmentation model with a lightweight channel attention mechanism. We also experimented with using different encoder backbones in the U-Net++ encoder-decoder architecture such as ResNet variants including ResNet-18, ResNet-34, ResNet-50, and ResNet-101, DenseNet-264, CSPNet, and EfficientNet-B8 and compared their performance. The experimental results show that the proposed architecture achieves an IoU (Intersection over Union) score of 91.8 %. It also boosts the accuracy, precision, and recall values creating crisp cloud boundaries and detecting even thin layers of clouds. We also experimented using transfer learning and found that it has a positive impact on the cloud segmentation task. The proposed model also beats the original U-Net++ architecture in terms of various evaluation metrics such as the IoU score, accuracy, precision, and recall.

We also compared the performance of the proposed model to the other state-of-the-art cloud segmentation models. The results indicate that the performance of our proposed model is akin to that of the best performing model, that, CloudNet+. SEUNet++ yields us an IoU score of 91.8 % which is just 0.23 % higher than CloudNet+. The performance efficacy of our proposed model can be attributed various factors such as the attention modules, skip connections, and residual blocks. The presence of skip connections and residual blocks in ResNet based encoder, similar to the attention mechanism, allows learning the residual between inputs and outputs and resulting in increased accuracy. Also, ResNet as backbone gives better performance in a fewer number of epochs as compared to other backbone networks due to the presence of skip connections which facilitate faster learning and training of deeper networks. Furthermore, the redesigned skip pathways in the U-Net++ decoder result in bridging the semantic gap between the feature maps of the encoder and the decoder paths as the fine-grained feature maps from the encoder sub-network are gradually enriched before blending them with the corresponding feature maps from the decoder sub-network. The dense skip connections were used in U-Net++ to improve the gradient flow. Finally, the application of lightweight SE attention module to the feature maps generated by the convolutional layers in each decoder block helped the model to pay attention to the most important features, which lifted the model IoU score by 1.63 % as compared to the corresponding model without using attention.

To further improve its performance on cloud segmentation, local and global features specific to clouds may be extracted and incorporated into model learning. Other choices for various hyperparameters such as optimization techniques, regularization techniques, augmentation techniques, trying different learning rates and learning strategies, etc. may be investigated in the future.

CRediT authorship contribution statement

Preetpal Kaur Buttar: Conceptualization, Methodology, Software, Formal analysis, Investigation, Data curation, Writing – original draft. **Manoj Kumar Sachan:** Supervision, Writing – review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Afan, H. A., Osman, A. I. A., Essam, Y., Ahmed, A. N., Huang, Y. F., Kisi, O., … El-Shafie, A. (2021). Modeling the fluctuations of groundwater level by employing ensemble deep learning techniques. *Engineering Applications of Computational Fluid Mechanics, 15*(1), 1420–1439. https://doi.org/10.1080/19942060.2021.1974093

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 39*(12), 2481–2495. https://doi.org/10.1109/TPAMI.2016.2644615

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *ArXiv E-Prints*, arXiv:1409.0473. Retrieved from http://arxiv.org/abs/1409.0473.

Bai, Y., Mas, E., & Koshimura, S. (2018). Towards operational satellite-based damage-mapping using U-net convolutional network: A case study of 2011 Tohoku Earthquake-Tsunami. *Remote Sensing, 10*(10). https://doi.org/10.3390/rs10101626

Banan, A., Nasiri, A., & Taheri-Garavand, A. (2020). Deep learning-based appearance features extraction for automated carp species identification. *Aquacultural Engineering, 89*, Article 102053. https://doi.org/10.1016/j.aquaeng.2020.102053

Bharath, K. (2021). U-Net Architecture for Image Segmentation. Retrieved March 21, 2022, from https://blog.paperspace.com/unet-architecture-image-segmentation/.

Bragagnolo, L., da Silva, R. V., & Grzybowski, J. M. V. (2021). Amazon forest cover change mapping based on semantic segmentation by U-Nets. *Ecological Informatics, 62*, Article 101279. https://doi.org/10.1016/j.ecoinf.2021.101279

Bragagnolo, L., Rezende, L. R., da Silva, R. V., & Grzybowski, J. M. V. (2021). Convolutional neural networks applied to semantic segmentation of landslide scars. *CATENA, 201*, Article 105189. https://doi.org/10.1016/j.catena.2021.105189

Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M., & Kalinin, A. A. (2020). Albumentations: Fast and flexible image augmentations. *Information (Switzerland), 11*(2), 1–20. https://doi.org/10.3390/info11020125

Chai, D., Newsam, S., Zhang, H. K., Qiu, Y., & Huang, J. (2019). Cloud and cloud shadow detection in Landsat imagery based on deep convolutional neural networks. *Remote Sensing of Environment, 225*, 307–316. https://doi.org/10.1016/j.rse.2019.03.007

Chen, C., Zhang, Q., Kashani, M. H., Jun, C., Bateni, S. M., Band, S. S., … Chau, K.-W. (2022). Forecast of rainfall distribution based on fixed sliding window long short-term memory. *Engineering Applications of Computational Fluid Mechanics, 16*(1), 248–261. https://doi.org/10.1080/19942060.2021.2009374

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. (2015). Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. *ArXiv:1412.7062*. Retrieved from https://arxiv.org/abs/1412.7062.

Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2018). DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. https://doi.org/10.1109/TPAMI.2017.2699184

Chen, S., Chen, X., Chen, J., Jia, P., Cao, X., & Liu, C. (2016). An iterative haze optimized transformation for automatic cloud/haze detection of landsat imagery. *IEEE Transactions on Geoscience and Remote Sensing, 54*(5), 2682–2694. https://doi.org/10.1109/TGRS.2015.2504369

Du, S. (2020, February 25). Understanding Dice Loss for Crisp Boundary Detection. Retrieved March 22, 2022, from AI Salon website: https://medium.com/ai-salon/understanding-dice-loss-for-crisp-boundary-detection-bb30c2e5f62b.

Dwarampudi, M., & Reddy, N. V. S. (2019). Effects of padding on LSTMs and CNNs. *CoRR, abs/1903.0*. Retrieved from http://arxiv.org/abs/1903.07288.

Fan, Y., Xu, K., Wu, H., Zheng, Y., & Tao, B. (2020). Spatiotemporal modeling for nonlinear distributed thermal processes based on KL decomposition, MLP and LSTM network. *IEEE Access, 8*, 25111–25121. https://doi.org/10.1109/ACCESS.2020.2970836

Francis, A., Sidiropoulos, P., & Muller, J.-P. (2019). CloudFCN: Accurate and robust cloud detection for satellite imagery with deep learning. *Remote Sensing, 11*(19). https://doi.org/10.3390/rs11192312

Fraz, M. M., Javed, M., & Basit, A. (2008). A threshold selection method from gray-level histograms. In *4Th IEEE International Conference on Emerging Technologies* (pp. 232–236).

Fukushima, K. (1980). A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybern., 36*, 193–202.

Gonzales, C., & Sakla, W. (2019). Semantic segmentation of clouds in satellite imagery using deep pre-trained U-nets. *Proceedings - Applied Imagery Pattern Recognition Workshop*. https://doi.org/10.1109/AIPR47015.2019.9174594

Gregor, K., Danihelka, I., Graves, A., Rezende, D., & Wierstra, D. (2015). DRAW: A Recurrent Neural Network For Image Generation. In F. Bach & D. Blei (Eds.), *Proceedings of the 32nd International Conference on Machine Learning* (pp. 1462–1471). Lille, France: PMLR. Retrieved from https://proceedings.mlr.press/v37/gregor15.html.

Guo, J., Yang, J., Yue, H., Tan, H., Hou, C., & Li, K. (2021). CDnetV2: CNN-based cloud detection for remote sensing imagery with cloud-snow coexistence. *IEEE Transactions on Geoscience and Remote Sensing, 59*(1), 700–713. https://doi.org/10.1109/TGRS.2020.2991398

Guo, Y., Cao, X., Liu, B., & Gao, M. (2020). Cloud detection for satellite imagery using attention-based U-net convolutional neural network. *Symmetry, 12*(6). https://doi.org/10.3390/sym12061056

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2016.90

Hou, Y., Liu, Z., Zhang, T., & Li, Y. (2021). C-UNet: Complement UNet for remote sensing road extraction. *Sensors, 21*(6), 1–21. https://doi.org/10.3390/s21062153

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., … Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR, abs/1704.0*. Retrieved from http://arxiv.org/abs/1704.04861.

Howe, J., Pula, K., & Reite, A. A. (2019). Conditional Generative Adversarial Networks for Data Augmentation and Adaptation in Remotely Sensed Imagery. *CoRR, abs/1908.0*. Retrieved from http://arxiv.org/abs/1908.03809.

Hu, J., Shen, L., Albanie, S., Sun, G., & Wu, E. (2020). Squeeze-and-excitation networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 42*(8), 2011–2023. https://doi.org/10.1109/TPAMI.2019.2913372

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017*, 2261–2269. https://doi.org/10.1109/CVPR.2017.243

Huang, Y., Wang, Q., Jia, W., Lu, Y., Li, Y., & He, X. (2021). See more than once: Kernel-sharing atrous convolution for semantic segmentation. *Neurocomputing, 443*, 26–34. https://doi.org/10.1016/j.neucom.2021.02.091

Hughes, M. J., & Hayes, D. J. (2014). Automated Detection of cloud and cloud shadow in single-date landsat imagery using neural networks and spatial post-processing. *Remote Sensing, 6*(6), 4907–4926. https://doi.org/10.3390/rs6064907

Hughes, M. J., & Kennedy, R. (2019). High-quality cloud masking of Landsat 8 imagery using convolutional neural networks. *Remote Sensing, 11*(21). https://doi.org/10.3390/rs11212591

Ian Goodfellow, Yoshua Bengio, A. C. (2016). *Deep Learning*. MIT Press. Retrieved from http://www.deeplearningbook.org.

Iglovikov, V., Mushinskiy, S., & Osin, V. (2017). Satellite Imagery Feature Detection using Deep Convolutional Neural Network: A Kaggle Competition. *ArXiv: 1706.06169*. Retrieved from https://arxiv.org/abs/1706.06169.

Irish, R., Barker, J., Goward, S., & Arvidson, T. (2006). Characterization of the Landsat-7 ETM Automated Cloud-Cover Assessment (ACCA) algorithm. *Photogrammetric Engineering and Remote Sensing, 72*, 1179–1188.

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., & Wilson, A. G. (2018). Averaging weights leads to wider optima and better generalization. *ArXiv Preprint*. ArXiv: 1803.05407.

Jaderberg, M., Simonyan, K., Zisserman, A., & kavukcuoglu, koray. (2015). Spatial Transformer Networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 28). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2015/file/33ceb07bf4eeb3da587e268d663aba1a-Paper.pdf.

Jadon, S. (2020). *A survey of loss functions for semantic segmentation*.

Jeppesen, J. H., Jacobsen, R. H., Inceoglu, F., & Skjødeberg, T. (2019). A cloud detection algorithm for satellite imagery based on deep learning Remote Sensing of Environment A cloud detection algorithm for satellite imagery based on deep learning. *Remote Sensing of Environment, 229*(August), 247–259. https://doi.org/10.1016/j.rse.2019.03.039

Ji, S., Dai, P., Lu, M., & Zhang, Y. (2021). Simultaneous cloud detection and removal from bitemporal remote sensing images using cascade convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing, 59*(1), 732–748. https://doi.org/10.1109/TGRS.2020.2994349

Jiao, L., Huo, L., Hu, C., & Tang, P. (2020a). Refined UNet: UNet-based refinement network for cloud and shadow precise segmentation. *Remote Sensing, 12*(12). https://doi.org/10.3390/rs12122001

Jiao, L., Huo, L., Hu, C., & Tang, P. (2020b). Refined UNet V2: End-to-end patch-wise network for noise-free cloud and shadow segmentation. *Remote Sensing, 12*(21). https://doi.org/10.3390/rs12213530

Kingma, D. P., & Ba, J. L. (2014). Adam: A Method for Stochastic Optimization. *Arxiv: 1412.6980*. Retrieved from https://arxiv.org/abs/1412.6980.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 1097–1105.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM, 60*(6), 84–90. https://doi.org/10.1145/3065386

Li, C., Tan, Y., Chen, W., Luo, X., Gao, Y., Jia, X., & Wang, Z. (2020). Attention Unet++: A nested attention-aware U-Net for Liver CT image segmentation. *IEEE International Conference on Image Processing (ICIP), 2020*, 345–349. https://doi.org/10.1109/ICIP40778.2020.9190761

Li, W., He, C., Fang, J., Zheng, J., Fu, H., & Yu, L. (2019). Semantic segmentation-based building footprint extraction using very high-resolution satellite images and multi-source GIS data. *Remote Sensing, 11*(4). https://doi.org/10.3390/rs11040403

Lin, G., Milan, A., Shen, C., & Reid, I. (2017). RefineNet: Multi-path refinement networks for high-resolution semantic segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017*, 5168–5177. https://doi.org/10.1109/CVPR.2017.549

Liu, W., & Zeng, K. (2018). SparseNet: A Sparse DenseNet for Image Classification. *CoRR, abs/1804.0*. Retrieved from http://arxiv.org/abs/1804.05340.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2015.7298965

Ma, D., Tang, P., & Zhao, L.-J. (2018). SiftingGAN: Generating and Sifting Labeled Samples to Improve the Remote Sensing Image Scene Classification Baseline in vitro. *CoRR, abs/1809.0*. Retrieved from http://arxiv.org/abs/1809.04985.

Mateo-García, G., Gómez-Chova, L., Amorós-López, J., Muñoz-Marí, J., & Camps-Valls, G. (2018). Multitemporal cloud masking in the google earth engine. *Remote Sensing, 10*(7). https://doi.org/10.3390/rs10071079

Milletari, F., Navab, N., & Ahmadi, S. A. (2016). V-Net: Fully convolutional neural networks for volumetric medical image segmentation. *Proceedings - 2016 4th*

International Conference on 3D Vision, 3DV 2016, 565–571. https://doi.org/10.1109/3DV.2016.79.

Mohajerani, S., Asad, R., Abhishek, K., Sharma, N., van Duynhoven, A., & Saeedi, P. (2019). Cloudmaskgan: A content-aware unpaired image-to-image translation algorithm for remote sensing imagery. *IEEE International Conference on Image Processing (ICIP), 2019*, 1965–1969. https://doi.org/10.1109/ICIP.2019.8803161

Mohajerani, S., & Saeedi, P. (2019). Cloud-Net: An end-to-end cloud detection algorithm for landsat 8 imagery. *International Geoscience and Remote Sensing Symposium (IGARSS), 1029–1032*. https://doi.org/10.1109/IGARSS.2019.8898776

Mohajerani, S., & Saeedi, P. (2021). Cloud and cloud shadow segmentation for remote sensing imagery via filtered Jaccard loss function and parametric augmentation. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 14*, 4254–4266. https://doi.org/10.1109/JSTARS.2021.3070786

Neves, A., Körting, T., Fonseca, L., Girolamo Neto, C., Wittich, D., Costa, G., & Heipke, C. (2020). Semantic segmentation of Brazilian Savanna vegetation using high spatial resolution satellite data and U-Net. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *V-3–2020*, 505–511. https://doi.org/10.5194/isprs-annals-V-3-2020-505-2020.

Noh, H., Hong, S., & Han, B. (2015). Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*. https://doi.org/10.1109/ICCV.2015.178

Öztürk, Ş., Özkaya, U., Akdemir, B., & Seyfi, L. (2018). Convolution kernel size effect on convolutional neural network in histopathological image processing applications. *International Symposium on Fundamentals of Electrical Engineering (ISFEE), 2018*, 1–5. https://doi.org/10.1109/ISFEE.2018.8742484

Penatti, O. A. B., Nogueira, K., & dos Santos, J. A. (2015). Do deep features generalize from everyday objects to remote sensing and aerial scenes domains? *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2015*, 44–51. https://doi.org/10.1109/CVPRW.2015.7301382

Peng, C., Zhang, X., Yu, G., Luo, G., & Sun, J. (2017). Large kernel matters — Improve semantic segmentation by global convolutional network. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017*, 1743–1751. https://doi.org/10.1109/CVPR.2017.189

Qiu, S., Zhu, Z., & He, B. (2019). Fmask 4.0: Improved cloud and cloud shadow detection in Landsats 4–8 and Sentinel-2 imagery. *Remote Sensing of Environment, 231*, Article 111205. https://doi.org/10.1016/j.rse.2019.05.024

Rakhlin, A., Davydow, A., & Nikolenko, S. (2018). Land cover classification from satellite imagery with U-Net and Lovász-Softmax loss. *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2018*, 257–2574. https://doi.org/10.1109/CVPRW.2018.00048

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In N. Navab, J. Hornegger, W. M. Wells, & A. F. Frangi (Eds.), *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (pp. 234–241). Cham: Springer International Publishing.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *CoRR*, *abs/1609.0*. Retrieved from http://arxiv.org/abs/1609.04747.

Shamshirband, S., Rabczuk, T., & Chau, K.-W. (2019). A survey of deep learning techniques: Application in wind and solar energy resources. *IEEE Access, 7*, 164650–164666. https://doi.org/10.1109/ACCESS.2019.2951750

Shotton, J., Johnson, M., & Cipolla, R. (2008). Semantic texton forests for image categorization and segmentation. *IEEE Conference on Computer Vision and Pattern Recognition, 2008*, 1–8.

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *3Rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.

Smith, L. N. (2017). Cyclical learning rates for training neural networks. *Proceedings - 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017*, (April), 464–472. https://doi.org/10.1109/WACV.2017.58.

Smith, L. N. (2018). A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay. *CoRR*, *abs/1803.0*. Retrieved from http://arxiv.org/abs/1803.09820.

Smith, L. N., & Topin, N. (2017). Super-Convergence: Very Fast Training of Residual Networks Using Large Learning Rates. *CoRR*, *abs/1708.0*. Retrieved from http://arxiv.org/abs/1708.07120.

Tan, M., & Le, Q. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In K. Chaudhuri, & R. Salakhutdinov (Eds.), *Proceedings of the 36th International Conference on Machine Learning* (pp. 6105–6114).

Tan, M., & Le, Q. V. (2021). *EfficientNetV2 : Smaller Models and Faster Training*.

Ulmas, P., & Liiv, I. (2020). Segmentation of Satellite Imagery using U-Net Models for Land Cover Classification. *ArXiv:2003.02899*. Retrieved from https://arxiv.org/abs/2003.02899.

Wang, C. Y., Mark Liao, H. Y., Wu, Y. H., Chen, P. Y., Hsieh, J. W., & Yeh, I. H. (2020). CSPNet: A new backbone that can enhance learning capability of CNN. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, *2020-June*, 1571–1580. IEEE Computer Society. https://doi.org/10.1109/CVPRW50498.2020.00203.

Woo, S., Park, J., Lee, J. Y., & Kweon, I. S. (2018). CBAM: Convolutional block attention module. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-030-01234-2_1

Wu, H., Zhang, J., Huang, K., Liang, K., & Yu, Y. (2019). FastFCN: Rethinking Dilated Convolution in the Backbone for Semantic Segmentation. *ArXiv:1903.11816*. Retrieved from http://arxiv.org/abs/1903.11816.

Xie, F., Shi, M., Shi, Z., Yin, J., & Zhao, D. (2017). Multilevel cloud detection in remote sensing images based on deep learning. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 10*(8), 3631–3640. https://doi.org/10.1109/JSTARS.2017.2686488

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., … Bengio, Y. (2015). In *Show, Attend and Tell: Neural Image Caption Generation with Visual Attention* (pp. 2048–2057). Lille, France: PMLR.

Xu, L., Wong, A., & Clausi, D. A. (2017). A novel Bayesian spatial-temporal random field model applied to cloud detection from remotely sensed imagery. *IEEE Transactions on Geoscience and Remote Sensing, 55*(9), 4913–4924. https://doi.org/10.1109/TGRS.2017.2692264

Yang, J., Guo, J., Yue, H., Liu, Z., Hu, H., & Li, K. (2019). CDnet: CNN-based cloud detection for remote sensing imagery. *IEEE Transactions on Geoscience and Remote Sensing, 57*(8), 6195–6211. https://doi.org/10.1109/TGRS.2019.2904868

Yi, Y., Zhang, Z., Zhang, W., Zhang, C., Li, W., & Zhao, T. (2019). Semantic segmentation of urban buildings from VHR remote sensing imagery using a deep convolutional neural network. *Remote Sensing, 11*(15). https://doi.org/10.3390/rs11151774

Zhang, Y., Guindon, B., & Li, X. (2014). A robust approach for object-based detection and radiometric characterization of cloud shadow using haze optimized transformation. *IEEE Transactions on Geoscience and Remote Sensing, 52*(9), 5540–5547. https://doi.org/10.1109/TGRS.2013.2290237

Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2017). Pyramid scene parsing network. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017*, 6230–6239.

Zhao, W., Du, S., Wang, Q., & Emery, W. J. (2017). Contextually guided very-high-resolution imagery classification with semantic segments. *ISPRS Journal of Photogrammetry and Remote Sensing, 132*, 48–60. https://doi.org/10.1016/j.isprsjprs.2017.08.011

Zheng, K., Wei, M., Sun, G., Anas, B., & Li, Y. (2019). Using vehicle synthesis generative adversarial networks to improve vehicle detection in remote sensing images. *ISPRS International Journal of Geo-Information, 8*(9). https://doi.org/10.3390/ijgi8090390

Zhou, Z., Rahman Siddiquee, M. M., Tajbakhsh, N., & Liang, J. (2018). UNet++: A Nested U-Net Architecture for Medical Image Segmentation. In D. Stoyanov, Z. Taylor, G. Carneiro, T. Syeda-Mahmood, A. Martel, L. Maier-Hein, … A. Madabhushi (Eds.), *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support* (pp. 3–11). Cham: Springer International Publishing.

Zhu, Z., Wang, S., & Woodcock, C. (2015). Improvement and expansion of the Fmask algorithm: Cloud, cloud shadow, and snow detection for Landsats 4–7, 8, and Sentinel 2 images. *Remote Sensing of Environment, 159*. https://doi.org/10.1016/j.rse.2014.12.014

Zhu, Z., & Woodcock, C. E. (2012). Object-based cloud and cloud shadow detection in Landsat imagery. *Remote Sensing of Environment, 118*, 83–94. https://doi.org/10.1016/j.rse.2011.10.028

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., … He, Q. (2019). A Comprehensive Survey on Transfer Learning. *CoRR*, *abs/1911.0*. Retrieved from http://arxiv.org/abs/1911.02685.

Zi, Y., Xie, F., & Jiang, Z. (2018). A cloud detection method for landsat 8 images based on PCANet. *Remote Sensing, 10*(6). https://doi.org/10.3390/rs10060877