# NU-LiteNet: Mobile Landmark Recognition using Convolutional Neural Networks

**Chakkrit Termritthikun**[1] and **Paisarn Muneesawang**[2]

**ABSTRACT**

The growth of high-performance mobile devices has resulted in more research into on-device image recognition. The research problems have been the latency and accuracy of automatic recognition, which remain as obstacles to its real-world usage. Although the recently developed deep neural networks can achieve accuracy comparable to that of a human user, some of them are still too slow. This paper describes the development of the architecture of a new convolutional neural network model, NU-LiteNet. For this, SqueezeNet was developed to reduce the model size to a degree suitable for smartphones. The model size of NU-LiteNet was therefore 2.6 times smaller than that of SqueezeNet. The model outperformed other Convolutional Neural Network (CNN) models for mobile devices (eg. SqueezeNet and MobileNet) with an accuracy of 81.15% and 69.58% on Singapore and Paris landmark datasets respectively. The shortest execution time of 0.7 seconds per image was recorded with NU-LiteNet on mobile phones.

**Keywords**: Deep Learning, Landmark Recognition, Convolutional Neural Networks, NU-LiteNet

## 1. INTRODUCTION

Landmark recognition is an important feature for tourists who visit important places. A tourist can use a smartphone that has a landmark-recognition application installed for retrieval of information about a place, such as the names of landmarks, the history, events that are currently taking place and opening times of shows. This process involves taking a picture of a landmark and letting the application software retrieve the relevant information. This effective mobile interface has created new trends for the tourist industry, mobile shopping, and other e-commerce applications.

In the past, landmark recognition [1–7] utilized the capability of computers. These computing devices can cope with the large size of databases and computational complexity with sufficient resources to operate the application. However, major problems have occurred due to inaccuracy of recognition and long processing times when these applications have been running on mobile devices. These may have been caused by the utilization of recognition methods, such as the scale invariant feature transform (SIFT), scalable vocabulary tree (SVT) and geometric verification (GV). Some of these methods have been studied extensively in the past because of their exceptional performance. However, their high degree of accuracy results in long processing times.

The application of machine learning models for landmark recognition has encountered various problems in practice. Landmark recognition needs a large amount of training with a dataset to obtain an effective machine learning model. This model is then utilized by the recognition program. The size of the model obtained is usually large, and thus requires a long processing time. The image processing and recognition are therefore usually done on a server computer. The picture is taken by the smartphone user and is sent to the server for recognition, after which the result is sent to the smartphone. Moreover, the smartphone has to be connected to the internet to perform the recognition function, so it cannot be performed in off-line mode. To solve this problem, the application needs to embed the machine learning model into the smartphone and perform on-device recognition. However, a large model cannot fit into the smartphone because of the phone's limited memory space, and so its size has to be reduced. One method for doing so is the application of a convolutional neural network (CNN). This has been recently studied with a view to extending the CPU and GPU modules to achieve high-performance image recognition.

CNN has received much attention for image recognition, object detection and image description. For the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), new models have been developed which are more effective than the previous models. Such models are AlexNet [8], GoogLeNet [9], VGG [10], and ResNet [11], which were the winners of ILSVRC in 2012–2015. These competitions have stimulated progress in the development of research on image recognition, and the CNN models are the most effective examples of machine learning at present.

As described in [12], AlexNet [8], the winner of ILSVRC 2012, was applied to a large-scale social image collection (500 classes of 2 million images) and compared with the Bag-of-Word (BoW) method us-

Manuscript received on January 5, 2019 ; revised on May 21, 2019.

Final manuscript received on June 30, 2019.

[1,2] The authors are with Department of Electrical and Computer Engineering, Faculty of Engineering, Naresuan University, Phitsanulok 65000 Thailand., E-mail: chakkritt60@email.nu.ac.th and paisarnmu@nu.ac.th

ing a SIFT descriptor. It was shown that CNN could attain 23.88% recognition accuracy, while the BoW method only reached 9.5%. This result indicated that CNN was more effective for image recognition than the BoW methods. As discussed in [13], the parameters in AlexNet were reduced by about 50 times to create SqueezeNet, which resulted in a "lite" version of CNN. The structure of SqueezeNet contains two parts: (1) A Squeeze block, which implements the convolution layer with a $1 \times 1$ filter, and (2) an Expand block, which implements the convolution layer with $1 \times 1$ and $3 \times 3$ filters. The Squeeze block reduces the data dimension, while the Expand block is effective in analyzing data. The reduced-size version of CNN can still maintain the same level of recognition accuracy as AlexNet.

GoogLeNet was developed by Google and was the winner of ILSVRC in 2014. A defining feature of GoogLeNet is its inception module, which has the ability to analyze data accurately. The network consists of a convolution layer with $1 \times 1$, $3 \times 3$, and $5 \times 5$ filters. It uses a convolution layer with a $1 \times 1$ filter to reduce the data dimensions. GoogLeNet can reduce a model size by up to 4.8 times more than AlexNet. The architecture of GoogLeNet includes nine inception modules arranged in a cascading manner, which increases performance in terms of recognition accuracy. However, this structure also increases the time required to train the network to about three times that of AlexNet.

For this paper, we adopted an idea for the development of SqueezeNet, which consisted of a Squeeze block and Expand block. The improvement consisted of the inclusion of a convolution layer with $5 \times 5$ and $7 \times 7$ filters to enable the Expand block to cope with the analysis of complex image content. It was also proposed to use the Squeeze block in order to reduce data dimensions. The newly proposed network, NU-LiteNet, achieved high recognition accuracy as well as reduced processing time, by using CNN models of a reduced size. This made on-device processing possible, particularly for implementing an offline landmark recognition facility on smartphones.

## 2. CONVOLUTIONAL NEURAL NETWORK

The convolutional neural network (CNN) has a structure which is the same as that of a normal neural network. It is classified as a feed-forward neural network, which consists of a convolution layer, a pooling layer and a fully connected layer. At least three of these layers are stacked on a network to learn and classify data. These layers, as well as the input layer, are placed in the following order: input, convolution, pooling, and fully connected.

The input layer is a layer that contains an image dataset for training and testing. The image data is in an RGB color space and the image size depends on the selected network model. For example, a network model that utilizes an image width of 256 pixels and a height of 256 pixels will have data for one image at $[256 \times 256 \times 3]$, where 3 is the number of color channels.

The convolution layer is the layer that operates the multiplication of each pixel with filter coefficients. This operation starts at the location (0,0) of the data and moves by one pixel (stride 1) each time from left to right and from top to bottom until all of the pixels are covered. This process will result in the creation of an activation map. For example, given that the size of the image data is $[224 \times 224 \times 3]$ and there are a total of 96 filters, each of which has a size of $3 \times 3$, the resulting activation map will be $[111 \times 111 \times 3]$ when the filter moves by two pixels (stride 2) each time.

The pooling layer comes after the convolution layer. Its main function is to reduce the dimensions of the data representation, which will reduce the number of parameters and calculations in the next layer. The max pooling is the function that performs this task. For example, in order to reduce data of size $[111 \times 111 \times 3]$ to half that size (i.e., $[55 \times 55 \times 3]$), a filter of size $3 \times 3$ and stride 2 are needed.

The last layer is the fully connected layer. Its main function is to convert the output data to one dimension. CNN can be developed to learn a dataset by increasing the number of hidden layers in order to increase leaning capability. The network divides image data into sub-images, each of which is analyzed for features such as color, shape, and texture. These features are used for image classification prediction patterns.

## 3. NU-LITENET

This section presents the development of two types of network architecture for CNNs: NU-LiteNet-A and NU-LiteNet-B.

### 3.1 Added $5 \times 5$ and $7 \times 7$ Convolution

Considering the Expand block of SqueezeNet [13], SqueezeNet chooses the use of small filters, such as $1 \times 1$ and $3 \times 3$ convolution, to detect smaller objects. Another reason for using a small filter comes from the design aspects of the model, because the size of the

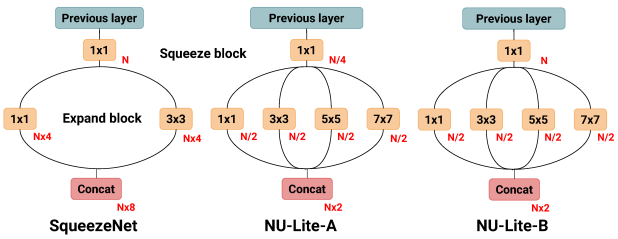


***Fig.1:*** *Squeeze block and Expand block of NU-LiteNet-A and NU-LiteNet-B compared with SqueezeNet*

parameters is small and the processing time is minimal. As a result of this, SqueezeNet is not as accurate as GoogLeNet [9], but it is as accurate as AlexNet [8]. In this paper, we chose to add large convolution filters, such as $5 \times 5$ and $7 \times 7$, to the Expand blocks in order to enhance accuracy, just like those used for the Inception module of the GoogLeNet [9]. The use of a large filter to detect objects is similar to a small filter, but the difference is that the large filter helps to identify or confirm the central position of the object. When the data from the small filter and large filter are concatenated, the model can confirm the position of the desired object as shown in [14–17]. For this reason, this model has greater accuracy. However, an increase in the number of filters resulting from adding the large $5 \times 5$ and $7 \times 7$ filter convolution expand blocks caused an increase in processing time and in the number of parameters. Therefore, due to the large filter expand blocks, there was a need to reduce the size and depth of the model of SqueezeNet. This was required to reduce the processing time and the number of appropriately sized parameters to allow the applications to be properly processed on smartphones.

### 3.2 NU-LiteNet-A

The NU-LiteNet-A was developed by modifying SqueezeNet, which had the Squeeze and Expand blocks shown in Fig. 1 (left). It introduced $5 \times 5$ convolution and $7 \times 7$ convolution into the Expand block, as shown in Fig. 1 (center). If N was the number of channels (depth) of the previous layer, NU-LiteNet-A reduced N in the $1 \times 1$ convolution or Squeeze block by one fourth (i.e., $\frac{N}{4}$) of the previous layer. This increased N in the expand block by 2 times that of N in the Squeeze block (i.e., $\frac{N}{2}$) in every convolution layer. As a result, the number of channels at the end of the Expand block was doubled (i.e., $N \times 2$) compared to that of the previous layer. The details of NU-LiteNet-A are summarized in Table 1.

### 3.3 NU-LiteNet-B

The NU-LiteNet-B changed the structure of NU-LiteNet-A by changing the amount of depth, N, of the Squeeze block to be the same as that of the previous layer. This corresponded to the structure of SqueezeNet, as shown in Fig. 1 (right). In this structure, the Expand block received an amount of depth, N, equal to that of the previous layer. This increased the effectiveness of the network for data analysis, but also increased the number of parameters and thus required longer processing time. The details of NU-LiteNet-B are summarized in Table 1.

### 3.4 Completed Network structures

The complete architectures of NU-LiteNet-A and NU-LiteNet-B are shown in Fig. 2. This research proposed to cut the number of layers and include an Expand block. NU-LiteNet-A and NU-LiteNet-B had only two modules each, and the number of channels (depth) was $N = 256$ channels. This was because the experimental data (shown in Section 4) had only 50 classes. If the amount of depth was increased, the network would have had a large number of parameters and required longer processing time. Therefore, the design of the network considered the number of parameters and the processing time that could be applied effectively on smartphones.

***Table 1:*** *NU-LiteNet-A and NU-LiteNet-B*

| Layer | Output | NU-Lite-A | NU-Lite-B |
|---|---|---|---|
| Input | 224×224 | - | |
| Convolution 1 | 113×113 | 5×5, 64, stride 2, pad 3 | |
| Pooling 1 | 56×56 | max pool, 3×3, stride 2 | |
| Convolution 2 | 56×56 | 1×1, 64, stride 2 | |
| Convolution 3 | 56×56 | 3×3, 64, stride 1, pad 1 | |
| Pooling 2 | 28×28 | max pool, 3×3, stride 2 | |
| NU-Lite-Block 1 | 28×28 | [Block-A], 128 | [Block-B], 128 |
| Pooling 3 | 14×14 | max pool, 3×3, stride 2 | |
| NU-Lite-Block 2 | 14×14 | [Block-A], 256 | [Block-B], 256 |
| Pooling 4 | 1×1 | average pool | |
| Fully connected | 50 | softmax | |

This design was suitable for smartphone processing. The aim was to obtain a network of high effectiveness that worked as well as other state-of-the-art CNN models while keeping the processing time to a minimum. In Fig. 2, GoogLeNet is shown in comparison with the proposed network architecture. GoogLeNet had nine modules, whereas the proposed network had only two modules, which reduced processing time and model size.

## 4. EXPERIMENTAL RESULT

In the experiment, we trained the networks with a high-performance computing (HPC) unit. It had the following specifications: Intel(R) Xeon(R) E5-2683 v3 @ 2.00GHz 56 Core CPU, 64 GB RAM, and NVIDIA Tesla K80 GPU. The operating system was Ubuntu Server 14.04.5. For testing, we used a smartphone with the following specifications: Samsung Exynos Octa 7580 @ 1.6 GHz 8 Core CPU and 3 GB RAM,

***Fig.3a:*** *Singapore landmarks*

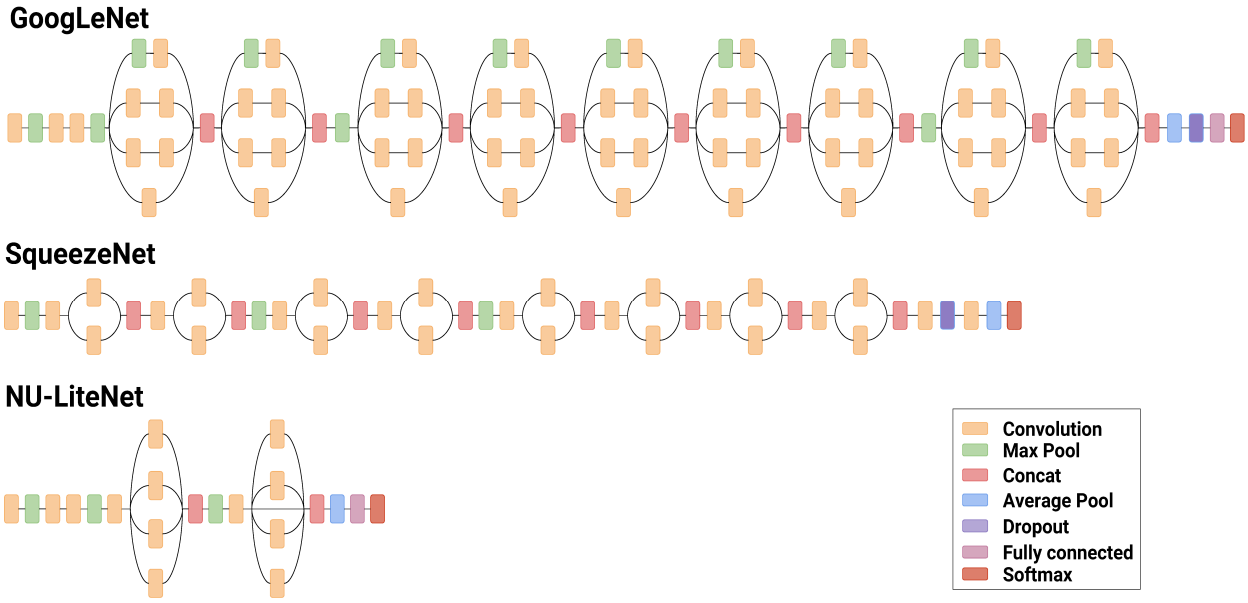


***Fig.2:*** *Architecture of NU-LiteNet.*

with Android 6.0.1 as the operating system.

### 4.1 Databases

The experimental data was obtained from two standard landmark datasets. The first set was from Singapore landmarks [2] and consisted of 50 landmarks that were popular with tourists (4,060 images), some of which are shown in Fig. 3a. The second dataset was the Paris dataset from Paris, France [18], which consisted of 12 landmarks (6,412 images), some of which are shown in Fig. 3b. For each dataset, images were divided into a training set and testing set, at 90% and 10% respectively. The images were resized to 256 × 256 pixels.

### 4.2 Comparison of NU-LiteNet and other models

In the experiment, all network models, including AlexNet, GoogLeNet, SqueezeNet, MobileNet [19], NU-LiteNet-A, and NU-LiteNet-B, were trained from scratch. The Singapore and Paris landmarks datasets were used, and each set was divided into two parts: a training set (90%) and a testing set (10%), with 10-fold cross-validation. The hyperparameters for NU-LiteNet-A and NU-LiteNet-B were as follows. Solver: Stochastic Gradient Descent (SGD) [20]; Momentum: 0.9; Mini-batch size: 128; Learning rate: 0.1; Weight decay: 0.0005; Epoch size: 100.

For the training process, we measured the parameters of the networks. The number of parameters indicated the model size. For the testing process, we measured the accuracy using 10-fold cross-validation. The accuracy was measured in terms of the top-1 accuracy as well as the top-5 accuracy.

Table 2 shows the experimental result obtained by 10-fold cross-validation for the Singapore landmark dataset. It can be observed from the result that both versions of NU-LiteNet were more effective for landmark recognition at top-1 accuracy as well as top-5 accuracy than AlexNet, GoogLeNet, SqueezeNet, and MobileNet. The accuracy was higher than that of GoogLeNet by up to 7.4-10.46%. For the number of parameters, it was discovered that NU-LiteNet-A had the lowest number of parameters: 0.28M. This

***Fig.3b:*** *Paris landmarks*

***Table 2:*** *Recognition accuracy obtained by 10-fold cross-validation. NU-LiteNet is compared with other models, using the Singapore dataset.*

| Model | Params (M) | top-1 (%) | top-5 (%) |
|---|---|---|---|
| AlexNet [8] | 62.37 | 64.82 | 84.94 |
| GoogLeNet [9] | 6.02 | 70.69 | 88.75 |
| SqueezeNet [13] | 0.75 | 60.08 | 83.24 |
| MobileNet [19] | 3.21 | 79.26 | 93.58 |
| NU-LiteNet-A | 0.28 | 78.09 | 92.75 |
| NU-LiteNet-B | 0.94 | 81.15 | 93.96 |

***Table 3:*** *Recognition accuracy obtained by 10-fold cross-validation. NU-LiteNet is compared with other models, using the Paris dataset.*

| Model | Params (M) | top-1 (%) | top-5 (%) |
|---|---|---|---|
| AlexNet [8] | 62.36 | 58.62 | 90.00 |
| GoogLeNet [9] | 6.01 | 59.97 | 91.10 |
| SqueezeNet [13] | 0.74 | 53.34 | 87.97 |
| MobileNet [19] | 3.20 | 64.53 | 94.06 |
| NU-LiteNet-A | 0.27 | 66.67 | 94.07 |
| NU-LiteNet-B | 0.93 | 69.58 | 94.65 |

***Table 4:*** *Execution time and model size obtained by recognition on smartphone.*

| Model | Image size (pixels) | Execution time (ms/image) | Size (MB) |
|---|---|---|---|
| AlexNet [8] | 1620 × 1080 | 1038 | 217 |
| GoogLeNet [9] | 1620 × 1080 | 1244 | 23 |
| SqueezeNet [13] | 1620 × 1080 | 773 | 2.86 |
| MobileNet [19] | 1620 × 1080 | 1053 | 16.2 |
| NU-LiteNet-A | 1620 × 1080 | 637 | 1.07 |
| NU-LiteNet-B | 1620 × 1080 | 706 | 3.6 |

was 2.5 times lower than that of SqueezeNet.

The experiment results from the Paris dataset showed similar trends to those of the Singapore dataset in terms of recognition accuracy. Both versions of NU-LiteNet gave higher accuracy than the other models. The accuracy was higher than that of GoogLeNet by up to 6.7-9.61%, as shown in Table 3.

From Table 2 and Table 3, it can be observed that NU-LiteNet-A used the lowest number of parameters. NU-LiteNet-B provided the highest accuracy, while the number of parameters obtained was about three times higher than that of NU-LiteNet-A.

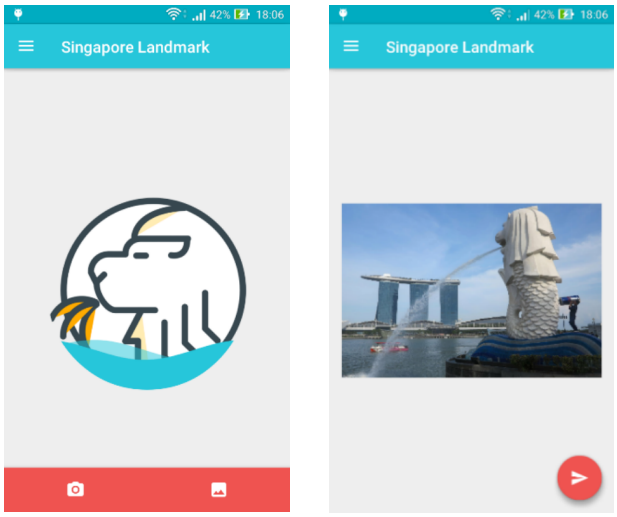


***Fig.4:*** *Snapshots from the landmark-recognition program on a smartphone with Android: (left) the first page, and (right) the query image taken by the device.*

### 4.3 Application for Landmark Recognition on Android

For the development of an application on smartphones using Android, the trained models were utilized for landmark recognition. The processing time and model size (the space required to store the model on a smartphone) were measured. Table 4 shows the result for the processing of an input image of size 1620 × 1080 pixels. The top three models that required the lowest processing time were NU-LiteNet-A (637 ms), NU-LiteNet-B (706 ms), and SqueezeNet (773 ms). The top three models that had the smallest model size were NU-LiteNet-A (1.07 MB), SqueezeNet (2.86 MB), and NU-LiteNet-B (3.6 MB). From this result, it can be observed that NU-LiteNet-A was the most effective model in terms of processing time as well as model size at 637 ms per image and 1.07 MB respectively.

Fig. 4 and 5 show snapshots of the application of mobile landmark recognition on a smartphone. The recognition function can be used in the offline mode, in which the on-device recognition module is implemented. The user can take a picture and start the process of recognition of the landmark using the phone. The retrieved data are the name and probability score of the predicted landmark class. There are also menus for history and events that can be used to retrieve the complete information about the landmark from the web (Wikipedia) if the phone is connected to the internet. The event menu shows the information about the event currently shown in the actual area around the landmark. This information can be used to advertise the landmark to tourists.

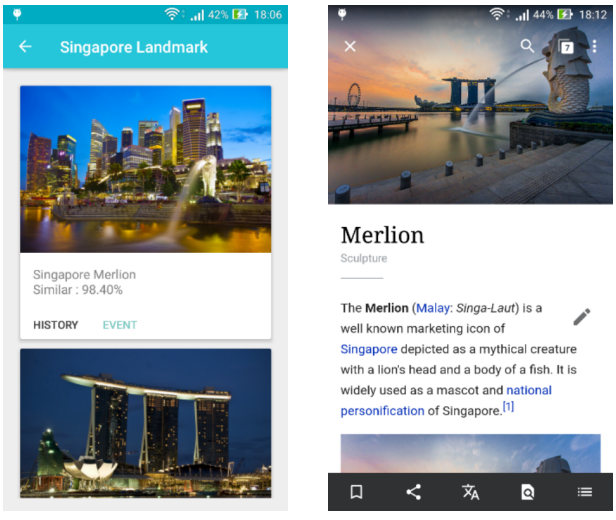


***Fig.5:*** *Snapshots from the landmark-recognition program on a smartphone with Android: (left) the recognition result showing the landmarks with the highest similarity scores in deceasing order, and (right) the information about the landmarks from Wikipedia.*

## 5. CONCLUSIONS

This paper presents NU-LiteNet, which adopted the development idea of SqueezeNet to improve the network structure of the convolutional neural network (CNN). It aimed to reduce the model size to a degree suitable for on-device processing on a smartphone. The two versions of the proposed network were tested on Singapore and Paris landmarks datasets, and it was determined that NU-LiteNet could reduce the model size by 2.6 times compared with SqueezeNet, and it improved recognition performance. The execution time of NU-LiteNet on a smartphone was also shorter than that of other CNN models. In future work, we will continue to improve accuracy and reduce model size for large-scale image databases, such as ImageNet, and country-scale landmark databases.

## APPENDICES

## A IMPLEMENTATION DETAILS

The data collected in the Singapore and Paris landmarks datasets were divided into two parts: training data and testing data. The training data for the two sets was 256 × 256 pixels. Data augmentation was done using the random crop image size of 224 × 224 pixels in a horizontal flip to increase the number of images in the dataset. An improvement to enhance the accuracy of neural networks with greater precision was developed in [21] by adding Batch Normalization after Convolutions of all layers, as well as in [11, 22] to allow much higher learning rates. A problem with the Internal covariate shift of [23] occurred during the data training in lower hidden layers. For the activation function, the Rectified Linear Unit [24, 25] (ReLU) was used after the convolution layers in NU-LiteNet-A and NU-LiteNet-B.

Looking at the performance of top-1 accuracy of AlexNet, GoogLeNet, SqueezeNet, MobileNet and both versions of NU-LiteNet, the training of the Singapore landmarks from epoch 1-100 was as shown in Fig. 6. Considering the accuracy of 60%, it was observed that NU-LiteNet-B converged first at epoch 10, followed by NU-LiteNet-A at epoch 15, then GoogLeNet at epoch 29, AlexNet at epoch 34, SqueezeNet at epoch 91 and finally MobileNet at epoch 21. Considering the epoch 1-25 at a learning rate of 0.1 (LR = 0.1), it was observed that both versions of NU-LiteNet converged better than the AlexNet, GoogLeNet, SqueezeNet and MobileNet models until epoch 26 at a LR = 0.01. NU-LiteNet-B had higher top-1 accuracy than the other models during the training with the highest accuracy of 81.15% with the Singapore landmarks dataset.

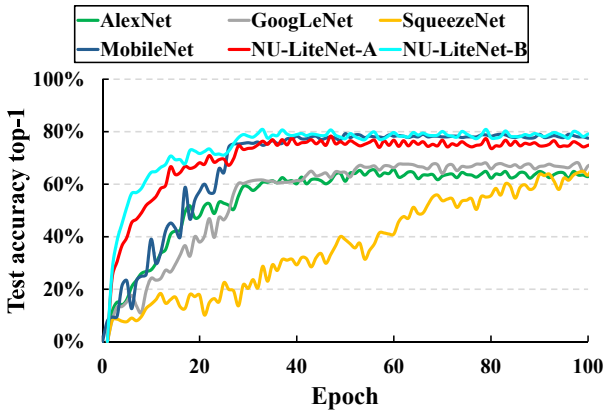


***Fig.6:*** *Top-1 accuracy vs. number of epochs for Singapore landmarks.*

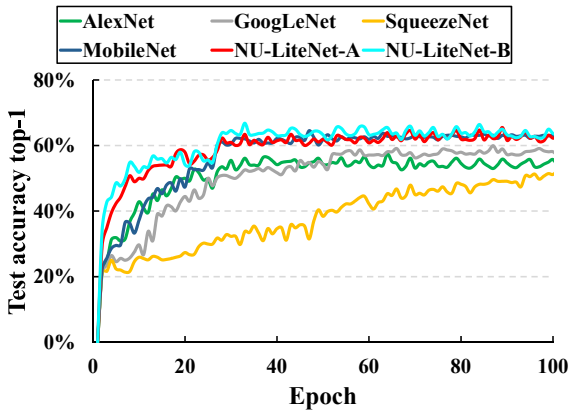


***Fig.7:*** *Top-1 accuracy vs. number of epochs for Paris landmarks.*

The top-1 accuracy of AlexNet, GoogLeNet, SqueezeNet, MobileNet and the two versions of the NU-LiteNet training data set with Paris landmarks are shown in Fig. 7. Considering the accuracy of 60%, it was observed that the NU-LiteNet-B and MobileNet models converged at epoch 28, followed by NU-LiteNet-A at epoch 29, but the AlexNet, GoogLeNet and SqueezeNet models did not converge. Accuracy was up to 60% for the AlexNet model convergence which is capped at 58.62%, followed by GoogLeNet at 59.97%, and 53.34% for SqueezeNet. The highest model top1-accuracy for Paris landmarks was recorded in NU-LiteNet-B with 69.58%, followed by NU-LiteNet-A and MobileNet with 66.67% and 64.53% respectively.

## ACKNOWLEDGMENTS

The authors would like to acknowledge the financial support received from the Thailand Research Fund through the Royal Golden Jubilee Ph.D. Program (Grant No. PHD/0101/2559) and further, we would like to extend our appreciation to Mr. Thomas Elliott of the Naresuan University Graduate School for his assistance in editing the English grammar and expression in the paper.

## References


[1] Kiatchai Banlupholsakul, Jirarat Ieamsaard, and Paisarn Muneesawang. Re-ranking approach to mobile landmark recognition. In *Computer Science and Engineering Conference (ICSEC), 2014 International*, pages 251–254. IEEE, 2014.

[2] Kim-Hui Yap, Zhen Li, Da-Jiang Zhang, and Zhan-Ke Ng. Efficient mobile landmark recognition based on saliency-aware scalable vocabulary tree. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 1001–1004. ACM, 2012.

[3] Tao Chen, Kim-Hui Yap, and Dajiang Zhang. Discriminative soft bag-of-visual phrase for mobile landmark recognition. *IEEE Transactions on Multimedia*, 16(3):612–622, 2014.

[4] Tao Chen and Kim-Hui Yap. Discriminative bow framework for mobile landmark recognition. *IEEE transactions on cybernetics*, 44(5):695–706, 2014.

[5] Tao Chen and Kim-Hui Yap. Context-aware discriminative vocabulary learning for mobile landmark recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(9):1611–1621, 2013.

[6] Jiuwen Cao, Tao Chen, and Jiayuan Fan. Fast online learning algorithm for landmark recognition based on bow framework. In *Industrial Electronics and Applications (ICIEA), 2014 IEEE 9th Conference on*, pages 1163–1168. IEEE, 2014.

[7] Jiuwen Cao, Tao Chen, and Jiayuan Fan. Landmark recognition with compact bow histogram and ensemble elm. *Multimedia Tools and Applications*, 75(5):2839–2857, 2016.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[9] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] David J Crandall, Yunpeng Li, Stefan Lee, and Daniel P Huttenlocher. Recognizing landmarks in large-scale social image collections. In *Large-Scale Visual Geo-Localization*, pages 121–144. Springer, 2016.

[13] Forrest N Iandola, Matthew W Moskewicz, Khalid Ashraf, Song Han, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and < 1mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[14] Chakkrit Termritthikun, Paisarn Muneesawang, and Surachet Kanprachar. Nu-innet: Thai food image recognition using convolutional neural networks on smartphone. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 9(2-6):63–67, 2017.

[15] Chakkrit Termritthikun and Surachet Kanprachar. Accuracy improvement of thai food image recognition using deep convolutional neural networks. In *Electrical Engineering Congress (iEECON), 2017 International*, pages 1–4. IEEE, 2017.

[16] Chakkrit Termritthikun and Surachet Kanprachar. Nu-resnet: Deep residual networks for thai food image recognition. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 10(1-4):29–33, 2018.

[17] Yoonsik Kim, Insung Hwang, and Nam Ik Cho. A new convolutional network-in-network structure and its applications in skin detection, semantic segmentation, and artifact reduction. *arXiv preprint arXiv:1701.06190*, 2017.

[18] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[19] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[20] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[22] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, pages 4278–4284, 2017.

[23] Devansh Arpit, Yingbo Zhou, Bhargava Kota, and Venu Govindaraju. Normalization propagation: A parametric technique for removing internal covariate shift in deep networks. In *International Conference on Machine Learning*, pages 1168–1176, 2016.

[24] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[25] George E Dahl, Tara N Sainath, and Geoffrey E Hinton. Improving deep neural networks for lvcsr using rectified linear units and dropout. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8609–8613. IEEE, 2013.

**Chakkrit Termritthikun** was born in Roi-Et, Thailand in 1991. He received his B.Eng. degree in computer engineering and M.Eng. degree in computer engineering from Naresuan University, Phitsanulok, Thailand, in 2013 and 2017, respectively. He is currently pursuing a Ph.D. degree in computer engineering. He has been a Visiting Research Student with the University of South Australia, Australia since 2018. His current research interests include convolutional neural networks, deep learning, machine learning, artificial intelligence, and blockchain.

**Paisarn Muneesawang** (M'14 - SM'14) received his B.Eng. degree in electrical engineering from the Mahanakorn University of Technology, Bangkok, Thailand, in 1996, his M.Eng.Sci. degree in electrical engineering from the University of New South Wales, Sydney, NSW, Australia, in 1999, and his Ph.D. degree from the School of Electrical and Information Engineering, University of Sydney, Sydney. He was a Post-Doctoral Research Fellow with Ryerson University, Toronto, ON, Canada, from 2003 to 2004, and an Assistant Professor with the College of Information Technology, University of United Arab Emirates, Al Ain, UAE, from 2005 to 2006. He has been a Visiting Professor with Nanyang Technological University, Singapore, and Ryerson University, Toronto, since 2012 and 2013, respectively. He was the Vice President of Administrative Affairs, Naresuan University, Phitsanulok, Thailand, where he is currently a Professor. He co-authored Multimedia Database Retrieval: A Human-Centered Approach (Springer, 2006) and Unsupervised Learning-A Dynamic Approach (Wiley-IEEE Press, 2013). He co-edited Advances in Multimedia Information Processing - PCM 2009 (Springer, 2009). His current research interests include multimedia signal processing, computer vision, and machine learning. He has served as the Registration Co-Chair of the International Conference on Multimedia and Expo 2006 and the Technical Program Co-Chair of the Pacific-Rim Conference on Multimedia 2009.