Comparison of CNNs for Remote Sensing Scene Classification

Mayar A. Shafaey[1]*, Mohammed A.-M. Salem[1,2], Maryam N. Al-Berry[1], Hala M. Ebied[1], Mohamed F. Tolba[1]
Scientific Computing dept.
[1] Faculty of Computers and Information Sciences, Ain Shams University
[2] Faculty of Media Engineering and Technology, German University in Cairo
Cairo, Egypt
mayar.al.mohamed@fcis.asu.edu.eg*, salem@cis.asu.edu.eg, maryam_nabil@cis.asu.edu.eg, halam@cis.asu.edu.eg, fahmytolba@cis.asu.edu.eg

Abstract— **Nowadays, the Convolutional Neural Networks (CNNs) are used in many remote-sensing applications sush as earth observation, urban planning, earth's scene classification, and so on. The deep learning manner, especially CNNs, has proved its accuracy for these practical applications. Hence, in this article, CNNs models are reviewed and its five different architectures are applied for comparisons; namely, *AlexNet, VGGNet, GoogleNet, Inception-V3*, and *ResNet-101.* These models are carried out on seven different remote-sensing image datasets for image scene classification purpose; namely, *WHU-RS19, UC-Merced Land Use, SIRI-WHU, RSSCN7, AID, PatternNet,* and *NWPU-RESISC45.* These datasets have different spatial resolutions, ranging from 0.2 to 30, to differentiate the classification accuracy from low to high-resolution images. As well, the classification accuracy of each model is assessed by using five different classifiers; namely, *Naïve Bayes, Decision Tree, Random Forest*, *K-Nearest Neighbor (KNN),* and *Support Vector Machine (SVM).* The best accuracy credits to ResNet-101 model with SVM classifier; it has reached about 98.6±0.02 % of the high-resolution dataset *PatternNet.***

Keywords— Convolutional Neural Networks; Deep Learning; Remote-Sensing; Satellite Images; Scene Classification

I. INTRODUCTION

For the time being, the artificial satellites can scan the whole earth in less than a day. Those satellites produce high spatial resolution images which are considered as large-scale ones. The analysis and classification of those remote-sensing images have been considered hot topics recently. It is very important to analyze and classify remote-sensing images in everyday applications, including urban planning, earth observation, military monitoring, natural hazards detection, and vegetation mapping [1]. For those reasons, many researchers carried out their experiments on the satellite images for the scene classification mission.

The volcano of research depending on supervised learning methods has started in 2006. The supervised manner use labeled data to extract more powerful features, especially, a deep learning method which proposed by *Hinton and Salakhutdinov* [2]. Different models of deep learning are introduced, i.e. deep belief nets (DBN) [3], deep Boltzmann machines (DBM) [4], stacked auto-encoder (SAE) [5], Convolutional Neural Networks (CNNs) [6], etc.

However, the most widely used deep learning method is the CNNs. For instance, in [7], the authors proposed a method for automatic object detection based on a CNN. The proposed training approach has been tested using a UC-Merced dataset [8] of the aerial image and achieved an accuracy of 98.6% approximately. In [9], a multi-scale input strategy for deep learning was planned for supervised multispectral land-use classification. It had been shown that a single Deep CNN may be trained at the same time with multi-scale views to boost the prediction accuracy over multiple single-scale views. The valuable performance was achieved on the UC Merced dataset [8]; it was achieved 93.48% of accuracy for multi-view deep learning. In [10], the authors proposed a practical CNN architecture called the large patch CNN for remote-sensing scene classification. Their experiments achieved a performance that was like to the state-of-the-art on public remote-sensing scene datasets. Also, in [11-15, 41], a deep learning method such as CNNs was investigated for the scene and semantic classification of remote sensing scenes, which guaranteed a significant performance improvement comparing with the state-of-the-art.

Therefore, the main objective of this paper is to provide a quick description of the CNN model, its layers, and its different architectures, as in section 2. As well, to review the common classifiers which are used with different architectures of CNN, section 3. Moreover, to record the results of applied experiments as in section 4, and to discuss those results in section 5. Finally, the important comments will be highlighted in the conclusion section.

II. CNN ARCHITECTURE AND MODELS

Convolutional Neural Networks are very comparable to traditional Neural Networks [16]. They consist of neurons having learnable weights and biases and all the tips developed for regular Neural Networks learning still apply. The main difference is that CNN architectures assume that the inputs are images, which allows encoding certain properties into the architecture. These properties then make the forward function implemented more efficient and vastly minimize the parameters number in the network [17].

In CNNs, the input image is passed through a series of *convolutional, pooling,* and *fully connected layers*, and then we get an *output layer*, see Figure1. The output can be a single class or a probability of classes that best describes the image.

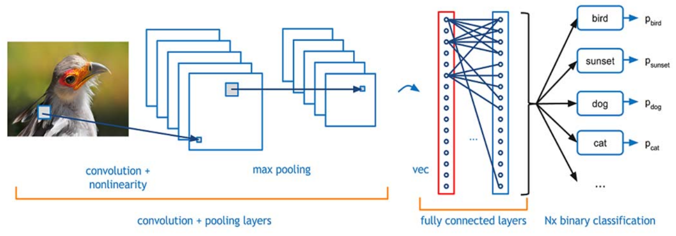


Fig. 1. The main layers of CNN. Starting from the left, an input image, passing through *convolutional, nonlinear, pooling,* and *fully connected layers*, and finally getting an output as a single class [16]

In principle, a convolution operation is applied to the input by the convolutional layers, and then the result is passed to the subsequent layer. Each convolutional neuron processes data for its appropriate field. Despite that the traditional fully connected neural networks and shallow architecture can be used to learn features and classify data; it is not effective to use this architecture to images input. It is required a large number of neurons. Imagine we have a small image with size [50 × 50], and then it has 2500 weights for each neuron in the next layer, where each pixel is a relevant variable.

The convolution operation solves this problem by allowing the network to be deeper with a few parameters. Regardless of image size, tiling regions with size [5 × 5], each with the same shared weights, requires only 25 learnable parameters. By this manner, it solves the problem of the exploding gradients of training the ordinary multi-layer neural networks using back-propagation fashion.

CNNs may include pooling layers, which join the outputs of neuron clusters at certain layer into one neuron at the subsequent layer. For instance, max pooling uses the maximum value from each cluster of neurons at the preceding layer. Another operation is average pooling, which uses the average value from each cluster of neurons instead of the maximum value [16-17].

Finally, the fully connected layers seem like the traditional Multi-Layer Perceptron neural network. It combines every neuron in certain layer to every neuron in beyond layer.

However, there are some frequently used architectures of CNN model; namely, *AlexNet* [18], *VGGNet* [19], *GoogleNet* [20], *Inception-V3* [21], and *ResNet-101*[22]. All those networks are trained on *ImageNet* data [23], which contains over 15 million marked images out of over 22,000 categories. These architectures are used for classification with 1000 possible categories. We can categorize these architectures into three categories, as shown in Figure 2:

- Classic models, which contain (*AlexNet,* and *VGGNet*)
- Inception models, which contain (*GoogleNet,* and *Inception-V3*), and
- Residual models, which contain (*ResNet-101*)

Firstly, the *AlexNet* network [18] contains 5 convolution layers, 5 max-pooling layers, 5 dropout layers, and 3 fully connected layers. It was developed in 2012 and trained on two GTX 580 GPUs for five to six days. The *VGGNet* [19] is the enhanced version of *AlexNet*. It has two versions, vggNet-16 and vggNet-19, where difference is in the number of layers. It was developed in 2014 and trained on 4 GPUs: Nvidia Titan Black, for 2 - 3 weeks.

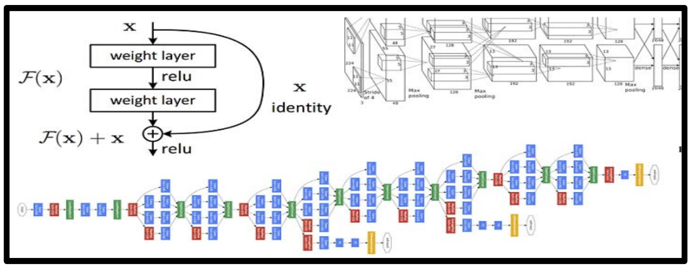


Fig. 2. Three categories of CNN models. The upper right sub-figure points to the classical model. The upper left sub-figure points to residual block, and the bottom sub-figure points to the inception blocks and model

Another category is based on the inception model. For instance, *GoogleNet* [20] is a 22 layer CNN that was developed in 2015. It was one of the first CNN architectures that deviated from the classical approach and sequential structure of simply stacking convolution and pooling layers upper of each other. This model placed notable consideration on memory and power usage. By the first look at the structure of *GoogleNet*, we sign that the stages are run in parallel not as the previous models sequentially. It is trained on a few GPUs within a week. However, *Inception-V3* [21] is the enhanced version of *GoogleNet* and trained on 3 GPUs for 2 - 3 weeks.

Thirdly, the idea behind a residual block, as *ResNet-101* [22], is that the input x pass through convolution-relu-convolution series. The result after passing will give some $F(x)$. This result is then added to the original x. If we imagine that $H(x) = F(x) + x$, then, in traditional CNNs, $H(x)$ is equal to $F(x)$. But, in residual model, The term $F(x)$ is added to the input x, which keeps the information about the original x. However, this model was trained on an 8 GPUs machine for 2 to 3 weeks because it is ultra-deep, in our case, it is 101 layers.

III. Common Classifiers

The classification process is defined as grouping the individual items based on the similarity and the description of the group [23]. The subsequent subsections have a detailed description of each classifier used in our work. In this paper, the classification process is based on five popular classifiers listed as follows: *Naïve Bayes (NB)* [24], *Decision Tree (DT)* [25], *Random forest (RF)* [26], *K-nearest neighbor (KNN)* [27], and *Support Vector Machine (SVM)* [28].

A. Naïve Bayes – NB

NB is a probabilistic classifier that based on Bayes' theorem [24]. It assumes that the existence of a particular feature in a class is unrelated to the existence of any other feature, as in the following equation:

$$P(c|x) = P(x|c)\,P(c)\,/\,P(x) \qquad (1)$$

Where *P(x)*: the prior probability of the predictor, *P(c):* the prior probability of the class, *P(x|c)*: the probability of predictor given class, and *P(c|x)*: the posterior probability of class (*c*, target) given predictor (*x*, features).

B. Decision Tree – DT

It is a simple and commonly used classification technique [25]. It applies a forthright idea for the classification problem solving. It poses a sequence of wisely crafted questions about the attributes of the testing question. Every time it obtains an answer, a next question is asked until reaching the class label of the record.

The root and internal nodes of the decision trees contain set of test conditions to distinct the records that have different features. Also, each terminal node has a class label "Yes" or "No". The record classification is going direct; beginning from the root node, the test condition is applied and follows the proper branch based on the result of the test. After reaching the leaf node, the class label associated with it is assigned to the test record.

While decision trees classify rapidly, the building time for a tree may be higher than another classifier, which is a serious problem when the number of classes increases [29].

C. Random Forest – RF

RF is one of the most recent classifiers [26]. It is a supervised learning algorithm. It creates a forest and somehow makes it random. The forest it constructs is an ensemble of Decision Trees, trained with the "bagging" method [30].

To say it in simple words: RF builds multiple DTs and merges them together to get a more accurate and stable prediction. It is a classifier suggested by *Bremein* [31] which offerings a lot of positive features:

- It handles a lot of input features i.e. thousands without deletion.
- It provides the essential features in the classification.
- It produces an internal unbiased approximation of the generalization error.
- For outliers and noise, it is very robust.
- RF is more efficient in complexity than other tree collaborative methods.
- It is a combination of classifiers and each classifier gives a vote to the most frequent class to the input *x*,

$$C_{rf}^{B} = \text{majority vote } \{C_b(x)\}_1^{B} \qquad (2)$$

where C_{rf}^{B} is the class prediction of the b^{th} RF tree.

The difference between DT and RF is that deep decision trees might suffer from over-fitting. RF prevents over-fitting most of the time, by creating random subsets of the features and building smaller trees using these subsets. Afterward, it combines the subtrees. It doesn't work every time and the computation becomes slower depending on how many trees the random forest builds.

D. K-Nearest Neighbor – KNN

KNN is the oldest non-parametric classification algorithms [27]. For classifying a new example, the distance is measured from that new example to every other training example. The *k* smallest distances are marked, and the most signified class by this *k* nearest neighbors is defined as the output class label.

The *Euclidean* distance is chosen to compare between the test and training samples sets. Let *C* is a testing set identified as [$C_{1,1}, C_{1,2}, C_{1,3} \dots, C_{1,k}$] and *T* be a training set described as [$T_{1,1}, T_{1,2}, T_{1,3} \dots, T_{1,k}$]. We can compute the *Euclidean* distance between these two sets *C* and *T* as defined by the following equation:

$$D(C, T) = |C - T| \qquad (3)$$

E. Support Vector Machine – SVM

SVM is used for pattern classification [28]. Its base is the idea of maximizing the minimum distance from the separating hyperplane to the nearest example (margins), see Figure 3.

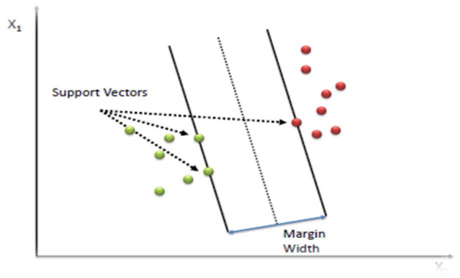


Fig. 3. Support vector lines [28]

The standard SVM supports only binary classification. The extensions have been proposed to handle the case of multiclass classification as well. SVM is applicable for both linearly and nonlinearly separable data. Through an optimization process, the support vectors are produced from the training samples. For SVM training, it is required to find the solution of a large Quadratic Programming (QP) optimization problem. Sequential Minimal Optimization (SMO) can divide this QP problem into small possible QP problems [32].

However, the crucial problem is to find a function that classifies the input features with a minimum error value. The following equation is defining the SVM decision function:

$$F(y) = \sum_{i=0}^{N} \alpha K(x_i, y) + b \qquad (4)$$

where x_i: the support vectors, *y*: the tested feature, α: the weights and *b* is a bias. *K* (x_i, y): the kernel function that performs a mapping into a high-dimensional feature space.

The Experimental results specify that support vector machine can achieve a performance exceeding other classifiers as well requiring less training time to achieve such a result.

IV. Applied Experiments

As mentioned in the preceding sections, this article focuses on applying the five different architectures of CNN on different datasets of the satellite images. The following subsections will address first the description of the used datasets, secondly, sum up the results of the applied experiments, and finally, mention the different platforms on which the experiments were carried out.

A. Remote-Sensing Image Datasets

During the recent, different groups introduced various high-resolution remote sensing image datasets to enable machine learning based research for scene classification. For evaluating different techniques in this field, the authors will review firstly some available sets in this sub-section. The table below, Table 1, shows the number of scene classes, images per each class, total images, size of images, and spatial resolution value for each dataset.

TABLE I. COMPARISON BETWEEN THE DIFFERENT REMOTE-SENSING DATASETS PROPOSED

Dataset	**Scene Classes**	**Images/ Class**	**Total Images**	**Spatial Resolution**	**Image Sizes**
WHU-RS19 [33]	19	~50	1005	up to 0.5	600×600
UC-Merced Land Use [8]	21	100	2100	0.3	256×256
SIRI-WHU [34]	12	200	2400	2	200×200
RSSCN7 [35]	7	400	2800	-	400×400
AID [36]	30	200-400	10000	high	600×600
PatternNet [37]	38	800	30400	up to 0.8	256×256
NWPU-RESISC45 [38]	45	700	31500	~30 - 0.2	256×256

Those datasets are mostly produced by Google® Earth Engine. They cover widespread areas, i.e. parking, runway, agricultural, airplane, baseball diamond, beach, buildings, chaparral, overpass, forest, freeway, harbor, sparse residential, intersection, river, mobile home park, storage tanks, and so on. Figure 4 shows samples of those classes and areas.

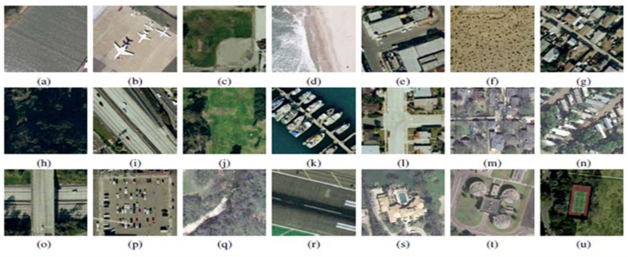


Fig. 4. 21 Representative classes of the UC-Merced Land-Use dataset [39]

Indeed, remote-sensing image quality is affected by the *Spatial Resolution* (SR) measure. SR describes how much details in a photographic image are visible to the human eye. It is usually expressed by *Ground Sample Distance* (GSD). The GSD is defined as the distance between pixel centers measured from air or space on the ground. For instance, in an image with a 1 meter GSD, adjacent pixels image positions are 1 meter apart on the ground. Figure 5 shows the degradation of the image resolution from left to right. The lowest value points to the best quality and vice versa. So, the classification accuracy is affected by the image quality [40].

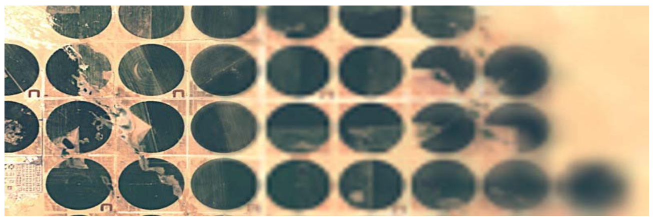


Fig. 5. Image resolution in meters (5-10-15-20-30-60-100-120-250) [40]

B. Experimental Results

To investigate the performance of the selected CNN architectures, namely *AlexNet, VGGNet, GoogleNet, Inception-V3*, and *ResNet-101*, we applied each model on the 7 different datasets listed in Table 1. Also, we tried to vary the classifiers in each experiment to ensure which classifier is the best in our case. Table 3 sums up the accuracy percentage and the sensitivity values of 175 experiments. These experiments contain the aforementioned 5 CNN architectures, with the 5 common classifiers, on the 7 datasets. For instance, the first model, *AlexNet*, with the classifier, *NB*, is applied on the *WHU-RS19* dataset and reached 82.1±0.04% accuracy value. However, the shaded cells illustrate that the best classifier is SVM, so all subsequent measurements will be based on the SVM classifier with the different architectures.

C. Experiments Platforms

The experiments carried out on two different PCs. PC 1 has Intel® Core™ i7-2670QM CPU @ 2.20GHz–8 GB RAM. The other machine is equipped with NVIDIA GTX 1050 4G cc: 6.1 GPU- Intel® Core™ i7-7700HQ @ 2.20GHz, 16 GB RAM. Table 2 shows the elapsed time, in minutes, of the CNN models with SVM classifier on the two different machines.

TABLE II. COMPARISON BETWEEN THE EXECUTION TIME ON CPU (WHITE COLUMNS) AND GPU (YELLOW COLUMNS) IN MINUTES

	WHU-RS19		**UC-Merced**		**SIRI-WHU**		**RSSCN-7**		**AID**		**Pattern Net**		**NWPU-RESISC45**	
AlexNet	14	0.3	30	0.45	34	0.41	37	0.63	93	3	160	5	180	5
VGGNet	240	2	490	4	512	5	560	10	1200	13	1.5 days	95	1.5 days	115
GoogleNet	1.5	0.3	3	0.86	4	0.91	4.5	1	11	1.7	42	8	48	8.5
Inception-V3	5.5	1.6	12	3.3	13	4	15	5	38	12	120	42	180	60
ResNet-101	5	0.8	12	1.8	13	2	16	2.4	38	5.5	45	26	49	27

For instance, the time elapsed on machine 1 for *alexNet* on *WHU-RS19* was 840 seconds and on machine 2 was 16 seconds. The Graphical Processing Unit (GPU) gave an impressive and significant execution time. It optimizes the performance 100 times in comparison to the CPU.

V. DISCUSSIONS

By a quick look, the authors observed that the most powerful architecture was the *ResNet-101*. That is for the reason of keeping the information about the original input, as mentioned in section 2. Figure 6 illustrates this observation. It has 7 blocks for our seven datasets. Each block has 5 bars, *dark blue, red, green, violet*, and *blue*, which indicate to the accuracy percentage of the 5 CNN architectures, *ResNet-101, AlexNet, VGGNet, GoogleNet,* and *Inception-V3, respectively*.

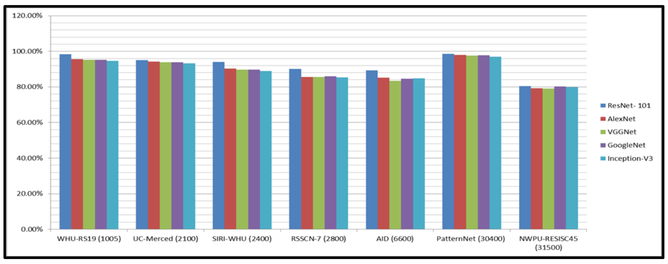


Fig. 6. Illustration of the CNNs accuracy with SVM on the 7 datasets

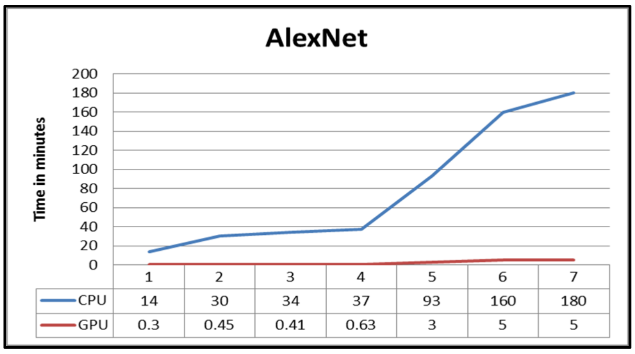
TABLE III. CLASSIFICATION RESULTS OBTAINED BY DIFFERENT CNN MODELS

Model	Dataset	Classifier				
		NB	DT	RF	KNN	SVM
AlexNet	WHU-RS19	82.1± 0.04	82.6± 0.04	90± 0.04	89.9± 0.04	95.7± 0.05
	UC-Merced Land Use	80± 0.03	81± 0.03	86.3± 0.04	88.1± 0.04	94.2± 0.04
	SIRI-WHU	77± 0.06	78± 0.06	83.3± 0.06	82.7± 0.06	90.2± 0.07
	RSSCN-7	71± 0.09	77± 0.06	82.4± 0.11	81.5± 0.11	85.6± 0.11
	AID	68.2± 0.02	72.5± 0.02	72± 0.02	76± 0.02	85.2± 0.02
	PatternNet	88.1± 0.02	92.6± 0.02	93.3± 0.02	95.7± 0.02	98± 0.02
	NWPU-RESISC45	65± 0.11	69.3± 0.01	71.1± 0.01	72± 0.01	79.2± 0.01
VGGNet	WHU-RS19	82± 0.05	80.3± 0.04	88.8± 0.04	90± 0.04	95.2± 0.05
	UC-Merced	79± 0.04	81± 0.03	85.3± 0.04	86± 0.04	93.8± 0.04
	SIRI-WHU	78.5± 0.08	76.5± 0.06	84± 0.06	86± 0.07	86.6± 0.07
	RSSCN-7	73.7± 0.11	72± 0.09	82.5± 0.10	79± 0.10	85.5± 0.11
	AID	70.5± 0.02	70.5± 0.02	71.3± 0.02	73.5± 0.02	83.2± 0.02
	PatternNet	89.3± 0.02	92.5± 0.02	92.8± 0.02	94.4± 0.02	97.5± 0.02
	NWPU-RESISC45	66.4± 0.01	68.7± 0.01	69.3± 0.01	71.4± 0.01	79± 0.01
GoogleNet	WHU-RS19	84.4± 0.04	78.2± 0.04	87.5± 0.04	89.8± 0.04	97.8± 0.05
	UC-Merced	81.6± 0.03	76.6± 0.03	86.3± 0.04	89± 0.04	93.8± 0.04
	SIRI-WHU	76.8± 0.06	75± 0.06	82± 0.06	80± 0.06	89.3± 0.07
	RSSCN-7	76.3± 0.10	73.2± 0.10	82.6± 0.10	79.6± 0.11	86± 0.11
	AID	76.7± 0.02	71.8± 0.02	73.3± 0.02	72.6± 0.02	84.6± 0.02
	PatternNet	88.2± 0.02	91.7± 0.02	90.1± 0.02	94± 0.02	97.7± 0.02
	NWPU-RESISC45	69± 0.01	70± 0.01	71.3± 0.01	70.8± 0.01	80.2± 0.01
Inception-V3	WHU-RS19	87.8± 0.04	75.5± 0.03	84.6± 0.04	88.2± 0.04	94.7± 0.05
	UC-Merced	86± 0.04	73.1± 0.03	83.3± 0.04	88.3± 0.04	91.1± 0.04
	SIRI-WHU	77.9± 0.06	69± 0.05	83± 0.06	79.1± 0.06	89± 0.06
	RSSCN-7	73.8± 0.10	69± 0.09	82.2± 0.10	79± 0.10	87± 0.10
	AID	74± 0.02	68.2± 0.02	69.7± 0.02	67.6± 0.02	84.7± 0.02
	PatternNet	90.3± 0.02	88± 0.02	91.5± 0.02	94.6± 0.02	97± 0.02
	NWPU-RESISC45	69.5± 0.01	67.4± 0.01	67.2± 0.01	71.5± 0.01	79.8± 0.01
ResNet-101	WHU-RS19	91.1± 0.04	84.2± 0.04	88.7± 0.04	89.5± 0.04	98.4± 0.05
	UC-Merced	82.8± 0.03	81.4± 0.03	85.4± 0.04	87.3± 0.04	95.4± 0.04
	SIRI-WHU	82.5± 0.06	77± 0.06	84.7± 0.06	84± 0.06	94± 0.07
	RSSCN-7	77.7± 0.10	73.5± 0.09	83.3± 0.10	79.8± 0.11	90.1± 0.12
	AID	76.8± 0.02	71.6± 0.02	74.4± 0.02	75.7± 0.02	89.3± 0.02
	PatternNet	90± 0.02	89.9± 0.02	94.1± 0.02	95.7± 0.02	98.6± 0.02
	NWPU-RESISC45	73.3± 0.01	70.5± 0.01	73.5± 0.01	73± 0.01	80.5± 0.01

The dark blue bar has the max height for each dataset which indicates to the *ResNet-101* model. As mentioned in section IV, the low value of spatial resolution points to better quality. So, if we are comparing the accuracy of *patternNet* and *NWPU-RESISC45* datasets, they have approximately the same number of images and the same image size, but obviously, the accuracy of all networks on *patternNet* exceeds by far the accuracy of *NWPU-RESISC45* (refer to the chart above). The reason is the spatial resolution of each one. For *patternNet*, the max spatial resolution is 0.8 while for *NWPU-RESISC45*, it is up to 30 m.

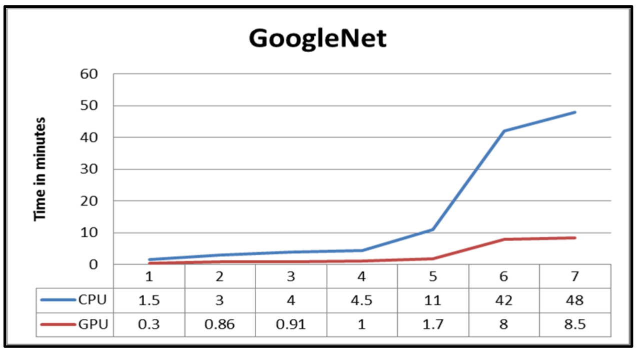
Also, the (BRISQUE), Blind/Referenceless Image Spatial Quality Evaluator, no-reference image quality factor was measured for those both datasets. The undistorted image has the best perceptual and thus the lowest BRISQUE value. To test that, a river-class for instance in each dataset was selected and BRISQUE score measured. For *patternNet*, it was 22.748, and for *NWPU-RESISC45* was 35.319.

Backing to the CNN architectures, the experiments demonstrated that *AlexNet* is the fastest one ran on GPU and *GoogleNet* model is the fastest one ran on CPU as shown in Figure 7 and 8 respectively. The red line shows the execution time on GPU and the blue line for CPU. In *AlexNet* graph, the red line is almost stable even with large datasets, while the blue line increased sharply. In *GoogleNet* graph, both the red and blue lines are slightly equal with small datasets and that is a result of the parallel behavior of the model itself, the stages of the network is running in parallel. However, the slowest model was *VggNet*. It takes more than 1 day for testing only one image of large datasets.



	1	2	3	4	5	6	7
CPU	14	30	34	37	93	160	180
GPU	0.3	0.45	0.41	0.63	3	5	5

Fig. 7. The elapsed time (in min) of *AlexNet* on our seven datasets. The red line shows the execution time on GPU and blue line for CPU.



	1	2	3	4	5	6	7
CPU	1.5	3	4	4.5	11	42	48
GPU	0.3	0.86	0.91	1	1.7	8	8.5

Fig. 8. The elapsed time (in min) of *GoogleNet* on our seven datasets. The red line shows the execution time on GPU and blue line for CPU.

VI. Conclusion

In this work, the concept of deep learning through CNN model was applied for remote-sensing images classification. The article firstly reviewed the description of CNN architecture and its common models; namely, *AlexNet, VGGNet, GoogleNet, Inception-V3,* and *ResNet-101*. As well, the article gave a quick review for the classifiers used in our experiments; namely, *Naïve Bayes, Decision Tree, Random Forest, KNN,* and *SVM*. Also, the 7 different publicly remote-sensing datasets were presented in details for the evaluation and comparisons.

The article applied the 5 models of CNNs in a combination of the 5 mentioned classifiers on the reviewed datasets. The best CNN model was the *ResNet-101*, with *SVM* classifier. It achieved 98.6% classification accuracy for high-resolution datasets, i.e. *PatternNet* dataset with high spatial resolution. The experiments were carried out on two platforms: CPU & GPU. Thanks to GPU for giving an impressive and significant execution time. The parallel computing optimizes the performance 100 times compared to the serial computations.

The researchers are devoted to learning better features for the scene classification tasks by applying the appropriate deep learning methods. And we are recommending applying CNNs models whereas its different architectures are accurate and robust. They are almost equal in the performance and accuracy to some extent. The classification accuracy is affected by the input image quality and the performance is affected by the high specification of the physical platform.

References

[1] L. Zhang, G. Xia, T. Wu, L. Lin, and X. Tai, "Deep Learning for Remote Sensing Image Understanding," Journal of Sensors, vol. 2016, pp.1-2, 2016.

[2] G. Hinton, and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," Science, vol. 313, pp.504-507, 2006.

[3] G. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," Neural Computing, vol. 18, pp.1527-1554, 2006.

[4] R. Salakhutdinov, and G. Hinton, "An efficient learning procedure for deep Boltzmann machines," Neural Comp., vol. 24, pp.1967-2006, 2012.

[5] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," Machine Learning Res., vol. 11, pp.3371-3408, 2010.

[6] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," Int. Conf. Learn. Represent., pp.1-16, 2014.

[7] I. Ševo, and A. Avramović, "Convolutional Neural Network Based Automatic Object Detection on Aerial Images," IEEE Geosci. Remote Sens. Lett., vol. 13, pp. 740-744, 2016.

[8] UC-Merceed Land Use Dataset, http://weegee.vision.ucmerced.edu/datasets/landuse.html.

[9] F. Luus, B. Salmon, F. Van Den Bergh, and B. Maharaj, "Multiview deep learning for land-use classification," IEEE Geosci. Remote Sens. Lett., vol. 12, pp. 2448-2452, 2015.

[10] Y. Zhong, F. Fei, and L. Zhang, "Large patch convolutional neural networks for the scene classification of high spatial resolution imagery," Applied Remote Sens., vol. 10, pp. 025006-025006, 2016.

[11] M. Längkvist, A. Kiselev, M. Alirezaie, and A. Loutfi, "Classification and Segmentation of Satellite Orthoimagery Using Convolutional Neural Networks," Remote Sensing, vol. 8, pp. 1-21, 2016.

[12] K. Nogueira, O. A. Penatti, and J. A. d. Santos, "Towards Better Exploiting Convolutional Neural Networks for Remote Sensing Scene Classification," Pattern Recognition, vol. 61, pp. 539-556, 2017.

[13] D. Marmanis, M. Datcu, T. Esch, and U. Stilla, "Deep Learning Earth Observation Classification Using ImageNet Pretrained Networks," IEEE Geosci. Remote Sens. Lett, vol. 13, pp.105-109, 2015.

[14] C. Jingbo, W. Chengyi, M. Zhong, C. Jiansheng, H. Dongxu, and A. Stephen, "Remote Sensing Scene Classification Based on Convolutional Neural Networks Pre-Trained Using Attention-Guided Sparse Filters," Remote Sensing, vol. 10, pp.1-16, 2018.

[15] L. Zhang, and B. Du, "Deep Learning for Remote Sensing Data: A Technical Tutorial on the State of the Art," IEEE Geoscience and Remote Sensing Magazine, vol. 4, pp. 22-40, 2016.

[16] White Paper, FPGA Acceleration of Convolutional Neural Networks. Molex Company: Nallatech, 2015.

[17] T. Guo, J. Dong, H. Li and Y. Gao, "Simple convolutional neural network on image classification," IEEE 2nd International Conference on Big Data Analysis (ICBDA). Beijing, pp. 721-724, 2017.

[18] X. Han, Y. Zhong, L. Cao, and L. Zhang, "Pre-Trained AlexNet Architecture with Pyramid Pooling and Supervision for High Spatial Resolution Remote Sensing Image Scene Classification," Journal of Remote Sensors, vol. 9, pp.1-22, 2017.

[19] K. Simonyan, and A. Zisserman, "Very Deep Convolutional Networks for Large Scale Image Recognition," CoRR, vol. 1409.1556, pp.1-14, 2014.

[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," CVF, vol. 2015, pp.1-9, 2015.

[21] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," CVF, vol. 2016, pp.2818-2826, 2016.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," CVF, vol. 2016, pp.770-778, 2016.

[23] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei, "ImageNet: A Large Scale Hierarchical Image Database," in CVPR09, pp. 248-255, 2009.

[24] G. Kaur, and E.N. Oberai, "A review article on Naive Bayes classifier with various smoothing techniques," International Journal of Computer Science and Mobile Computing, vol. 3, pp. 864–868, 2014.

[25] S. B. Kotsiantis, "Decision trees: a recent overview," Artificial Intelligence Review, vol. 39, pp. 261, 2013.

[26] A. M. Prasad, L. R. Iverson, and A. Liaw, "Newer classification and regression tree techniques: Bagging and random forests for ecological prediction," Ecosystems, vol. 9, pp.181–199, 2006.

[27] T. Araújo, N. Nunes, H.Gamboa, and A. Fred, "Generic biometry algorithm based on signal morphology information: Application in the electrocardiogram signal," in Pattern Recognition Applications and Methods. Cham: Springer, 2015, pp. 301–310.

[28] V. Vapnik, The Nature of Statistical Learning Theory. Verlag, New York: Springer Science & Business Media, 2013.

[29] A. M. Ahmed, A. Rizaner, and A. H. Ulusoy, "A novel decision tree classification based on post-pruning with Bayes minimum risk,". PLoS ONE, vol. 13, 2018.

[30] K. Machová, F. Barčák, and P.Bednár, "A Bagging Method using Decision Trees in the Role of Base Classifiers," Acta Polytechnica Hungarica, vol. 3, pp. 121-132, 2006.

[31] L. Breiman, Random Forests. Berkeley, CA 94720: University of California, 2001.

[32] J. C. Platt, Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. MSR-TR-98-14: Microsoft Research, 1998.

[33] WHU-RS19 Dataset, https://www.google.com/url?q=http%3A%2F%2Fwww.xinhua-fluid.com%2Fpeople%2Fyangwen%2FWHU-RS19.html&sa=D&sntz=1&usg=AFQjCNFzrOnViW6TWOoFbN1IaIMfyLdJhQ.

[34] SIRI-WHU Dataset, http://www.lmars.whu.edu.cn/prof_web/zhongyanfei/e-code.html.

[35] RSSCN7 Dataset, https://www.dropbox.com/s/j80iv1a0mvhonsa/RSSCN7.zip?dl=0.

[36] AID Dataset, http://www.lmars.whu.edu.cn/xia/AID-project.html.

[37] PatternNet Dataset, https://sites.google.com/view/zhouwx/dataset?authuser=0.

[38] NWPU-RESISC45 Dataset, https://www.google.com/url?q=http%3A%2F%2Fwww.escience.cn%2Fpeople%2FJunweiHan%2FNWPU-RESISC45.html&sa=D&sntz=1&usg=AFQjCNGs2uMeX7KT2QvEMzcD5uF4-aQChw.

[39] M. Castelluccio, G. Poggi, C. Sansone, and L.Verdoliva, “Land Use Classification in Remote Sensing Images by Convolutional Neural Networks,” CoRR, vol. abs/ 1508.00092, pp. 1-11, 2015.

[40] A. Orych, “Review of Methods for Determining the Spatial Resolution of Uav Sensors,” International Conference on Unmanned Aerial Vehicles in Geomatics. Canada, vol. XL-1/W4, pp. 391-395, 2015.

[41] M. A. Shafaey, M. A.-M. Salem, H. M. Ebied, M. N Al-Berry, and M. F. Tolba, “Deep Learning for Satellite Image Classification,” International Conference on Advanced Intelligent Systems and Informatics. Egypt, vol. 845, pp. 383-391, 2018.