Pertanika J. Sci. & Technol. 30 (2): 1157 - 1170 (2022)

PERTANIKA JOURNALS

SCIENCE & TECHNOLOGY

Journal homepage: http://www.pertanika.upm.edu.my/

Image Classification for Edge-Cloud Setting: A Comparison Study for OCR Application

Kenneth Kean Hoong Tan[1], Yee Wan Wong[2] and Hermawan Nugroho[1]*

[1]*Department of Electrical and Electronic Engineering, University of Nottingham in Malaysia, Selangor, Semenyih 43500, Malaysia*
[2]*Merimen Malaysia, UPM-MTDC Technology Center III, Universiti Putra Malaysia, 43400 UPM, Serdang, Selangor, Malaysia*

ABSTRACT

The increasing number of smart devices has led to a rise in the complexity and volume of the image generated. Deep learning is an increasingly common approach for image classification, a fundamental task in many applications. Due to its high computational requirements, implementation in edge devices becomes challenging. Cloud computing serves as an enabler, allowing devices with limited resources to perform deep learning. For cloud computing, however, latency is an issue and is undesirable. Edge computing addresses the issue by redistributing data and tasks closer to the edge. Still, a suitable offloading strategy is required to ensure optimal performance with methods such as LeNet-5, OAHR, and Autoencoder (ANC) as feature extractors paired with different classifiers (such as artificial neural network (ANN) and support vector machine (SVM)). In this study, models are evaluated using a dataset representing Optical Character Recognition (OCR) task. The OCR application has recently been used in many task-offloading studies. The evaluation is based on the time performance and scoring criteria. In terms of time performance, a fully connected ANN using features from the ANC is faster by a factor of over 60 times compared to the fastest performing SVM. Moreover, scoring performance shows that the SVM is less prone to overfit in the case of a noisy or imbalanced dataset in comparison with ANN. So, adopting SVM in which the data distribution is unspecified will be wiser as there is a lower tendency to overfit. The training and inference time, however, are generally higher than ANN.

Keywords: Artificial neural networks, convolutional neural networks, edge computing, image classification, support vector machines

ARTICLE INFO

Article history:
Received: 01 July 2021
Accepted: 28 December 2021
Published: 14 March 2022

DOI: https://doi.org/10.47836/pjst.30.2.17

E-mail addresses:
Kenneth.Tan@nottingham.edu.my (Kenneth Kean Hoong Tan)
yeewan6823@gmail.com (Yee Wan Wong)
Hermawan.Nugroho@nottingham.edu.my (Hermawan Nugroho)
* Corresponding author

ISSN: 0128-7680
e-ISSN: 2231-8526

INTRODUCTION

Image classification is assigning an input image the one or more classes. It is a fundamental task with many applications, including object identification, image captioning, as well as face and emotion recognition. Typically, the task can be approached manually with a set of algorithms designed by an expert in digital image processing or autonomously using a machine learning classifier. In many cases, it involves a combination of both approaches, where the image is pre-processed before being fed into a classifier. With the advent of the Internet-of-Things (IoT) and the rise in smart household and personal devices, digital image data is generated faster than ever. As the volume and complexity of the data increase exponentially, more sophisticated methods are required to tackle the problem. It has led many data scientists to turn to deep learning as a means of solving the classification problem (Chen & Ran, 2019).

However, a higher degree of computing power is required for the task, making it difficult for the solution framework to be deployed on the end device. Cloud computing seemingly addresses the problem at the cost of time as uploading data to the cloud introduces latency to the system. It is further complicated when factoring in bandwidth, reliability, and resource constraints such as battery life (Chang et al., 2019), leading to the ever-familiar conundrum of modern data scientists–balancing computational accuracy, resource availability, and cost-efficiency.

In many real-world image classification applications, the problem often requires a real-time solution. The latency that entails using a cloud-based solution thus poses a problem for such applications. For instance, in a security-surveillance application, the system is required to detect trespassers and other anomalies in real-time. However, implementing the entire framework on the edge device is expensive and will lead to feasibility and scalability issues, particularly when there is a multitude of devices involved. Streaming entire video feeds to the cloud is equally implausible due to bandwidth and security issues. The need to resolve these challenges has given rise to edge intelligence (Zhou et al., 2019).

In recent years, artificial intelligence (AI) researchers have been looking into edge computing as a means to bridge the gap between computational accuracy, latency, and resource management. Edge computing extends the capabilities of centralized cloud computing by distributing program methods and data to the network edge to enhance performance and efficiency (Dube et al., 2021). It also addresses scalability issues posed by network bottlenecking from many devices connected to the network (Chen & Ran, 2019).

In implementing edge intelligence, there are several approaches that can be taken depending on the application involved, and typically involves data and task offloading to and from sensors, devices, and the cloud (Nee & Nugroho, 2020). In many cases, middleware such as cloudlets and data centers are used and are covered in greater detail (Xu et al., 2019). Further focusing on the scenario of offloading from cloud to edge, there

are two main partitioning strategies: static and dynamic. Static partitioning is established offline before the execution of an application, while dynamic partitioning occurs during the runtime of an application (Yan et al., 2018). While the latter may lead to better practicality when considering communication conditions, the study is interested in the former due to its lower energy cost.

For interpretability and modularity, the study splits the classification framework into a feature extractor to be implemented on the edge and a classifier to be implemented on the cloud. However, it becomes difficult to pick a suitable feature extractor and classifier without a priori knowledge of its effectiveness and efficiency. To that end, this paper presents a comparison study on the image classification performance of fully connected feed-forward artificial neural networks (ANN), support vector machines (SVM), and an enhanced SVM (ESVM), with emphasis on-time performance as well as the effect of various CNN architectures on feature extraction performance. The classifier and the feature extraction methods are selected based on the recent trends for OCR applications (Cheriet et al., 2007; Elleuch et al., 2016a; Verma & Ali, 2019).

MATERIALS AND METHODS

Experimental Setup

The limited memory and processing capabilities on an edge device would make it slow and ineffective, whereas sending many high-resolution images to the cloud would lead to service bottlenecks and the inherent latency in delivery. In order to achieve a good balance in performance, the classifier is statically offloaded to the cloud, similar to model partitioning (Zhou et al., 2019). Thus, the study emulates having a pre-trained feature extractor on the edge device and a classifier on the cloud. It means smaller data will be passed to the cloud, reducing service bottleneck issues and the accompanying latency.

In order to emulate an edge-cloud computing framework, the classification task is conducted in two parts. The first part consists of a CNN feature extractor, while the second consists of a classifier framework. As smart devices become commonplace and cameras more sophisticated, performing feature extraction and classification on either the cloud or edge becomes infeasible for real-time performance. For the application, an optical character recognition (OCR)-based function was emulated. The OCR application has been adopted in many studies regarding task offloading of edge computing/intelligence (Cao et al., 2019; Li et al., 2018; Lin et al., 2019).

The experiment was conducted using the R statistical programming language and the LIBSVM package,

MNIST Dataset. For the study, the MNIST handwritten digit dataset is used as the model data (LeCun et al., 1998). Widely used in many experiments as benchmark data, the data

consists of 70000 28x28 grayscale images of handwritten digits in 10 classes (Amri et al., 2018; Ghiassirad et al., 2019). Each pixel has a value ranging from 0 to 255, and the categorical labels range from 0 to 9, corresponding to the handwritten digit of the image. The dataset is split into a training and testing set at a ratio of 6:1 and is distributed, as shown in Table 1.

Table 1
Distribution of MNIST data by class

Label	Training	Testing
0	5923	**980**
1	6742	**1135**
2	5958	**1032**
3	6131	**1010**
4	5842	**982**
5	5421	**892**
6	5918	**958**
7	6265	**1028**
8	5851	**974**
9	5949	**1009**

Feature Extraction Using CNN. Convolutional Neural Networks (CNN) are a class of deep learning algorithms that excel at handling images and other forms of data that exhibit spatial or temporal dependencies. It is a deep neural network architecture consisting of stacked convolutional and pooling layers. It creates multiple feature maps that are subsequently connected to a classifier or regressor for the final task. The network learns the features which best represent the model via back-propagation and can also be used as an auto-encoder via unsupervised learning. It makes it highly suited for image processing applications where there is a high level of spatial correlation in the data. Many researchers have turned to CNNs for various image-related applications, such as face recognition (Ding & Tao, 2015), biometric authentication (Hammad et al., 2019), and object detection (Ismail et al., 2020). CNNs have been reported to exhibit feature extraction capabilities exceeding those of conventional methods, as well as being easier to train due to having fewer connections and parameters (Krizhevsky et al., 2012). It has also been reported that CNNs outperform Deep Belief Networks (DBN) and Deep Neural Networks (DNN) in terms of speed (Amri et al., 2018). However, CNNs are still expensive to implement on high-resolution images, limiting their applications on edge devices with limited memory and processing power.

Here, three different CNN architectures implemented are scrutinized using the Keras backend (Allaire & Chollet, 2019), namely LeNet-5 (Lecun et al., 1998), OAHR (Elleuch et al., 2016b), and a simple autoencoder (ANC) (Chollet, 2016). The CNNs were selected as they were designed with 28x28 handwritten character images, thus suitable for use with the MNIST dataset. The architectures of each network are detailed in Table 2. LeNet-5 and OAHR were trained with a batch size of 32 over 10 epochs, while the autoencoder was trained with a batch size of 128 over 50 epochs. The ANN dense layers in LeNet-5 and OAHR were treated as classifiers. The same layers were appended after layer 6 in the autoencoder and trained for classifier performance after the autoencoder training was concluded.

Table 2
CNN architectures

Name	Optimizer	Loss	Layers
LeNet-5	RMSProp	categorical crossentropy	**2d_conv(3x3, filters = 6, activation = ReLu) average_pooling(2x2) 2d_conv(3x3, filters = 16, activation = ReLu) average_ pooling(2x2) flatten(units = 400) dense(units = 120, activation = ReLu) dense(units = 84, activation = ReLu) output(units = 10, activation = softmax)**
OAHR	RMSProp	categorical crossentropy	**2d_conv(5x5, filters = 6, activation = ReLu) max_ pooling(2x2) 2d_conv(5x5, filters = 12, activation = ReLu) max_ pooling(2x2) flatten(units = 192) dense(units = 120, activation = ReLu) dense(units = 84, activation = ReLu) output(units = 10, activation = softmax)**
Autoencoder (ANC)	AdaDelta	binary crossentropy	**2d_conv(3x3, filters = 16, padding = same, activation = ReLu) max_pooling(2x2, padding = same) 2d_conv(3x3, filters = 8, padding = same, activation = ReLu) max_pooling(2x2, padding = same) 2d_conv(3x3, filters = 8, padding = same, activation = ReLu) max_pooling(2x2, padding = same) 2d_conv(3x3, filters = 8, padding = same, activation = ReLu) 2d_upsampling(2x2) 2d_conv(3x3, filters = 8, padding = same, activation = ReLu) 2d_upsampling(2x2) 2d_conv(3x3, filters = 16, padding = same, activation = ReLu) 2d_upsampling(2x2) 2d_conv(3x3, filters = 1, activation = sigmoid)**

Classification with SVM and Euclidean SVM (ESVM). While support vector machines (SVM) are no strangers to the field of AI, they are by no means obsolete either. Originating from the work of Vapnik (2000), SVMs implement Structural Risk Minimization (SRM) to compute the optimal separating hyperplane. Input data is mapped into higher dimensional feature space using the "kernel trick," where the kernel transformation need not be done explicitly but can be found implicitly using dot products in the form of kernel functions. These kernel functions are always convergent, provided that they satisfy Mercer's conditions (Burges, 1998). Thus, a unique set of support vectors representing each class is always obtainable after training. The optimal separating hyperplane is then computed by maximizing the margin between the support vectors of each class. In practice, the problem is implemented as a minimization of the dual form of the Lagrangian, as in Equation 1:

$$\min_{\alpha} -\frac{1}{2}\sum_i \sum_j y_i y_j \alpha_i \alpha_j (x_i \cdot x_j) + \sum_j \alpha_j , \quad (1)$$

$$s.t. \quad \sum_i y_i \alpha_i = 0$$

$$0 \le \alpha_i \le C, \quad \forall i$$

SVMs are desirable over loss minimization classifiers whose performance may vary due to convergence at local minima (Boser et al., 1992). Thus, SVMs are highly resilient against the Curse of Dimensionality, making them suitable for processing data that is large, complex, and noisy. While the SVM is a good classifier by nature, there are many challenges to optimizing and using it due to its complex nature. Various improvements on the fundamental SVM model have also been observed in recent years, such as the use of weighted kernel functions (Varatharajan et al., 2018), optimization algorithms (Shen et al., 2016), and multiple kernel learning (MKL) methods (Liu & Gu, 2020; Saeed & Ong, 2019). While much work has been done in those areas, the bottleneck of an SVM's performance lies ultimately in the selection of kernel function, which exhibits varying performance depending on the nature of the input data (Lee et al., 2012). In dealing with an unknown problem, SVMs will more often than not produce subpar performance in terms of accuracy and robustness. It is mainly due to the problem of hyperparameter and kernel selection (Johnson & Khoshgoftaar, 2019). Thus, the study turns to Euclidean SVM (ESVM), an enhanced SVM reported having a low dependency on kernel and hyperparameter value selection (Lee et al., 2012; Wan et al., 2012). The ESVM is fundamentally different from an SVM due to its inference strategy. While the training is carried out in the same manner, the hyperplane classification task is replaced by a Euclidean similarity measure. Unseen examples are compared in data space to the support vectors belonging to each class and are thus resistant to performance degradation via incorrect kernel selection. A breakdown of its implementation as well as its performance under various hyperparameters and kernels, can be found (Lee et al., 2012; Wan et al., 2012). However, previous work is performed only on text documents, where the nature of features is different compared to images. Hence, the study sets up the experiment to compare its performance versus a fundamental SVM. Parameter selection is performed based on the default values of LIBSVM (Chang & Lin, 2011), as depicted in Table 3. In the LIBSVM configuration, the radial basis function (RBF) kernel is used, which is defined as Equation 2:

$$K(x, y) = exp^{-\gamma|x-y|^2} \quad (2)$$

Table 3
LIBSVM default configuration

Parameter	Value
svm_type	**0 (C-SVC)**
kernel_type	**2 (RBF kernel)**
gamma	**1/num_features**
cost	**1**
tolerance	**0.001**

where γ is the equivalent of $\frac{1}{2\alpha^2}$. The hyperparameter for cost is used in the minimization problem posed in Equation 1, whereas tolerance is the training termination criterion.

Performance Evaluation Metrics. In addition to time performance, several performance metrics are used to measure the selected classifiers' performance. These metrics are summarized from the confusion matrix, where metrics commonly used to evaluate machine learning algorithms such as true positive (TP), false positive(FP), false negative(FN), and true negative (TN) values can be derived (Sujatha & Rajagopalan, 2017).

Table 4 shows a confusion matrix for a multi-class classification problem. We can extend the matrix as required to obtain the terms necessary as follows (Sokolova & Lapalme, 2009):

- Average accuracy (*AvgAcc*) – The average per-class performance of a classifier in Equation 3.

$$AvgAcc = \frac{\sum_{i=1}^{l} \frac{TP_i + TN_i}{TP_i + FN_i + FP_i + TN_i}}{l} \quad (3)$$

- Error rate (ErrRate)– The average per-class classification error of a classifier in Equation 4.

$$ErrRate = \frac{\sum_{i=1}^{l} \frac{FP_i + FN_i}{TP_i + FN_i + FP_i + TN_i}}{l} \quad (4)$$

- Positive predictive value (PPV) – Also known as precision, PPV is the possibility of a sample predicted in class to belong to the class truly. Here, micro-averaging (denoted with subscript μ) favors bigger classes while macro-averaging (denoted with subscript *M*) treats all classes equally. Thus, if both measures of the same metric are similar, it can be said that the dataset is balanced, which is usually not the case for real-world data. The micro-and macro-averaged values are defined as in Equation 5:

Table 4
Multiclass confusion matrix

Reference	Predicted		
	Class 1	**Class 2**	Class 3
Class 1	$[TP_1, TN_2, TN_3]$	$[FN_1, FP_2, TN_3]$	$[FN_1, TN_2, FP_3]$
Class 2	$[FP_1, FN_2, TN_3]$	$[TN_1, TP_2, TN_3]$	$[TN_1, FN_2, FP_3]$
Class 3	$[FP_1, TN_2, FN_3]$	$[TN_1, FP_2, FN_3]$	$[TN_1, TN_2, TP_3]$

$$PPV_{\mu} = \frac{\sum_{i=1}^{l} TP_i}{\sum_{i=1}^{l}(TP_i+FP_i)} \quad (5)$$

$$PPV_M = \frac{\sum_{i=1}^{l} \frac{TP_i}{TP_i+FP_i}}{l}$$

- Negative predictive value (NPV)—The mirror version of the PPV, NPV is the possibility of a rejection not belonging to the class (Sokolova & Lapalme, 2009)] and the micro-and macro-averaged values can be defined as Equation 6:

$$NPV_{\mu} = \frac{\sum_{i=1}^{l} TN_i}{\sum_{i=1}^{l}(TN_i + FN_i)} \quad (6)$$

$$NPV_M = \frac{\sum_{i=1}^{l} \frac{TN_i}{TN_i + FN_i}}{l}$$

- True positive rate (TPR)—Also known as sensitivity or recall, TPR measures the ability of the classifier to classify a sample that belongs to the class correctly. The micro-and macro-averaged values are as in Equation 7:

$$TPR_{\mu} = \frac{\sum_{i=1}^{l} TP_i}{\sum_{i=1}^{l}(TP_i+FN_i)} \quad (7)$$

$$TPR_M = \frac{\sum_{i=1}^{l} \frac{TP_i}{TP_i+FN_i}}{l}$$

- True negative rate (TNR)—Also known as specificity, TNR is a mirror version of TPR and measures the classifier's ability to reject samples not belonging to a class correctly. The micro-and macro-averaged values are as in Equation 8:

$$TNR_{\mu} = \frac{\sum_{i=1}^{l} TN_i}{\sum_{i=1}^{l}(TN_i+FP_i)} \quad (8)$$

$$TPR_M = \frac{\sum_{i=1}^{l} \frac{TN_i}{TN_i + FP_i}}{l}$$

- F-score – F-score is the harmonic mean of recall and precision. It is typically used to show the relation between the data's positive labels and those given by the classifier. While there are many variants of the F-score, such as the F-1, F-0.5, and F-2 score, the study is interested in the classifier's ability to balance both false positives and false negatives; hence the study uses the F-1 metric with $\beta = 1$ (Equation 9).

$$Fscore_{\mu} = \frac{(\beta^2+1)PPV_{\mu} \cdot TPR_{\mu}}{\beta^2 \cdot PPV_{\mu} + TPR_{\mu}} \quad (9)$$

$$Fscore_{M} = \frac{(\beta^2+1)PPV_{M} \cdot TPR_{M}}{\beta^2 \cdot PPV_{M} + TPR_{M}}$$

RESULTS AND DISCUSSION

Time Performance

Table 5 shows the time performance of the three feature extractors discussed in the Material and Method section in terms of training and testing performance, while Table 6 depicts the time performance of the various classifiers under different features extracted from the feature extractors. LeNet-5 and OAHR were trained together with the ANN's fully connected (FC) layers due to its supervised nature, and thus the training time is combined in Table 5.

In terms of time performance, the study finds that feature extraction using only the encoder layers in the ANC gave the best time performance, despite having more parameters compared to LeNet-5 and OAHR at their best. Output dimension-wise, both LeNet-5, and OAHR at CNN+2FC give a vector of 84 features, whereas ANC gives 128 features. These studies favor LeNet-5 and OAHR as fewer features equate to a smaller payload transmitted to the cloud. The training time for ANC is almost double that of LeNet-5 and OAHR due to having twice the number of convolutional layers; however, as the networks are to be pre-trained before implementation on the edge device, the setback is minor.

At the classifier end, the fastest result was achieved for the testing on 10000 samples by a 3-layer fully connected ANN using features from the ANC. It is faster by a factor of over 60 times compared to the fastest performing SVM and over 1000 times against the fastest ESVM. It can also be observed. It can also be observed that in most cases, the inference time of ESVM approaches that of the fundamental SVM training time. In terms of training time, the study observed that training with the most number of features (LeNet-5 CNN only, 400 features) took only 934.27s, which was substantially faster than

Table 5
Feature extraction time performance

Feature extractor	Time performance
LeNet-5	**CNN + 3FC training: 95.68s** **CNN + 2FC feature extraction: 2.43s** **CNN only feature extraction: 2.5s**
OAHR	**CNN + 3FC training: 93.89s** **CNN + 2FC feature extraction: 2.52s** **CNN only feature extraction: 2.89s**
Autoencoder (ANC)	**CNN only training: 180.01s** **CNN only feature extraction: 2.3s**

Table 6
Classifier time performance

Input features	Time performance
CNN + 3-layer FC	**ANC features train: 73.64s** **ANC features test: 0.28s** **LeNet-5 features test: 0.43s** **OAHR features test: 0.50s**
CNN + 2-layer FC	**LeNet-5 features SVM train: 559.71s** **LeNet-5 features SVM test: 17.03s** **LeNet-5 features ESVM test: 304.83s** **OAHR features SVM train: 2268.71s** **OAHR features SVM test: 45.91s** **OAHR features ESVM test: 2337.05s**
CNN only	**ANC features SVM train: 1031.72s** **ANC features SVM test: 65.05s** **ANC features ESVM test: 1037.36s** **LeNet-5 features SVM train: 934.27s** **LeNet-5 features SVM test: 62.00s** **LeNet-5 features ESVM test: 652.61s** **OAHR features SVM train: 13348.05s** **OAHR features SVM test: 193.24s** **OAHR features ESVM test: 3566.28s**

both OAHR configurations (84 features, 2268.71s; 192 features, 13348.05s) and marginally faster than the ANC configuration (128 features, 1031.72s). The inference times of said configurations exhibit a similar trend. Hence, there is no direct apparent relation between the number of features used and the training time of an SVM, and further work should be conducted to discover the relation between the nature of features extracted and the time performance of the SVM.

Classification Performance

Table 7 shows the classification performance of the best performing classifier corresponding to each feature extractor. For LeNet-5, The best performing classifier was the fundamental SVM using CNN-only features. In OAHR and ANC, both feature extractors worked best with the 3-layer ANN. In all cases, both the fundamental SVM and ANN outperform ESVM in classification performance. While ESVM has been shown in the literature to have a low dependency on kernel and hyperparameter selection, its effectiveness is still poor compared to a fundamental SVM. It makes it unsuitable for applications where classification performance is important. The study also notes that the fundamental SVM achieved marginally better performance when using CNN-only features in LeNet-5, as depicted in Table 8. However, there was no distinct difference in performance for ESVM regardless of whether CNN only or CNN+2FC features are used.

From the metrics described in the previous section, the performance of the classifiers can be scrutinized further. Looking at the selected classifiers in Table 7, despite the slight imbalance of the MNIST dataset, the SVM does not show much difference between the macro and micro attributes of the extended metrics. However, the ANNs have slightly better overall macro performance. It may indicate that in the event of a highly imbalanced dataset, ANNs are more susceptible to overfitting compared to SVMs.

Comparing the extended metrics of the two classifiers, while both SVMs and ANNs exhibit a strong ability to reject correctly due to high values of NPV and TNR, their abilities to classify correctly are comparatively weaker, more so with ANN. Thus, while there is no clear winner among the classifiers, selecting SVM over ANN and ESVM for an application where the data distribution is unknown will be wiser as there is a lower tendency to overfit

Table 7
Best classification performance by feature extractor

Feature extractor	Classifier	Accuracy	PPV	NPV	TPR	TNR	F-Score
LeNet-5	SVM	AvgAcc: 0.9979	PPV_M : 0.9894	NPV_M: 0.9988	TPR_M: 0.9892	NPR_M: 0.9988	**$Fscore_M$: 0.9893**
		ErrRate: 0.002	PPV_μ: 0.9893	NPV_μ: 0.9988	TPR_μ: 0.9893	NPR_μ: 0.9988	**$Fscore_\mu$: 0.9893**
OAHR	ANN	AvgAcc: 0.9948	PPV_M : 0.9741	NPV_M : 0.9971	TPR_M: 0.9744	NPR_M: 0.9971	**$Fscore_M$: 0.9743**
		ErrRate: 0.0052	PPV_μ: 0.9741	NPVμ: 0.9971	TPR_μ: 0.9741	NPR_μ: 0.9971	**$Fscore_\mu$: 0.9741**
Autoencoder (ANC)	ANN	AvgAcc: 0.9889	PPV_M : 0.9456	NPV_M : 0.9939	TPR_M: 0.9443	NPR_M: 0.9939	**$Fscore_M$: 0.9449**
		ErrRate: 0.0111	PPV_μ: 0.9446	NPVμ: 0.9938	TPR^μ: 0.9446	NPR_μ: 0.9938	**$Fscore_\mu$: 0.9446**

Table 8
SVM vs ESVM performance with LeNet-5 features

Classifier	Features	Accuracy	F-score
SVM	CNN+2FC	AvgAcc: 0.9971	**$Fscore_M$: 0.9854**
		ErrRate: 0.0029	**$Fscore_\mu$: 0.9854**
	CNN only	AvgAcc: 0.9979	**$Fscore_M$: 0.9893**
		ErrRate: 0.0021	**Fscoreμ: 0.9893**
ESVM	CNN+2FC	AvgAcc: 0.8072	**$Fscore_M$: 0.0196**
		ErrRate: 0.1928	**$Fscore_\mu$: 0.0361**
	CNN only	AvgAcc: 0.8072	**$Fscore_M$: 0.0195**
		ErrRate: 0.1928	**$Fscore_\mu$: 0.0361**

CONCLUSION

In this study, the researchers investigated the efficiency and effectiveness of LeNet-5, OAHR, and ANC as feature extractors paired against ANN, SVM, and ESVM as classifiers

for OCR application in the Edge-Cloud settings. The evidence found from this comparison study suggests that while ANNs have good classification and time performance, SVMs are less prone to overfit in the case of a noisy or imbalanced dataset. While ESVM has been shown in the literature to be resilient to incorrect kernel and parameter selection, the time performance in both training and inference suggests that it is unsuited for applications where latency is a major concern. In addition, this study finds that the training time of an SVM classifier is independent of the number of input features used. The study believes the findings will assist future image classification application research directions, especially for OCR applications in the cloud and edge settings.

ACKNOWLEDGMENT

This work is supported by the Fundamental Research Grant Scheme (FRGS) FRGS/1/2018/ICT02/UNIM/02/4 from the Ministry of Higher Education Malaysia (MOHE).

REFERENCES

Allaire, J. J., & Chollet, F. (2019). keras: R Interface to 'Keras'. In *R package version 2.2.5.0.* RStudio. https://CRAN.R-project.org/package=keras.

Amri, A. A., Ismail, A. R., & Zarir, A. A. (2018). Comparative performance of deep learning and machine learning algorithms on imbalanced handwritten data. *International Journal of Advanced Computer Science and Applications*, *9*(2), 258-264. https://doi.org/10.14569/IJACSA.2018.090236

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). Training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory* (pp. 144-152). ACM Publishing. https://doi.org/10.1145/130385.130401

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery, 2*(2), 121-167. https://doi.org/10.1023/A:1009715923555

Cao, J., Yang, L., & Cao, J. (2019). Revisiting computation partitioning in future 5G-based edge computing environments. *IEEE Internet of Things Journal*, *6*(2), 2427-2438. https://doi.org/10.1109/JIOT.2018.2869750

Chang, C. C., & Lin, C. J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, *2*(3), 1-27. https://doi.org/10.1145/1961189.1961199

Chang, C., Srirama, S. N., & Buyya, R. (2019). Internet of things (IOT) and new computing paradigms. In *Fog and Edge Computing: Principles and Paradigms, 6*, 1-23. https://doi.org/10.1002/9781119525080.ch1

Chen, J., & Ran, X. (2019). Deep learning with edge computing: A review. *Proceedings of the IEEE*, *107*(8), 1655-1674. https://doi.org/10.1109/JPROC.2019.2921977

Cheriet, M., Kharma, N., Liu, C. L., & Suen, C. Y. (2007). *Character recognition systems: A guide for students and practioners*. John Wiley & Sons. https://doi.org/10.1002/9780470176535

Chollet, F. (2016). *Building autoencoders in keras*. The Keras Blog.

Ding, C., & Tao, D. (2015). Robust face recognition via multimodal deep face representation. *IEEE Transactions on Multimedia*, *17*(11), 2049-2058. https://doi.org/10.1109/TMM.2015.2477042

Dube, S., Wan, W. Y., & Nugroho, H. (2021). A novel approach of IoT stream sampling and model update on the IoT edge device for class incremental learning in an edge-cloud system. *IEEE Access*, *9*, 29180-29199. https://doi.org/10.1109/ACCESS.2021.3059251

Elleuch, M., Lahiani, H., & Kherallah, M. (2016). Recognizing Arabic handwritten script using support vector machine classifier. In *2015 15th International Conference on Intelligent Systems Design and Applications (ISDA)* (pp. 551-556). IEEE Publishing. https://doi.org/10.1109/ISDA.2015.7489176

Elleuch, M., Maalej, R., & Kherallah, M. (2016). A new design based-SVM of the CNN classifier architecture with dropout for offline Arabic handwritten recognition. *Procedia Computer Science*, *80*, 1712-1723. https://doi.org/10.1016/j.procs.2016.05.512

Ghiassirad, H. A., Shoorehdeli, M. A., & Farivar, F. (2019). Application of constrained learning in making deep networks more transparent, regularized, and biologically plausible. *Engineering Applications of Artificial Intelligence*, *85*, 421-428. https://doi.org/10.1016/j.engappai.2019.06.022

Hammad, M., Liu, Y., & Wang, K. (2019). Multimodal biometric authentication systems using convolution neural network based on different level fusion of ECG and fingerprint. *IEEE Access*, *7*, 26527-26542. https://doi.org/10.1109/ACCESS.2018.2886573

Ismail, A., Ahmad, S. A., Soh, A. C., Hassan, M. K., & Harith, H. H. (2020). Deep learning object detector using a combination of convolutional neural network (CNN) architecture (minivggnet) and classic object detection algorithm. *Pertanika Journal of Science and Technology*, *28*(Special Issue 2), 161-171. https://doi.org/10.47836/pjst.28.S2.13

Johnson, J. M., & Khoshgoftaar, T. M. (2019). Survey on deep learning with class imbalance. *Journal of Big Data*, *6*(1), 1-54. https://doi.org/10.1186/s40537-019-0192-5

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems, 25*, 1097-1105.

Lecun, Y., Bottou, L., Bengio, Y., & Ha, P. (1998). *LeNet*. IEEE Publishing.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324. https://doi.org/10.1109/5.726791

Lee, L. H., Wan, C. H., Rajkumar, R., & Isa, D. (2012). An enhanced support vector machine classification framework by using Euclidean distance function for text document categorization. *Applied Intelligence*, *37*(1), 80-99. https://doi.org/10.1007/s10489-011-0314-z

Li, B., He, M., Wu, W., Sangaiah, A. K., & Jeon, G. (2018). Computation offloading algorithm for arbitrarily divisible applications in mobile edge computing environments: An OCR case. *Sustainability*, *10*(5), Article 1611. https://doi.org/10.3390/su10051611

Lin, L., Liao, X., Jin, H., & Li, P. (2019). Computation offloading toward edge computing. *Proceedings of the IEEE*, *107*(8), 1584-1607. https://doi.org/10.1109/JPROC.2019.2922285

Liu, T., & Gu, Y. (2020). Multiple kernel learning for hyperspectral image classification. *Advances in Computer Vision and Pattern Recognition*, *55*(11), 259-293. https://doi.org/10.1007/978-3-030-38617-7_9

Nee, S. H., & Nugroho, H. (2020). Task distribution of object detection algorithms in fog-computing framework. In *2020 IEEE Student Conference on Research and Development (SCOReD)* (pp. 391-395). IEEE Publishing. https://doi.org/10.1109/SCOReD50371.2020.9251038

Saeed, S., & Ong, H. C. (2019). Performance of SVM with multiple kernel learning for classification tasks of imbalanced datasets. *Pertanika Journal of Science and Technology*, *27*(1), 527-545.

Shen, L., Chen, H., Yu, Z., Kang, W., Zhang, B., Li, H., Yang, B., & Liu, D. (2016). Evolving support vector machines using fruit fly optimization for medical data classification. *Knowledge-Based Systems*, *96*, 61-75. https://doi.org/10.1016/j.knosys.2016.01.002

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, *45*(4), 427-437. https://doi.org/10.1016/j.ipm.2009.03.002

Sujatha, J., & Rajagopalan, S. P. (2017). Performance evaluation of machine learning algorithms in the classification of parkinson disease using voice attributes. *International Journal of Applied Engineering Research*, *12*(21), 10669-10675.

Vapnik, V. N. (2000). *The nature of statistical learning theory*. Springer Science & Business Media. https://doi.org/10.1007/978-1-4757-3264-1

Varatharajan, R., Manogaran, G., & Priyan, M. K. (2018). A big data classification approach using LDA with an enhanced SVM method for ECG signals in cloud computing. *Multimedia Tools and Applications*, *77*(8), 10195-10215. https://doi.org/10.1007/s11042-017-5318-1

Verma, R., & Ali, J. (2019). Comparative analysis of advanced classification techniques for multilingual ocr systems. *International Journal of Scientific and Technology Research*, *8*(11), 1036-1040.

Wan, C. H., Lee, L. H., Rajkumar, R., & Isa, D. (2012). A hybrid text classification approach with low dependency on parameter by integrating K-nearest neighbor and support vector machine. *Expert Systems with Applications*, *39*(15), 11880-11888. https://doi.org/10.1016/j.eswa.2012.02.068

Xu, X., Liu, Q., Luo, Y., Peng, K., Zhang, X., Meng, S., & Qi, L. (2019). A computation offloading method over big data for IoT-enabled cloud-edge computing. *Future Generation Computer Systems*, *95*, 522-533. https://doi.org/10.1016/j.future.2018.12.055

Yan, L., Zhang, R., Han, Z., Qin, M., & Yang, S. (2018). Mobile computation offloading strategy based on static information and dynamic partition. In *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)* (pp. 1-5). IEEE Publishing. https://doi.org/10.1109/VTCSpring.2018.8417705

Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K., & Zhang, J. (2019). Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, *107*(8), 1738-1762. https://doi.org/10.1109/JPROC.2019.2918951