Object based convolutional neural network for cloud classification in very high-resolution hyperspectral imagery

R. Rizkiyanto¹, B. Rabbani¹, D. Y. Perwira¹, and A. M. Arymurthy²

¹Faculty of Computer Science, Universitas Indonesia
²Professor, Faculty of Computer Science, Universitas Indonesian Kampus UI Depok 16424, West Java, Republic of Indonesia
e-mail: rahmat.rizkiyanto@ui.ac.id

Abstract. Remote sensing has a critical role for spatial data-based information systems and monitoring of the earth's surface. The presence of clouds in optical sensing remote sensing satellite images is often a problem for many remote sensing applications. Therefore, the proper detection and classification of clouds in optical sensor remote sensing applications is quite a challenging task. This study aims to classify cloud objects in remote sensing satellite image data. The data used in this study is Pleiades very high-resolution satellite imagery data. The number of datasets used amounted to 1299 data. Cloud objects in this study are categorized into three classes, namely thick cloud, thin cloud, and clear. This study uses a deep learning algorithm, Convolutional Neural Network (CNN) for the classification of cloud objects. The CNN model used is LeNet with architectural modifications and parameters adjusted to the research needs. Classification of cloud objects with the LeNet model results in increased accuracy in each epoch during the training process and takes 1150.355 seconds for 200 epochs with the best accuracy value of 97.50%. The performance of LeNet is better than the VGG16 model as a comparison with the best accuracy of 96.50% with 600 data inputs.

1. Introduction
Remote sensing has a critical role in the information system and monitoring of the earth's surface based on temporal space. In recent times, research using remote sensing image data has been carried out, such as object detection and classification. Some research objects in remote sensing imagery include Land cover [1], cloud on whole-sky image [2], Land cover and Land Use [3], urban land use [4], oil palm tree [5], airport [6], snow [7], and clouds [8], [9], [7].

In optical sensor remote sensing satellite imagery, the presence of clouds often interferes with the process of identifying and delineating earth surface objects and their classification processes. In the field of remote sensing, very high-resolution satellite imagery is widely used in land cover monitoring, target detection, and geographical mapping [10]. Clouds significantly affect the spectral band channel of high-resolution optical images [11]. Accurate cloud detection of remote sensing images is a significant task for many remote sensing applications [12]. The existence of clouds can cause severe problems for many remote sensing applications such as target recognition segmentation, atmospheric correction, and many more [13]. Therefore, the proper detection and classification of clouds in remote sensing applications is a somewhat challenging task.

Many methods are used for image classification, especially remote sensing satellite images. They are ranging from simple classification methods to the use of deep learning. Deep learning is a branch of machine learning algorithm or technique which is currently being studied by data science activists. One of the deep learning models that are popular right now is the Convolutional Neural Network (CNN). CNN is known to be useful in solving problems related to images so that almost all Machine Learning systems related to images are currently based on CNN. Research on CNN also continues to grow, as
The CNN algorithm has shown excellent results in various fields, such as the introduction of bank checks, face detection, pedestrian detection, object detection, object recognition, and classification. Even research conducted by [19] show excellent accuracy, reaching 97.25% for face recognition. Not only that, studies for the detection of clouds and snow using SPOT-6 images also produce high accuracy values which reach 99.16% [7]. The use of Proba-V imagery for cloud detection also results in an accuracy value of more than 90% [9]. Based on these achievements is the basis of the author in using the CNN algorithm to recognize cloud patterns in remote sensing satellite image data. In this study, the classification of cloud objects will be carried out from very high-resolution satellite imagery, which will be categorized into three classes, namely thin cloud, thick cloud, and clear.

This research is expected to be able to produce a model with the following criteria:

- Models that can detect and classify cloud objects from very high-resolution satellite imagery data.
- The model can classify clouds into three classes, namely thin cloud, thick cloud, and clear with high accuracy.

2. Materials and Methods

2.1. Materials

In previous studies, a lot of remote sensing image data is used for cloud detection and classification. Starting from Landsat-8 data [12], SPOT-6 [7], [8], SPOT-7 [8] to sentinel data [20]. This research uses the Pleiades optical sensor Very High-Resolution Satellite Image (CSRST) data. At present, there are two Pleiades satellites that have the same specifications, namely the PLEIADES 1 A satellite which was orbited in 2011 and the Pleiades 1 B satellite which was orbited in 2012.

Pleiades satellite imagery has four multispectral channels with a spatial resolution of 2 meters and a Panchromatic channel with a spatial resolution of 0.5 meters. With these specifications, it is expected to be able to detect and classify clouds well.

| Band       | Wavelength | Spatial Resolution |
|------------|------------|--------------------|
| Band 1 – Blue | 430 – 550 nm | 2 m                |
| Band 2 – Green | 490 – 610 nm | 2 m                |
| Band 3 – Red | 600 – 720 nm | 2 m                |
| Band 4 – NIR | 750 – 950 nm | 2 m                |
| Panchromatic | 480 – 830 nm | 0.5 m              |

This study uses ortho level data that has been corrected radiometrically with the recording date 31 March 2018, and the dimensions of the data are 10621 x 4072 x 3.

2.2. Methods

2.2.1. Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is a type of neural network commonly used in image data. CNN can be used to detect and recognize objects in an image. CNN is not much different from the usual neural network. CNN consists of neurons that have weight, bias, and activation functions.

The CNN architecture consists of an input layer, a convolution layer, pooling layer, fully connected layer, and an output layer [21]. The first convolution layer calculates the convolution of the input image...
with a convolutional kernel [22]. The convolution layer is often followed by an activation function, after the pooling layer becomes a fully connected layer [20] as illustrated in Figure 1.

**Figure 1. CNN standard architecture.**

a. **Convolutional Layer**
   Convolution is the operation of extracting different features from an input. Convolution contains the kernel, which is also called the filter that is applied to the sliding window. After an operation on the input is complete, this filter shifts as much as stride. The matrix multiplication between the kernel and the current input area will be calculated for each operation. If the dimensions of the filter do not match the input, it will be handled with padding. Padding can be divided into the same or valid padding. The same padding will add value around the input border so that the output value has the same dimensions as the input. Valid padding will not add value, so the output dimensions are different from the input.

**Figure 2. Convolutional layer** [23].

b. **Pooling Layer**
   The pooling layer consists of a filter of a specific size and stride that will shift across the entire feature map. Pooling can mean doing downsampling from an image. Pooling functions to reduce the dimensions of features by using methods to conclude contiguous information. There are several different pooling techniques, such as max pooling, mean pooling, and average pooling. This research uses the max-pooling method, which will take the maximum value on each filter shift, as shown in Figure 3.

**Figure 3. Max pooling operation** [23].
c. Fully Connected Layer
The last part of CNN is a fully connected layer, as shown in Figure 3. This layer takes input from all neurons in the previous layer and performs operations of each neuron in the current layer to produce output [23].

![Figure 4. Fully connected layer [23].](image)


d. ReLu Activation Function
In ANN, the signal between neurons is influenced by weight, which is usually determined randomly as the initial value. Optimal weight can be obtained by doing some calculations on the input data and weight itself. The calculation is also carried out to achieve the appropriate output. This calculation is a linear combination between input and weight followed by ReLU (Rectified Linear Unit) Activation Function using the formula in equation 1. Activation Function is used after Convolution and Max Pooling.

\[ f(x) = \max(0, x) \]  

where \( f(x) \) is the output of the activation function.

2.2.2. LeNet-5
In 1998, LeCun et al. introduced the CNN model to classify digits in handwriting. Their CNN model, called LeNet-5 [14], as shown in Figure 5, has seven lighted (trainable) layers. Among them, two (C1, C3) convolutional layers, two (S2, S4) average pooling layers, one (F6) fully connected layer, and one output layer.

![Figure 5. Architecture of LeNet-5 [14].](image)
2.2.3. **Adam**
Adam adalah algoritma optimasi object function yang sering digunakan dalam deep neural network. Dalam algoritma ini, proses perubahan parameter tergantung pada gradient, learning rate, nilai momen pertama dan kedua dari gradient [24].

2.2.4. **Dropout**
Overfitting is a severe problem in deep neural networks. Dropout is a technique to prevent overfitting. The main idea of the dropout is to ignore some randomly selected units during training [25]. During training, neglected units will not affect the next layer and will not experience weight changes, which can reduce overfitting.

![Dropout neural network model](image)

**Figure 6.** Dropout neural network model. (a) A standard neural network with 2 hidden layers. (b) An example of neural network by applying dropout to the network on the left. Crossed units have been dropped [25].

### 3. Results and Discussion

#### 3.1. **Experimental Configuration**
Overall experiments conducted in this study use a platform released by Google Internal Research called Google Collaboratory (Google Colab). This platform is similar to a Jupyter Notebook, and they built above the Jupyter environment. The difference is that Google Colab runs on Google's cloud and integrated with Google Drive. The experiment runs on two Intel (R) Xeon (R) CPUs @ 2.2GHz with 12GB of RAM and an NVIDIA Tesla K80 GPU. Data storage media uses a 300GB disk provided by Google Colab service and integrated with Google Drive.

#### 3.2. **Hyperspectral Image Dataset**
In this study, the dataset was created using the Geospatial Data Abstraction Library (GDAL). The dataset is produced from a single Pleiades image data which is cut based on raster data containing data classes. The dataset is divided into three data classes, namely thick cloud, thin cloud, and clear. This study uses 1299 datasets, which are divided into data train, test data, and data validation. The dataset used will be split with 80% split size which means 80% of the data will be used as a data train, and 20% of the data will be used as a test data.
Table 2. Remote Sensing Image Dataset: thick clouds, thin clouds, and clear

| Thick clouds | Thin clouds | Clear |
|--------------|-------------|-------|
| ![Image of Thick Clouds](image1) | ![Image of Thin Clouds](image2) | ![Image of Clear Sky](image3) |
| ![Image of Thick Clouds](image4) | ![Image of Thin Clouds](image5) | ![Image of Clear Sky](image6) |
| ![Image of Thick Clouds](image7) | ![Image of Thin Clouds](image8) | ![Image of Clear Sky](image9) |
| ![Image of Thick Clouds](image10) | ![Image of Thin Clouds](image11) | ![Image of Clear Sky](image12) |

3.3. Model Architecture
Model testing is done by making a cloud object dataset in the Pleiades image as input into the architecture of the CNN LeNet model. Input parameters in the CNN LeNet model architecture use multiple epochs with batch size 64. This study uses Conv2D layer, MaxPooling2D, and Flatten. The configuration and parameters of the CNN architecture used for cloud classification are shown in Tables 3 and 4.

Table 3. Configuration of the CNN Architecture Used for Classifying Clouds

| CNN proposed topologies | Convolutional Layers | Fully Connected Layer |
|-------------------------|----------------------|-----------------------|
| Kernel size | Activation function | Pooling | Regularization | Neurons | Activation Function | Regularization |
| 5 x 5 x 6 | ReLU | 2 x 2 | No | 120 | ReLU | Dropout (0.2) |
| 5 x 5 x 16 | ReLU | 2 x 2 | No | 84 | ReLU | Dropout (0.2) |
| 3 | Softmax | No | | | | |
Table 4. Parameter of the CNN Architecture Used for Classifying Clouds

| Common Parameters | Batch size | Epoch | Learning rate | Optimizer |
|-------------------|------------|-------|---------------|-----------|
|                   | 64         | 200   | 0.001         | Adam      |

The CNN architecture applied has two convolution layers and two pooling layers. The first convolution layer has six filters and a 5x5 kernel size. A max-pooling layer of 2x2 size follows the layer. Then, the second convolution layer has 16 filters with a size of 5x5. This layer is also followed by the max-pooling layer, which is identical to the first pooling layer. We added two fully connected layers, each with 120 and 84 nodes in the classification process. At the output layer, three nodes describe the number of classes, namely clear, thin cloud, and thick cloud. All layers in this architecture use Rectifier Linear Units (ReLU) as an activation function, except for the output layer that uses the softmax activation function.

![CNN Architecture Diagram](image)

**Figure 7.** The CNN architecture used for classifying clouds.

![CNN Model Diagram](image)

**Figure 8.** CNN model used for classifying clouds.
In testing the model, validation data is used to see the accuracy value at each epoch. After the introduction process by the model is complete, the results of the training are stored in the form of processing time and accuracy values for each epoch. An architectural model is said to be good if the accuracy value at each epoch has increased. Therefore, the CNN model depends on determining the architecture of the model. If the architecture of the model is not good, then the accuracy value on the CNN model will also be less good.

3.4. Hyperparameter Tuning

3.4.1. Cross Validation
We use the cross-validation method by dividing the initial data into three folds to get better evaluation results. Then, we make each fold test data and make the other two folds for the training process. So, there are three models with three different test data. Final accuracy is obtained by calculating the average of the three accuracies of each model.

3.4.2. Grid Search
The grid search method is widely used in machine learning applications to get hyperparameters that are capable of producing the best performance. This study uses grid search to look for two hyperparameters, namely the size of the kernel and the number of feature maps in the two convolution layers. We conducted experiments with various values, including assigning different values to the two convolution layers. The conclusion is with a value of 5 on both layers and with values 6 and 16 on the first and second layers able to provide the highest accuracy. Besides, the hyperparameter also produces a model with stable performance.

3.4.3. Batch Normalization
We experimented using batch normalization to improve the stability of the model. Several experiments have been conducted relating to the placement of this method in the model architecture. From these experiments, it was concluded that the use of batch normalization, both at the fully connected layer and at the convolution layer would result in worse and unstable performance. Therefore, this method is not used in the final model.

3.4.4. Dropout
Dropout is a popular method of regularization and has been proven to reduce the risk of overfitting. This model attaches a dropout module to each hidden layer and uses a hyperparameter of 0.2. This method can reduce the effects of overfitting that can be observed from the differences in the results of the accuracy of test data and training data that is getting thinner.

3.4.5. Epoch
In the beginning, we only exported up to 50 epochs. However, at the end of the training process with these values, the model still looks unstable. So, we experimented to find a better epoch value. In the end, an epoch of 200 is capable of delivering results with good accuracy and stability.

3.5. Performance Evaluation
We do accuracy tests on each dataset class. Quantitative accuracy evaluation results for cloud classification shown in table 4. The classification metric shows that the model has relatively high precision accuracy for all classes. Likewise, with the recall value and the f1-score gets a relatively high value. The clear class gets a precision value of 0.98, the thin cloud class gets a precision value of 0.92, and the thick cloud class gets the highest precision value of 1.00. These results illustrate that the model built in this study is valid.
Table 5. Cloud Classification Metric

|               | Precision | Recall | F1-score |
|---------------|-----------|--------|----------|
| Clear         | 0.98      | 0.90   | 0.94     |
| Thin cloud    | 0.92      | 0.99   | 0.95     |
| Thick cloud   | 1.00      | 1.00   | 1.00     |

Besides, the result of the confusion matrix also shows how well the model is predicted for each class and how often the model predictions are wrong. The confusion matrix in Figure 9 shows that there are prediction errors in a clear and thin cloud class. As much as 10% of the total data with clear class is predicted as a thin cloud class. Then, for a thin cloud class, as much as 1% of the total data is predicted as a clear class.

Figure 9. Normalized confusion matrix.

3.6. Classification Result
In the final model, we divide the dataset into training data and validation data. In this model hyperparameter is used which produces the best accuracy based on experiments with cross-validation, so we get a model that uses 80 percent of the available data and is evaluated with another 20 percent of the data. This model gets the best accuracy for 200 epoch training processes in 1150,355 seconds by 97.5%.

Table 6. The Results of the Classification Accuracy of the Lenet Model

| Accuracy Table |
|----------------|
| Epoch | Time (second) | Accuracy (%) | Loss | Number of train images | Number of validate images |
|-------|---------------|---------------|------|------------------------|--------------------------|
| 10    | 810.821       | 72.500        | 0.559| 1299                   | 433                      |
| 20    | 929.821       | 85.500        | 0.335| 1299                   | 433                      |
| 50    | 980.092       | 91.000        | 0.199| 1299                   | 433                      |
| 100   | 1055.352      | 96.500        | 0.095| 1299                   | 433                      |
| 200   | 1150.355      | 97.500        | 0.043| 1299                   | 433                      |
In Figure 10, it can be seen that the performance of the model is very bad in epoch 1 to 50. However, in epoch 150 to 200 the volatility of accuracy decreases and is stable.

3.7. Comparison with Other CNN Models
The VGG16 model used as a comparison of the LeNet model used in this research. The dataset used amounts to 600 training data and 90 validation data. The results of experiments with the VGG16 model shown in table 7.

| Epoch | Time (second) | Accuracy (%) | Loss | Number of train images | Number of validate images |
|-------|---------------|--------------|------|------------------------|--------------------------|
| 10    | 469.211       | 85.560       | 0.516| 600                    | 90                       |
| 20    | 436.518       | 89.000       | 0.195| 600                    | 90                       |
| 50    | 483.743       | 91.780       | 0.180| 600                    | 90                       |
| 100   | 506.865       | 95.520       | 0.115| 600                    | 90                       |
| 200   | 653.299       | 96.500       | 0.090| 600                    | 90                       |

The VGG16 model ran ten epoch training processes in 469,211 seconds and got the best accuracy value of 85.56%. The accuracy of the VGG16 model increases with many training processes. In table 7, it seen that for 200 epoch training processes, this model gets the best accuracy value of 96.50%. The accuracy is smaller than the LeNet model. However, with this accuracy, it turns out that there is still a loss of 0.090 in the model, which means that the VGG16 model still does not correctly recognize the cloud object in the Pleiades image. This accuracy also obtained by entering fewer data than the LeNet model. Besides, the use of different parameters with the LeNet model also causes differences in the accuracy value between the two models.

4. Conclusion
The LeNet model architecture can be used for the classification of cloud objects in very high-resolution remote sensing satellite images. The cloud classes recognized by this model are thick cloud, thin cloud, and clear. An architectural model is said to be good if the accuracy value at each epoch has increased. The LeNet model obtains increasing accuracy in each epoch during the training process and takes 1150,355 seconds for 200 epochs with the best accuracy value of 97.50%. The model in this study will be even better if it uses a more in-depth architecture. For example, adding CNN architecture or can be
called double CNN. The use of a more varied dataset can also increase the value of the accuracy of the model being applied.

5. Acknowledgement
Authors would like to Thank Remote Sensing Technology and Data Center LAPAN for providing the opportunity for the author to obtain the dataset needed in this study.

6. References
[1] M. E. Paoletti, J. M. Haut, J. Plaza, and A. Plaza, “A new deep convolutional neural network for fast hyperspectral image classification,” ISPRS Journal of Photogrammetry and Remote Sensing, 2017.
[2] A. Taravat, F. Del Frate, C. Comaro, and S. Vergari, “Neural networks and support vector machine algorithms for automatic cloud classification of whole-sky ground-based images,” IEEE Geosci. Remote Sens. Lett., vol. 12, no. 3, pp. 666–670, 2015.
[3] C. Yang, F. Rottensteiner, and C. Heipke, “CLASSIFICATION of LAND COVER and LAND USE BASED ON CONVOLUTIONAL NEURAL NETWORKS,” ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci., vol. 4, no. 3, pp. 251–258, 2018.
[4] C. Zhang et al., “An object-based convolutional neural network (OCNN) for urban land use classification,” Remote Sens. Environ., vol. 216, pp. 57–70, Oct. 2018.
[5] W. Li, H. Fu, and L. Yu, “Deep convolutional neural network based large-scale oil palm tree detection for high-resolution remote sensing images,” Int. Geosci. Remote Sens. Symp., vol. 2017–July, pp. 846–849, 2017.
[6] F. Chen, R. Ren, T. Van de Voorde, W. Xu, G. Zhou, and Y. Zhou, “Fast automatic airport detection in remote sensing images using convolutional neural networks,” Remote Sens., vol. 10, no. 3, pp. 1–20, 2018.
[7] L. Wang, Y. Chen, L. Tang, R. Fan, and Y. Yao, “Object-based convolutional neural networks for cloud and snow detection in high-resolution multispectral imagers,” Water (Switzerland), vol. 10, no. 11, 2018.
[8] M. Le Goff, J.-Y. Tourneret, H. Wendt, M. Ortner, and M. Spigai, “Deep Learning for Cloud Detection,” 8th Int. Conf. Pattern Recognit. Syst. (ICPRS 2017), p. 10 (6.-)10 (6.-), 2017.
[9] G. Luis and J. Mu, “CLOUD DETECTION MACHINE LEARNING ALGORITHMS FOR PROBA-V Image Processing Laboratory (IPL), University of Valencia, Spain,” pp. 2251–2254, 2017.
[10] V. Kovalskyy and D. P. Roy, “A one year Landsat 8 conterminous United States study of cirrus and non-cirrus clouds,” Remote Sens., vol. 7, no. 1, pp. 564–578, 2015.
[11] X. Xu, Y. Guo, and Z. Wang, “Cloud image detection based on Markov Random Field,” J. Electron., vol. 29, no. 3–4, pp. 262–270, 2012.
[12] Z. Zhu and C. E. Woodcock, “Object-based cloud and cloud shadow detection in Landsat imagery,” Remote Sens. Environ., vol. 118, pp. 83–94, 2012.
[13] H. Shen, H. Li, Y. Qian, L. Zhang, and Q. Yuan, “An effective thin cloud removal procedure for visible remote sensing images,” ISPRS J. Photogramm. Remote Sens., vol. 96, pp. 224–235, 2014.
[14] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, “Object Recognition with Gradient-Based Learning,” no. c, pp. 2–6, 1998.
[15] A. Krizhevsky, I. Sutskever, and Geoffrey e. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” Handb. Approx. Algorithms Metaheuristics, pp. 60-1-60-16, 2007.
[16] K. Simonyan, A. Vedaldi, and A. Zisserman, “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps,” pp. 1–8, 2014.
[17] C. Szegedy et al., “Going Deeper with Convolutions Christian,” IEEE Conf. Comput. Vis. Pattern Recognit., 2015.
[18] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” Proc. IEEE
Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2016–Decem, pp. 770–778, 2016.

[19] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “DeepFace: Closing the gap to human-level performance in face verification,” Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 1701–1708, 2014.

[20] G. Scarpa, M. Gargiulo, A. Mazza, and R. Gaetano, “A CNN-based fusion method for feature extraction from sentinel data,” Remote Sens., vol. 10, no. 2, pp. 1–20, 2018.

[21] C. Dong, C. C. Loy, K. He, and X. Tang, “Image Super-Resolution Using Deep Convolutional Networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 38, no. 2, pp. 295–307, 2016.

[22] D. Pouliot, R. Latifovic, J. Pasher, and J. Duffe, “Landsat super-resolution enhancement using convolution neural networks and Sentinel-2 for training,” Remote Sens., vol. 10, no. 3, 2018.

[23] F. Sultana, A. Sufian, and P. Dutta, “Advancements in image classification using convolutional neural network,” Proc. - 2018 4th IEEE Int. Conf. Res. Comput. Intell. Commun. Networks, ICRCICN 2018, pp. 122–129, 2019.

[24] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” pp. 1–15, 2014.

[25] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting Nitish,” J. Mach. Learn. Res., vol. 15, pp. 1929–1958, 2014.