Empirical Analysis of Feature Reduction in Deep Learning and Conventional Methods for Foot Image Classification

JERMPHIPUT JARUENPUNYASAK\textsuperscript{1} AND RAKKRIT DUANGSOITHONG\textsuperscript{2}

\textsuperscript{1}Department of Biomedical Sciences and Biomedical Engineering, Faculty of Medicine, Prince of Songkla University, Songkhla 90110, Thailand
\textsuperscript{2}Department of Electrical Engineering, Faculty of Engineering, Prince of Songkla University, Songkhla 90110, Thailand

Corresponding author: Rakkrit Duangsoithong (rakkrit.d@psu.ac.th)

This work was supported in part by the Coordinating Center for Thai Government Science and Technology Scholarship Students (CSTS), in part by the National Science and Technology Development Agency (NSTDA), in part by the Prince of Songkla University for Graduate School Dissertation Funding for Thesis, and in part by the Thailand Research Fund (TRF) and Thaicom under Grant RDG6150001.

\section*{ABSTRACT}
Deep learning algorithms are employed in many applications, especially in medical fields such as gait analysis and human pose detection for rehabilitation. However, creating the desired model with deep learning algorithms requires high memory and computing costs, which is problematic because deep learning technologies must be run on low-power devices such as edge computing equipment. To deal with these problems, feature reduction methods reduce the memory and energy costs. This paper presents an empirical analysis of deep learning with feature reduction. The method classifies foot images for knee rehabilitation using convolutional and dense autoencoders. The obtained results are compared with those of conventional methods (histograms of oriented gradients and local binary pattern algorithms). The features were classified and compared using support vector machine, k-nearest neighbor, and multilayer perceptron methods. The experimental results demonstrate that the conventional method uses fewer features than the deep learning method with higher accuracy because its algorithm projects pixels onto the histogram. In addition, using fewer features in deep learning layers maintains high accuracy, which is beneficial for edge computing implementations.

\section*{INDEX TERMS} Feature Reduction, autoencoder, LBP, SVM, foot classification.

\section*{I. INTRODUCTION}
The global elderly population aged 80 years or over is expected to increase from 143 million in 2019 to 426 million in 2050 \cite{1}. The elderly are susceptible to various degenerative diseases such as osteoarthritis (OA). An estimated 130 million people worldwide suffer from OA \cite{2}. Knee OA patients have malfunctional knee movement and experience pain in their knee joints.

OA doctors typically prescribe pain-killing medicine, and promote exercise and physical therapy that maintain joint movement. However, some OA patients cannot perform their exercises correctly and continuously, either at home or in a hospital. Lack of understanding by doctors seems to reduce the effectiveness of treatment.

Knee rehabilitation patients typically move their foot while fixing their knee joint, as shown in Figure 1. In this scenario, the foot movement provides information on knee function, by which the doctor can assess the knee treatment. Using a goniometer, doctors typically measure the knee’s range of motion (ROM) by comparing the foot’s location relative to the knee joint, as shown in Figure 2. A doctor can also follow, assess, and suggest the patients. However, a goniometer cannot easily record all data manually at home and in the hospital during the exercise period.

Computer vision technologies are frequently used in many human pose-monitoring applications. For instance, a camera captures real-time human movement via object recognition. Besides being comfortable for home-based rehabilitation, computer vision technology sets up the monitoring systems of both patients and devices without difficulty. The Viola–Jones algorithm for face and hand detection \cite{5}, \cite{6} can monitor human activities, but its effectiveness is limited by the light...
This study presents an empirical analysis of deep learning for knee rehabilitation using convolutional and dense feature extraction methods, which classifies foot conditions, occluded objects, and the color contrast between the object and the background. To solve these problems, the authors of [7] applied a deep learning algorithm to object detection and human pose monitoring.

Deep learning is a new machine learning technique that can detect objects and monitor human pose activities. Deep learning extracts the low- and high-level information from large datasets. Typically, the deep learning architecture is similar to that of artificial neural networks, but involves a greater number of hidden layers and nodes than artificial neural networks. Accordingly, this model can accurately identify objects such as handwritten digits and pedestrians [8]–[10]. Convolutional neural networks (CNNs) [11], [12] are deep learning techniques for image recognition. A CNN typically comprises a series of convolution layers with kernel filters, pooling layers, and fully connected layers with a SoftMax function that classifies the target objects. However, CNNs consume a lot of memory and require high performance devices to compute the model. Moreover, the deep learning model must be customized to each application [7], [13].

Feature reduction performs to decrease the memory and energy costs, which is beneficial in applications such as low-power devices and edge computing. Feature reduction is operated by two main methods: feature selection, which filters the irrelevant or redundant features from an original feature dataset and maintains a subset of the original feature dataset, and feature extraction, which creates a new feature dataset. Feature extraction identifies the dominant features or attributes of the dataset, through for example, principle components analysis [14], linear discriminant analysis (LDA) [15], and autoencoders [16]–[18]. By reducing the number of features that describe the dataset, feature extraction also increases the speed of machine learning techniques such as classification techniques.

This study presents an empirical analysis of deep learning based on feature extraction methods, which classifies foot images for knee rehabilitation using convolutional and dense autoencoders. The results are compared with those of histogram of oriented gradients (HOG) [19] and local binary pattern (LBP) algorithms [20]. These features are compared and classified using the support vector machine (SVM) [21], k-nearest neighbor (kNN) [22], and multilayer perceptron (MLP) [23] methods.

The remainder of this paper is organized as follows. Section 2 reviews the literature, and Section 3 describes the materials and methods. Results are presented in Section 4 and discussed in Section 5. Conclusions are given in Section 6.

II. LITERATURE REVIEW

Data reduction through feature extraction has been a mainstay of image classification for many years. The information in an image is both complex and high-dimensional, necessitating a process that extracts the informative features from the image for object classification. This literature review discusses the various techniques of feature extraction for image classification, focusing on transforming the high-dimensional image data into low-dimensional feature vectors such as color, shape, texture, and deep learning features.

A color feature is a low-level feature for image classification. Combined with a threshold, color features are extracted by a pixel operation. Xiaoying et al. presented a human skin classification method using color feature vectors on the YCbCr color space [24]. Although this technique was a simple process, the human skin color required calibration from the dataset. Rachmawati et al. proposed a color histogram on RGB color space for multiclass fruit classification [25]. The method accurately classified thirty-two fruit subcategories, but the experiment was limited to specific viewpoints of image capture. Kaur and Dhingra [26] applied a color histogram on RGB color space to determine the probability of color histogram between the target and the query image. They used the statistical features for similarity matching in content-based image retrieval (CBIR). In an experiment, this CBIR classified ten classes of datasets from various views of the images. However, the performance of this CBIR depended on the similarity measurement and the size of the dataset. Briefly, the performance of image classification based on color features depends on the threshold value, color space, and classifier.

Unlike color features, shape features rely on the intensity of image pixels to create shape vectors such as object edges. Dalal and Triggs [19] identified the edges of objects using the HOG features, which represent the object shape based on the magnitudes and distribution of local intensity gradients. However, HOG feature extraction required a long computational time and might not perform well when the orientation of the object changes. Banerji et al. defined new HaarHOG features for image classification [27]. They performed a Haar wavelets transform on four frequency ranges in different orientations before the HOG feature extraction process. The HaarHOG features achieved higher image classification performance than the traditional HOG, but the HaarHOG feature size is four times that of the HOG feature.
To consider the details in the object, researchers have introduced texture features to image classification. Ojala et al. proposed the initial local binary pattern (LBP) for extracting texture features [31]. The LBP converts a grayscale image to a matrix of integer numbers using a pixel operation. The LBP method is simple, low complexity, and is scale- and rotation-invariant [20], [31], but the traditional LBP is limited to texture features of small size and structure. To remove these limitations, the LBPs have been modified by multiresolution grayscale and rotation invariance [32]. However, as the dimension of the LBP features increases with the number of neighborhoods, this method requires more time for feature calculation than the traditional LBP. Wan et al. proposed a block-based LBP (BLBP) for tissue classification [33], which compares the average intensities of the pixels in a fixed area around the center pixel. Accordingly, the BLBP feature can represent the global texture information. Wu and Sun [34] proposed a joint-scale LBP (JLBP) for global texture information. This method uses a multi-scale scheme that concatenates the individual LBP before mutual integration of a local block. To obtain the spatial contextual information, Xiao et al. proposed the two-dimensional Local Binary Pattern (2D-LBP) method [35], which uses a sliding window to calculate the weighted occurrence of pattern pairs in a feature map of the original LBP. The feature map computed by 2D-LBP was compared with that of the rotation-invariant uniform LBP pattern. In summary, the traditional LBP can represent the local textures of the object for image classification with low complexity. However, for global textures, the sampling parameters of LBP (scale, size, shape, and radius) must be adjusted for image classification.

In deep learning, features are commonly extracted by a trained model. Lopes and Valiati [36] presented a pre-trained CNN model with a SVM classifier for tuberculosis diagnosis. In their experiments, they extracted the features from the last layer of the fully connected layer using various models (GoogLenet, VGGNet, and ResNet). On the Montgomery dataset, the feature vector from the ResNet model was less accurate than the other models. It appeared that the ResNet architecture with 152 layers was excessive for binary image classification. Şengür et al. compared the features extracted from pre-trained AlexNet and VGG16 models [37]. The feature vectors from the NUAA and CASIA databases were extracted by the first fully connected layer, and face liveliness was detected by the SVM classifier. The concatenated features from the AlexNet and VGG16 models achieved a detection accuracy of 88%. Jogin et al. modified the AlexNet model for feature extraction on the CIFAR-10 dataset [38], and reported an image classification accuracy of 85.97%. Tranget et al. classified plant diseases using a convolutional autoencoder [39] with two kernels of code layer to extract the plant leaf images. Although the classification accuracy reached 98.8% with the SVM classifier, the number of encoder layers and pooling functions were optimized to the dataset. In summary, both CNNs and autoencoder are capable of feature extraction, but the types of pre-trained model, layer, and kernel function vary among datasets. Moreover, the pre-trained model might not contain common labels for the classes to be classified. Consequently, the fully connected layers may be modified for satisfactory classification accuracy.

The contributions of this paper are as follows: (a) to reduce the numbers of low- and high-level features in foot images, (b) to apply a single class of autoencoders to improve the classification performance of feature extraction, (c) to analyze the number of feature reductions and complexity, and therefore (d) to analyze the accuracy of conventional feature extraction based on three classifiers for binary classification. This study demonstrates that feature reduction can potentially increase the accuracy of classifying foot images using state-of-the-art techniques.

III. MATERIALS AND METHODS
This section describes the dataset used in this study, feature extraction with conventional and deep learning methods, the classifiers used, and our experimental evaluations. The experimental process is illustrated in Figure 3. There are three main processes: image input, feature extraction, and classification.

A. DATASET
The input images comprised two classes (N = 8,000: foot images (N = 4,000; Figure 4) and non-foot images (N = 4,000; Figure 5)). The foot images included images of feet from different perspectives, various brightness, bare feet, and wearing shoes/socks. The non-foot images were randomly cropped from living room images. Both image classes were taken from online datasets such as Pascal VOC2012 [40] and image-net.org [41]. These experiments used five-fold cross validation. A model was trained using four-folds of the data (N = 6,400), and the rest of the remaining dataset (N = 1,600) was reserved for testing the resulting model.
B. CONVENTIONAL FEATURE EXTRACTION

Two conventional feature extraction methods were used in these experiments: projecting and reducing the image pixels to a histogram of the image.

1) HISTOGRAM OF ORIENTED GRADIENTS

Dalal and Triggs [19] proposed the HOG for human recognition. This method (Figure 6) calculates the HOG of the image. The HOG process is initiated to configure the cell size, block size, and bin size parameters. Here, a gray image is calculated the gradients of both x-axis and y-axis, magnitude an angle of its. The edge histogram is created via gradient vote depending on the bin parameter. Then, the magnitude of the gradient votes is normalized (Equation 1) to be suitable for a variety of lighting conditions [19]. Finally, the 2D features are converted into a single vector of image features. An example of foot images of HOG is shown in Figure 7. In this study, there were two main experimental HOG configurations (Table 1).

\[
M_i = \frac{m_i}{\sqrt{\sum_{j=1}^{K} m_j^2 + e^2}}
\]  

![FIGURE 6. Histogram of Oriented Gradients method.](image)

Here, \( M_i \) is the normalized magnitude of the gradient vote in bin \( i (i = 1 \) to \( K \)), \( K \) is the number of cells in one block multiplied by the number of bins (\( N_{bin} \)), and \( e \) is a small constant value.

2) LOCAL BINARY PATTERNS

An LBP [42], [43] is a simple feature that calculates the converted gray color space for determining the LBP mask, as shown in Figure 8. Depending on its radius, the mask divides the image window into cells. Considering pixels along the clockwise direction, if the center pixel’s gray value is less than the neighbor’s value, the result is 1. In contrast, if the
center pixel’s gray value is greater than the neighbor’s value, the result is 0. Then, the eight digits of the binary number are converted to decimal numbers (Figure 9). The LBP features are then calculated using Equations 2 and 3.

\[
LBP(P, R) = \sum_{i=0}^{P-1} s(g_i - g_c)2^i
\]  

(2)

\[
s(x) = \begin{cases} 
1, & \text{if } x \geq 0 \\
0, & \text{otherwise} 
\end{cases}
\]  

(3)

Here, \( P \) is pixels on a circle of radius \( R \) at the center pixel that forms a circular symmetry set of neighbors, \( R \) is the radius of circle (\( R > 0 \)) (the distance for each of the neighbor pixels \( i \)), \( g_c \) is the gray level intensity at the center pixel, \( g_i \) is the gray level intensity at the neighbor \( i \), and \( s(x) \) is a step function defined by Equation 3.

### C. DEEP LEARNING USING AUTOENCODER FOR FEATURE EXTRACTION

Generally, autoencoders [16], [46] are a type of feedforward neural network, where the input is the same as the output. This method reduces the input feature space to a lower-dimensional coder, and then reconstructs the output from this representation. The coder is a latent representation that summarizes or compresses the dominant information of the input.

There are three main components in the autoencoder (Figure 10).

- The encoder reduces the dimension of the input when \( h = f(x) \), where \( x \) is the input image and \( h \) is the coder layer.
- The coder attempts to minimize the dimension of the input image.
- The decoder reconstructs the code to the output when \( r = g(h) \), where \( r \) the image output from the decoder process.

Two autoencoder structures were used in the experiments, i.e., a convolutional autoencoder and dense autoencoder. Both autoencoders were trained a single class of foot images (\( N = 4,000 \)) with 80% of the samples, and 20% of the samples were used for validation. The autoencoders were trained over 200 epochs. In addition, the Adam optimizer was used, and the mean squared error was used as the loss function.

1) CONVOLUTIONAL AUTOENCODER

There are two main layers in the convolutional autoencoder. The first layer is a convolution layer for operating as filtering on the image pixel. The second layer is a pooling layer that reduces the number of features from the convolution layer by using a function to represent the feature (Figure 11). In the experiments, one block of the encoder layer was included the convolution and pooling layers. An example feature map result obtained using the convolutional autoencoder is shown in Figure 12. The parameters of each layer are shown in Table 2.

2) DENSE AUTOENCODER

The dense autoencoder (Figure 13) compresses the feature to a one-dimensional feature. Here, the number of nodes of each layer is reduced continuously. Also, each layer is connected in a sequential structure. An example feature result obtained using the dense autoencoder is shown in Figure 12. The parameters of each layer are shown in Table 3.
### TABLE 2. Parameters of each layer in convolutional autoencoder configuration.

| Layer          | Convolutional Layer | Pooling layer |
|----------------|--------------------|---------------|
|                | No. of filter $^a$ | Act. fn. $^b$ | P. fn. $^c$ | size |
| Encoder layer1 | 32                 | relu          | max        | 2x2  |
| Encoder layer2 | 64                 | relu          | max        | 2x2  |
| Coder layer    | 128                | relu          | max        | 2x2  |

Note: $^a$ is the number of filter, $^b$ is the activation function, and $^c$ is the pooling function.

### FIGURE 12. Example of feature map of foot image in the autoencoder methods.

### TABLE 3. Parameters of each layer in dense autoencoder configuration.

| Layer          | Dense Layer |
|----------------|-------------|
|                | Number of dense | Activation function |
| Encoder layer1 | 256         | relu               |
| Encoder layer2 | 128         | relu               |
| Coder layer    | 64          | relu               |

### D. CLASSIFIERS

Three classifiers were used in these experiments: SVM, kNN, and MLP methods. Figure 3 shows the three steps of training a foot classifier.

1) Input images: a dataset of foot and non-foot images (Figure 4 and Figure 5, respectively).
2) Feature extraction using conventional methods and deep learning methods.
3) Classifiers: The model is trained the SVM, kNN, and MLP methods. The result is evaluated according to the accuracy of foot/non-foot detection.

1) SUPPORT VECTOR MACHINE

The SVM [47], [48] is based on decision planes, which divide the boundaries of classes. These planes can separate a group of objects into different classes. The SVM comprises training and classification phases. For the training phase, the feature dataset is training which involves the minimization of the error function. The kernel is used to transform data from the input (independent) to the feature space. Note that the larger upper bound results in more error penalization; thus, the upper bound should be chosen with care to avoid overfitting. In the classification phase, the unknown data are classified by the optimal plane (Equation 4) [47] to identify separable patterns, as shown in Figure 14.

$$\text{Optimal hyperplane} = \min \left( \frac{1}{2} w^T w \right)$$  \hspace{1cm} (4)

### FIGURE 14. Learning step using the SVM algorithm.

Here, $\text{Optimal hyperplane}$ is the optimal decision planes that divide the class boundaries with subject to: $y_n(w^T x_n + b) \geq 1$ for $n = 1, 2, 3, \ldots, N$, $w \in \mathbb{R}^d$, $b \in \mathbb{R}$, $w$ is the vector of coefficients, $y_n$ represents the class labels and $x_n$ represents the independent variables, $b$ is a constant, and $N$ is the number of training cases.

In these experiments, the SVM classifier was utilized with a varying kernel function: linear, polynomial, and radial basis function (RBF).

2) K-NEAREST NEIGHBOR

The kNN [49] method is straightforward to understand and calculate. It is also a lazy algorithm because it does not involve a training phase. In the kNN method, feature data are typically in a metric space in scalars or multidimensional vectors. When considering the feature space, they provide a notion of distance, e.g., the Euclidean distance. The classification process in the kNN methods comprises five parts (Figure 15).

1) Select the $k$ values for classification.
2) Calculate the distance of feature between the unknown point and every data (Equation 5).
3) Sort the list of distances.
4) Find the class of the nearest $k$-distances.
FIGURE 15. Learning step using the kNN algorithm.

5) The highest frequency of the class corresponding the nearest k-distances will classify to the class of unknown point.

$$\| P \| = \sqrt{\sum_{i=1}^{N} (u_i - d_i)^2}$$ (5)

Here, $P$ is the Euclidean distance of feature, $N$ represents the number of feature and $u_i$ represents the feature variables of unknown point, and $d_i$ is the feature variables of data point.

In these experiments, the $k$ values for foot classification are varied from 3, 5, 7, 9, and 11.

3) MULTILAYER PERCEPTRON

The MLP [50] is a type of feed-forward artificial neural network (ANN) that comprises three main layers (Figure 16). The first layer is an input layer to receive the data or features, and the middle layer is a hidden layer. The third layer is the output layer, which classifies or predicts the input. An MLP with a single hidden layer can approximate any continuous function. Supervised learning is typically applied to train the MLP. They train a set of feature and class pairs to learn to model the correlation between those features and their class. The training process adjusts the weights and biases of the model to minimize their error. Backpropagation is one method to modify the weight and bias adjustments, which is responded to the error function such as root mean squared error. In these experiments, the MLP with a single hidden layer was used with a varying number of nodes ($N_c$) from 1, 2, 4 to 1,024 nodes.

FIGURE 16. Multilayer Perceptron Models.

E. EVALUATION

In this section, we discuss our evaluation relative to feature reduction, complexity, and accuracy.

1) FEATURE REDUCTION

Feature reduction presents the number of features ($N_f$) of each method obtained via feature extraction. Moreover, it shows the percentage of the number of feature reduction comparing with ground truth. Here, a positive percentage value indicates feature reduction, and a negative value represents an increasing number of feature increases.

2) COMPLEXITY

Generally, the $O()$ notation [51] is used in computer science to analyze the performance or measure algorithm complexity. To illustrate this, if the algorithm has an $N$ iteration loop for calculation, the $O()$ is approximately $O(N)$ for this algorithm. As a result, the greater number of $N$ in $O()$ indicates higher algorithm complexity.

3) ACCURACY

We evaluated classification performance using the test images based on the confusion matrix. The most common measurement of classification is a percentage of accuracy, which assesses the total efficiency of the classification (Equation 6) [52].

Table 4 presents the evaluation of the frameworks. The columns show the frameworks’ predicted class and the rows are the actual class. The list details of Table 4 are defined as:

**TABLE 4. Confusion matrix for two classes’ classification.**

| Actual  | Foot   | Non-Foot |
|---------|--------|----------|
|         | TP     | FP       |
| Foot    | TP     | FN       |
| Non-Foot| TN     | TN       |

- $TP$ is a number of foot detection images in a foot data set.
- $FN$ is a number of non-foot detection images in a foot data set.
- $FP$ is a number of foot detection images in a non-foot data set.
- $TN$ is a number of non-foot detection images in a non-foot data set.

$$\text{Accuracy} \, (%) = \frac{TP + TN}{TP + TN + FP + FN} \cdot 100$$ (6)

IV. RESULTS

Here, we present the feature reduction, algorithm complexity, and the accuracy results for each compared algorithm.

A. FEATURE REDUCTION

In these experiments, the original number of features for a $128 \times 128$-pixel image was 16,384. According to Table 5, after pixel normalization, a number of features was similar to the original number of features ($N_f = 16,384$). Number of features in two HOG models (HOG_128 and HOG_512) can be reduced to 128 and 512, respectively. For LBP with $R = 1$, it exhibited the smallest number of features ($N_f = 10$). With the increase of $R = 2, 3, $ and $4$, the $N_f$ values were 18, 26, and 34, respectively.

With the autoencoder methods, a number of features of the convolutional autoencoder in encoder layer1 was the...
TABLE 5. Number of features.

| Methods        | Types            | $N_f$ | FR (%) |
|----------------|------------------|-------|--------|
| Ground truth pixels | Divided by 255   | 16,384| 0.00   |
| HOG            | HOG_128          | 128   | 99.22  |
|                | HOG_312          | 512   | 98.88  |
| LBP            | $R = 1$          | 10    | 99.94  |
|                | $R = 2$          | 18    | 99.89  |
|                | $R = 3$          | 26    | 99.84  |
|                | $R = 4$          | 34    | 99.79  |
| Convolutional  | Encoder layer1   | 32,768| -100.00* |
|                | Encoder layer2   | 4,096 | 75.00  |
|                | Coder layer      | 1,024 | 93.75  |
| Dense          | Encoder layer1   | 256   | 98.44  |
|                | Encoder layer2   | 128   | 99.22  |
|                | Coder layer      | 64    | 99.61  |

Note: $N_f$ is the number of features, FR is the percentage of feature reduction comparing with the ground truth, and *the negative number means increasing the number of features comparing with the ground truth.

maximum ($N_f = 32,768$). Note that this is greater than the original number of features. The encoder layer2 and coder layer reduced number of features to 4,096 and 1,024, respectively. The dense model can reduce the number of features higher than using the convolutional model for all layers. The encoder layer1, encoder layer2, and coder layer in the dense autoencoder minimized the number of features to 256, 128, and 64, respectively.

Considering the percentage of feature reduction, most feature extraction can achieve more than 93% of feature reduction compared to the ground truth with normalized pixels. However, a number of features of the convolutional autoencoder in encoder layer1 was double the number of ground truth features.

B. COMPLEXITY

Table 6 shows that the kNN classifier demonstrated complexity of $O(N_s \times N_f + k \times N_f)$. The SVM and MLP methods had also complexities of $O(N_f)$ and $O(N_f \times N_c)$, respectively, where $N_s$ is the number of samples, $N_f$ is the number of features, $N_c$ is the number of nodes in hidden layer in the MLP classifier, and $k$ is the number of nearest neighbors in the kNN classifier.

TABLE 6. Complexity of classifier.

| Classifier | Complexity |
|------------|------------|
| SVM        | $O(N_f)$   |
| kNN        | $O(N_s \times N_f + k \times N_f)$ |
| MLP        | $O(N_f \times N_c)$ |

C. ACCURACY

Figures 17 to 28 show the foot detection performance of the SVM, kNN, and MLP classifiers. Here, different parameter values were used for the number of nearest neighbors ($k = 3, 5, 7, 9, \text{and } 11$) and the number of nodes in hidden layer of MLP (i.e., 1 to 1,024 nodes).

1) CONVENTIONAL FEATURE EXTRACTION METHODS

As shown in Figure 17 and Table 7, it is clear that LBP features provide high average accuracy with the three classifiers. Specifically, LBP with the kNN classifier made great advances in classification accuracy (greater than 90%). Normalized pixels with the three classifiers led to an average accuracy of approximately 63%. However, the HOG features derived a low average accuracy of 50% with the three classifiers.

• SVM Using Conventional Features: The accuracy of obtained by the SVM using conventional methods is shown in Figure 18. As can be seen, the SVM with a linear kernel using the LBP feature outperformed the SVM with the RBF kernel. Besides, increasing the value of $R$ in LBP did not yield significantly different accuracy. With the normalized pixels, the SVM with RBF kernel achieved high accuracy of 71%. It is clear that the SVM classifier could detect objects at approximately 50% accuracy for HOG features.

• kNN Using Conventional Features: Figure 19 shows the accuracy of the kNN classifier using conventional methods. Here, LBP with the kNN classifier reached over 90% accuracy, which is higher than other features. Additionally, the increasing $k$ in the kNN classifier and the $R$ values in LBP did not obtain significantly different accuracies. For normalized pixels, increasing the number of $k$ in the kNN classifier reduced accuracy slightly. However, using HOG features with various $k$ values provided a detection accuracy of 50%.

• MLP Using Conventional Features: As shown in Figure 20, the accuracy of MLP using the conventional method such as LBP with the MLP classifier was reached to 80% when the number of nodes in hidden
layer was greater than four. For normalized pixels, reducing the number of nodes in hidden layer is significantly reduced accuracy. Although this experiment considered a wide range for the number of nodes in hidden layer, HOG features with MLP still produced 50% detection accuracy.

2) DEEP LEARNING USING AUTOENCODER FOR FEATURE EXTRACTION

The results obtained with the convolutional autoencoder are shown in Figure 21 and Table 8. As can be seen, the coder layer features provided high average accuracy for all three classifiers. The coder layer with the MLP classifier provided average accuracy of greater than 81%. The encoder layer2 was an average accuracy of approximately 70%. However, the encoder layer1 showed an average accuracy of 66% with the three classifiers. With the dense autoencoder, as shown in Figure 22 and in Table 9, among the features of the dense encoder model, the MLP classifier offered a high average accuracy close to 76%. SVM and kNN classifiers with these features achieved an accuracy of 72%.

- SVM Using Autoencoder Features: Figure 23 shows the accuracy of the SVM using the convolutional encoder model. The result demonstrated that the SVM with the linear kernel could gain about 75% for all features in convolutional encoder model. Features from coder layer and encoder layer2 with polynomial and RBF kernels contributed to higher accuracy than using encoder layer3. For the dense encoder, the results indicate that the SVM with polynomial and RBF kernels gave a classification accuracy of 75% for all features with the dense encoder model. The SVM with the linear kernel provided an accuracy of near 65%, as shown in Figure 24.
- kNN Using Autoencoder Features: Figure 25 demonstrates the accuracy obtained by the kNN classifier with the convolutional encoder model. Here, the results show that increasing the $k$ values in kNN did not affect accuracy significantly. The result still remained the same ($> 65\%$). Similar to the convolutional encoder model,
TABLE 9. Average of accuracy (%) using dense encoder model methods.

| Methods    | SVM (SD) | kNN (SD) | MLP (SD) |
|------------|----------|----------|----------|
| Encoder_layer1 | 74.60 (3.94) | 72.76 (1.83) | 78.99 (4.39) |
| Encoder_layer2 | 77.11 (5.17) | 72.23 (1.71) | 76.41 (9.78) |
| Encoder_layer3 | 76.78 (7.90) | 74.31 (1.34) | 78.09 (6.83) |

The dense encoder model (Figure 26) did not have a significant influence on accuracy when the $k$ values increased from 3 to 11. Here, an accuracy of more than 65% was obtained.

- **MLP Using Autoencoder Features**: Figure 27 describes that, for the convolutional encoder model, the coder layer with the MLP classifier achieved great accuracy with different numbers of nodes in hidden layer (except four nodes in hidden layer). In addition, using only one node in hidden layer with the coder layer features gained accuracy of 75%. Using encoder layer2 gained an accuracy of 75% with greater than using one node in hidden layer. Encoder layer1 performed high accuracy (approximately 75%) with greater than eight nodes in hidden layer. The accuracy was reported about 75% using dense encoder model with greater than two nodes in hidden layer (Figure 28). Additionally, the MLP with 32 to 1,024 nodes in hidden layer obtained accuracy of 80%. However, the MLP with 1 to 16 nodes in hidden layer reduced detection accuracy.

V. DISCUSSION

This section discusses the feature reduction analysis, algorithm complexity, and the detection accuracy of each algorithm.
A. FEATURE REDUCTION ANALYSIS
As shown in Table 5, considering the conventional methods, both the LBP and HOG methods projected the image pixels to their histogram. The HOG method performed multiple blocks and sliding windows to calculate its histograms, while the LBP method filtered the radius size to compute features around the center pixel to create histograms. Consequently, the number of features using LBP methods is less than that of HOG methods.

The convolutional autoencoder method used many kernel filters to manipulate image pixels. As a result, the number of features was still enormous. Unlike the convolutional autoencoder, the dense autoencoder method reduced the number of features by compressing the number of features in each layer; thus, the number of features used by this method is lower than that of the convolutional autoencoder method.

The conventional methods outperformed the autoencoder method in feature extraction because the conventional methods extract features by projecting or transforming the pixel to their histograms. Additionally, the autoencoder methods consider the relationship between all pixels and many kernel operations from the current layer to the next layer.

B. COMPLEXITY
The complexity of SVM was related to a number of features. As a result, LBP with \( R = 1 \) feature provides the lowest complexity among those feature extraction methods. Using coder layer in dense encoder model with SVM formulates the complexity lower than among methods of deep learning. The complexity of kNN classifier depended on a number of input images, a number of nearest neighbors, and a number of features. We found that an increasing number of kNN classifiers is no significant difference in accuracy, so the kNN classifier with \( k = 3 \) is sufficient for object classification. The complexity of MLP was conditional on the number of features and nodes in hidden layer. In consideration of accuracy and efficiency, the coder layer in dense encoder model with the SVM demonstrated the lowest complexity for encoder models, while the combination between LBP with \( R = 1 \) and SVM classifier showed the lowest complexity among the conventional methods.

C. ACCURACY
1) CONVENTIONAL FEATURE EXTRACTION METHODS
The kNN classifier without HOG features provided high detection accuracy. For HOG features, the variety of foot and non-foot images can reduce accuracy due to scaling and rotation invariant problems in HOG features [53]. Using LBP features, both kNN and MLP classifiers obtained high accuracy since these methods can detect images with scaling and rotating invariance. The LBP features also represent the rotation-invariant of dataset [20], [44], [45].

- **SVM Using Conventional Features**: When increasing the \( R \) values in the LBP features, there was no significant difference in detection accuracy because the foot outline tended to be similar to a rectangular trapezoid shape. As a result, using only \( R = 1 \) for LBP is adequate for detecting foot images. Both linear and RBF kernels outperformed the polynomial kernel with quadratic. It also requires suitable parameters for the SVM classifier [48].

- **kNN Using Conventional Features**: Increasing the \( k \) values did not yield important changes to detection accuracy due to the distribution of similarity of training images. Accordingly, the distance of \( k \)-nearest neighbors algorithm is a majority vote close to a group of training images [49].

- **MLP Using Conventional Features**: It requires optimizing the suitable number of nodes in hidden layer relating to type of feature and number of input features. However, using HOG features does not achieve acceptable results in classification due to the variety of scaling and rotation invariant [53].

2) DEEP LEARNING USING AUTOENCODER FOR FEATURE EXTRACTION
In the convolution model, using coder layer, an average accuracy was high because the features map of image using was identical to the edge line of foot images, as shown in Figure 12. With the dense model, the average accuracy for among feature extraction provided high accuracy than 72%. Although the features map of image using was not similar to the shape of foot images as shown in Figure 12, the dense model extracts dominant features or information about the foot image structure. Consequently, when using the non-foot image to extract features, the three classifiers could detect the differences between the foot and non-foot images.

- **SVM Using Autoencoder Features**: For the convolutional model, both the linear and RBF kernels among features of the encoder model gained high accuracy than using a polynomial kernel with quadratic. Thus, two degrees of the polynomial kernel may not be suitable, and this needs be adjusted the kernel to achieve acceptable accuracy in classification [47]. For the SVM with the dense model, both polynomial kernel with quadratic and RBF kernel with among features of encoder model can gain high accuracy than using the linear kernel. The features map of these might be a non-linear feature. Thus, the linear kernel cannot distinguish foot and non-foot images effectively.

- **kNN Using Autoencoder Features**: For the convolutional and dense models, increasing the \( k \) values did not remarkable differences in detection accuracy due to the balanced differentiation of distribution of foot and non-foot images, as shown in Figures 4 and 5.

- **MLP Using Autoencoder Features**: With the convolutional model, we found that using at least 16 nodes in hidden layer of the encoder model facilitated correct detection results (>75%). In addition, for the coder layer features of the convolutional models, using only one node in hidden layer of the MLP resulted in 75%
Relative to algorithm complexity, the SVM with the coder
than using a deep learning method with high foot detection
ods for foot classification. We found that the convolutional
feature reduction using deep learning and conventional meth-
In this paper, we have presented an empirical analysis of
of epochs which might be over-fitted to the training dataset
in the autoencoder model. As a result, these features might
not adequately provide high performance for practical foot
classification.

VI. CONCLUSION
In this paper, we have presented an empirical analysis of
feature reduction using deep learning and conventional meth-
ods for foot classification. We found that the convolutional
method using an LBP reduces the number of features more
than using a deep learning method with high foot detection
accuracy with the kNN classifier. The coder layer of the
dense autoencoder with the MLP classifier reduced the num-
ber of features and maintained high foot detection accuracy.
Relative to algorithm complexity, the SVM with the coder
layer of the dense encoder model and LBP with R = 1
provided the lowest complexity for the deep learning and
conventional methods, respectively. In the future, we plan
to modify autoencoder training by training on both foot
and non-foot images with a variety of epochs. Furthermore,
we intend to implement this work on edge computing for
foot classification to support a knee rehabilitation monitoring
system.

REFERENCES
[1] United Nations. Ageing. Accessed: Apr. 30, 2020. [Online]. Available:
https://www.un.org/en/global-issues/ageing/
[2] World Health Organization. Osteoarthritis. Accessed: Apr. 30, 2020.
[Online]. Available: https://www.who.int/medicines/areas/priority_
medicines/Ch6_12Osteo.pdf
[3] Versus Arthritis Organization. Osteoarthritis (OA) of the Knee.
Accessed: Apr. 30, 2020. [Online]. Available: https://www-versusarthritis.org/about-arthritis/conditions/osteoarthritis-of-the-knee/
[4] C. Wilson, (May 21, 2019). Knee Range of Motion. Knee-Pain-Explained. Accessed: May 2, 2020. [Online]. Available: https://www.knee-pain
explained.com/knee-range-of-motion.html
[5] J. Huang, Y. Shang, and H. Chen, “Improved Viola–Jones face detec-
tion algorithm based on Holo.ens,” EURASIP J. Image Video Process.,
vol. 2019, no. 1, p. 41, Feb. 2019.
[6] H. Lahiani, M. Kherallah, and M. Neji, “Hand pose estimation sys-
tem based on Viola-Jones algorithm for Android devices,” in Proc.
IEEE/ACS 13th Int. Conf. Comput. Syst. Appl. (AICCSA), Agadir,
Morocco, Nov. 2016, pp. 1–6.
[7] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521,
no. 7553, pp. 436–444, May 2015.
[8] S. Jain and R. Chauhan, “Recognition of handwritten digits using DNN,
CNN, and RNN,” in Proc. 2nd Int. Conf. Adv. Comput. Data Sci. (ICACDS), Dehradun, India, Apr. 2018, pp. 239–248.
[9] X. Chen, P. Wei, W. Ke, Q. Ye, and J. Jiao, “Pedestrian detection
deep convolutional neural network,” in Computer Vision-ACCV
2014 Workshops (Lecture Notes in Computer Science), vol. 9008. Cham,
Switzerland: Springer, Nov. 2014, pp. 354–365.
[10] A. R. Pathak, M. Pandey, and S. Rautaray, “Application of deep learning
for object detection,” Procedia Comput. Sci., vol. 132, pp. 1706–1717,
Jan. 2018.
[11] W. Rawat and Z. Wang, “Deep convolutional neural networks for image
classification: A comprehensive review,” Neurot Comput., vol. 29, no. 9,
p. 2352–2449, Sep. 2017.
[12] T. Liu and T. Stathaki, “Faster R-CNN for robust pedestrian detection using
semantic segmentation network,” Frontiers Neurorobots., vol. 12, p. 64,
Oct. 2018.
[13] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M.
Pietikäinen, “Deep learning for generic object detection: A survey,”
Int. J. Comput. Vis., vol. 128, no. 2, pp. 261–318, Jan. 2020.
[14] R. Rani, A. P. Singh, and R. Kumar, “Impact of reduction in descriptor size
on object detection and classification,” Multimedia Tools Appl., vol. 78,
n. 7, pp. 8965–8979, Apr. 2019.
[15] N. Ahmed, “Image clustering using exponential discriminant analysis,”
IET Comput. Vis., vol. 9, no. 1, pp. 1–12, Feb. 2015.
[16] M. Gogoi and S. A. Begum, “Image classification using deep autoen-
coders,” in Proc. IEEE Int. Conf. Comput. Intell. Comput. Res. (ICICC),
Coimbatore, India, Dec. 2017, pp. 1–5.
[17] H. Itoh, A. Imiya, and T. Sakai, “Dimension reduction and construction of
feature space for image pattern recognition,” J. Math. Imag. Vis., vol. 56,
n. 1, pp. 1–31, Feb. 2016.
[18] W. Wang, Y. Huang, Y. Wang, and L. Wang, “Generalized autoencoder:
A neural network framework for dimensionality reduction,” in Proc. IEEE
Conf. Comput. Vis. Pattern Recognit. Workshops, Columbus, OH, USA,
Jun. 2014, pp. 496–505.
[19] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detec-
tion,” in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.
(CVPR), San Diego, CA, USA, Jun. 2005, pp. 886–893.
[20] C. Kalyoncu and O. Toygar, “GTCLC: Leaf classification method using
multiple descriptors,” IET Comput. Vis., vol. 10, no. 7, pp. 700–708,
Jun. 2016.
[21] Y. Qi and G. Zhang, “Strategy of active learning support vector machine
for image retrieval,” IET Comput. Vis., vol. 10, no. 1, pp. 87–94, Feb. 2016.
[22] T. George, S. P. Potty, and E. Kim, “A novel on-road vehicle detection
method using HOG,” in Proc. Int. Conf. Control, Instrum., Commun.
Comput. Technol. (ICCICCT), Kanyakumari, India, Jul. 2014, pp. 461–465.
[23] G. Khan, A. Siddiqi, M. U. G. Khan, S. Q. Wahla, and S. Samyan,
“Geometric positions and optical flow based emotion detection using MLP
and reduced dimensions,” IET Image Process., vol. 13, no. 4, pp. 634–643,
Mar. 2017.
[24] X. Fang, W. Gu, and C. Huang, “A method of skin color identification
based on color classification,” in Proc. Int. Conf. Comput. Sci. Netw.
Technol., Harbin, China, Dec. 2011, pp. 2355–2358.
[25] E. Rachmawati, M. L. Khodra, and I. Supriana, “Histogram based color
pattern identification of multiclass fruit using feature selection,” in Proc.
Int. Conf. Electr. Eng. Informat. (ICEEI), Denpasar, Indonesia, Aug. 2015,
p. 43–48.
[26] M. Kaur and S. Dhingra, “Comparative analysis of image classification
techniques using statistical features in CBIR systems,” in Proc. Int. Conf.
I-SMAC (IOT Social, Mobile, Analytics Cloud) (I-SMAC), Palladam, India,
Feb. 2017, pp. 265–270.
[27] S. Banerji, A. Sinha, and C. Liu, “HaarHOG: Improving the HOG descrip-
tor for image classification,” in Proc. IEEE Int. Conf. Syst., Man, Cybern.,
Manchester, U.K., Oct. 2013, pp. 4276–4281.
[28] J. Kim, J. Baek, and E. Kim, “A novel on-road vehicle detection method
using π HOG,” IEEE Trans. Intell. Transp. Syst., vol. 16, no. 6,
pp. 3414–3429, Dec. 2015.
[29] M. Bilal, A. Khan, M. U. K. Khan, and C.-M. Kyung, “A low-complexity
pedestrian detection framework for smart video surveillance systems,”
IEEE Trans. Circuits Syst. Video Technol., vol. 27, no. 10, pp. 2260–2273,
Oct. 2017.
[30] Y. Liu, Y. Ge, F. Wang, Q. Liu, Y. Lei, D. Zhang, and G. Lu, “A rotation
invariant HOG descriptor for tire pattern image classification,” in Proc.
IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), Brighton, U.K.,
May 2019, pp. 2412–2416.
[31] T. Ojala, M. Pietikäinen, and T. Maenpää, “Multiresolution gray-scale
and rotation invariant texture classification with local binary patterns,” IEEE
Trans. Pattern Anal. Mach. Intell., vol. 24, no. 7, pp. 971–987, Jul. 2002.
