Deep Feature Learning for Cleanliness Classification in Restrooms

Lahiru Jayasinghe, Nipun Wijerathne, and Chau Yuen, *Senior Member, IEEE*.

Abstract—In order to revamp the cleaning contract from the head-count basis into performance basis, a fair and unbiased cleanliness classification is necessary. However, the perception of cleanliness is very subjective to the observer. Hence, it is not an easy task to quantify the cleanliness. This paper presents a deep feature learning approach using Principal Component Analysis (PCA) and Deep Convolutional Neural Networks (DCNN) for classification of the cleanliness of a restroom into three categories; namely *dirty*, *average*, and *clean*. Our method sheds new lights on data augmentation, feature extraction, and knowledge transfer between models.

Index Terms—Cleanliness classification, deep learning, convolutional neural networks, principle component analysis, data augmentation, feature learning, internet of things

1 Introduction

With the increasing of man power cost, there is a suggestion to move away from the traditional "head-count" system to a "performance-based" system for the cleaning services, i.e. instead of hiring a fixed number of cleaners, the cleaning services is charged based on the cleanliness performance. Hence, there is a need of cleanliness measuring standards, which provide precise definitions. One of such standard can be found in [1].

However, the interpretation of a cleanliness standard and the perception of cleanliness could still be subjective, and different persons could rank differently. A fully automated systems that can provide measurement on the cleanliness according to the cleaning standard, would move away the human bias or time-consuming human training. Hence, the system should be equipped with Artificial Intelligence (AI) to make an autonomous decision by providing a fair judgment about the cleanliness based on certain cleaning standard. With such system, the building managers and cleaning service providers can then focus on performance-based cleaning system rather than involving a large number of cleaners to work on restroom facilities.

In the literature, there have been a number of work related to the restroom cleanliness. A research group has proposed a solution for smart cleaning called Restroom Visitilizer System [2], which utilize people counters and odour sensors to determine the ammonia level present in restrooms. Two other research groups have proposed similar solutions, which utilize people counters and ammonia sensors to detect the dirtiness in restrooms and they have provided an end-to-end solution for the facility managers to monitor the cleanliness in their restrooms [3], [4].

To this end, the previous work focused on the odour based sensor systems to detect the dirtiness, but there could be other odourless dirtiness, which can appear in a restroom. For instance, tissue papers often spread all over the restroom floor and sink mirrors, taps, knobs and other hand touching places get dirty very easily. Odour based sensor systems need a considerable number of sensors to cover the whole restroom, which can be sometimes inconvenient. Furthermore, according to the implementers of [3], limited interpretability of existing odour sensors and complex correlations among readings from various sensors deployed in different facilities could be a challenge.

In this paper, we focus on the restroom cleanliness, we aim to design an AI driven architecture, that can access the restroom cleanliness, based on certain stardard, e.g [1].This paper presents a vision based deep feature learning schema for classification of odourless dirtiness in restrooms. We employed deep convolutional neural networks for feature extraction, principal component analysis for feature analysis, and a multilayer perceptron neural network (MLP-NN) as the classifier for cleanliness classification. The aim of our study is to implement a robust model, which is capable of classifying the cleanliness of a specific restroom by analyzing image features. Our dataset comprises images of urinal bowls that are captured by a normal smart phone camera. The data labeling is provided by professional building managers, where they have labeled the images into three categories, namely *clean*, *average*, and *dirty*.

The main challenges we encountered are as follows.

1) There are imbalanced data set across different category, especially in the dirty case, which it is impossible to cover all potential dirty scenarios.
2) There are wide variations in terms of scaling, camera angles, lighting conditions, and shadowing of images.
3) There is only limited dataset available for training.
4) It is not known if the model trained from one restroom can be applied for another restroom.

In order to address first and second challenges, we introduce data augmentation techniques on the collected data to expand the dataset size. Data augmentation techniques include scaling, random cropping, histogram equalizations, and color augmentation. Specifically, a novel PCA based color augmentation method is proposed in order to enhance the dirtiness in images or to introduce new dirt in the images, which can be beneficial in the latter stages of the proposed architecture. To address the third challenge, we introduced a robust PCA analysis approach to select a feature extractor capable of understanding the dirtiness features of data, and a training process to depreciate the classifier over-fit during training.

• *L. Jayasinghe, N. Wijerathna, and C. Yuen are with the Singapore University of Technology and Design (e-mail: aruna_jayasinghe,gallage_wijerathne,yuenchau@sutd.edu.sg)*

Fig. 1: Data samples from each category in dataset $\boldsymbol{s}$ shown here. Images in (a), (b), and (c) represent data relevant to *dirty*, *average*, and *clean* categories respectively.

Regarding the fourth challenge on generalization, we performed a brief experiment using our trained model with some urinal bowl images taken from other restrooms, the results showed that our proposed model can still accurately perform the classification without much over-fitted. However, such generalization deserves in-depth study, which will be treated as a future work.

The contributions of this paper can be summarized as follows:

- Design a novel color augmentation approach to enhance dirtiness regions in images.
- Propose a robust analytical approach to select a feature extractor and its transfer layer that is capable of extracting meaningful features from the given dataset.
- Implement a robust classifier through a vigorous training procedure

The paper is organized as follows. State-of-the-art image feature extraction methods and classification methods are discussed in Section 2. The proposed architecture is presented in Section 3 and their results are discussed in each subsections separately. Finally, the Section 4 concludes the paper.

2 Literature Review

State-of-the-art techniques for image feature learning and how these learned features have being used in the image classification are reviewed in this section. In particular, we present three recent DCNN architectures that have been proposed for image classification.

2.1 Image Feature Learning

The most significant features of an image can be stated as its color and the texture. Usually, color information included of an image could be measured by color histograms (RGB, HSV and LAB histograms), while texture details are described using LBP histograms and Gabor filters [5]. One famous method for image feature analysis is Bag-of-feature (BoF) [6]. It contains three phases, namely; feature extraction, feature encoding, and feature pooling. The feature extractor uses SIFT [7], HOG [8], or SURF [9] techniques to extract features, while encoding is performed by sparse coding related methods such as [10], [11], and Bof feature pooling could be performed using Spatial Pyramid Matching (SPM) or max-pooling [12], [13]. Moreover, literature describe that the PCA can be used to learn robust features in an image. In [14] outlined a method of combining PCA and ICA (Independent Component Analysis) for face recognition. Authors in [15] have implemented distinctive descriptors using PCA and SIFT.

Recently, there has been much research conducted on feature learning using Convolutional Neural Networks (CNN). With the recent development in CNN, it has been commonly applied in diverse real-world applications such as medical image analysis

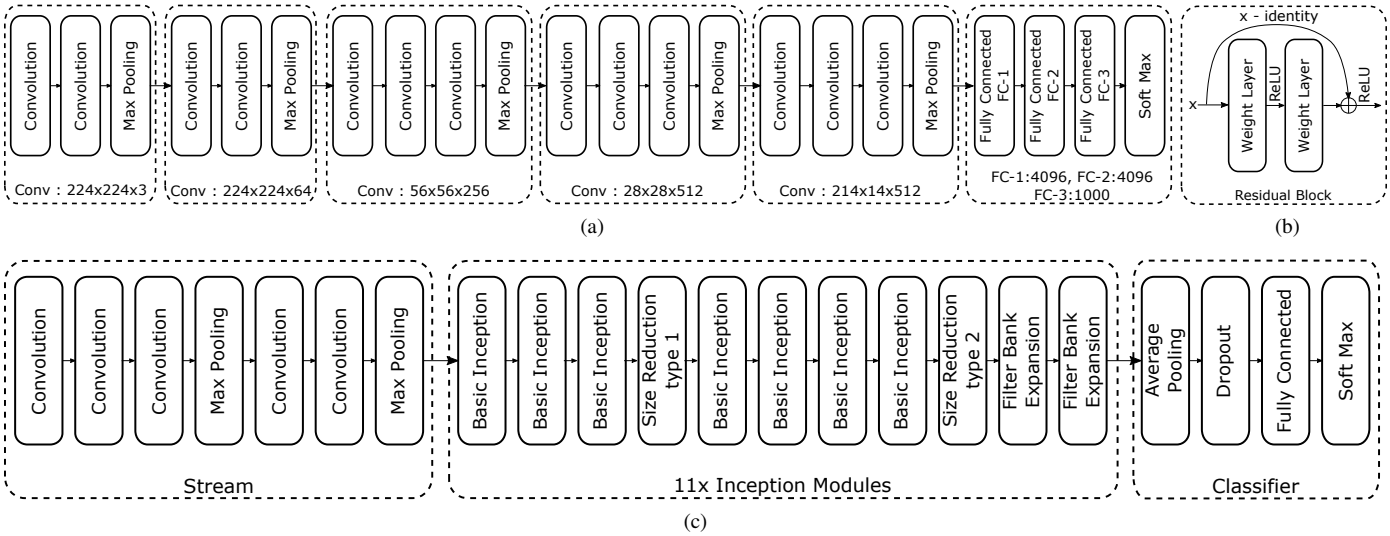


Fig. 2: State-of-the-art DCNN architectures and their building blocks are shown here. (a) VGG-16 DCNN architecture, (b) the fundamental building block of ResNet-50 DCNN architecture and (c) is the Inception-V3 DCNN architecture.

[16], time series analysis [17], speech recognition [18], etc. Generally, CNN occupies three main layers; convolutional layer, pooling layer, and fully connected layer. DCNN contains a large number of deep stacks of these layers, hence the name DCNN. It has the ability to learn meaningful features from a given dataset, while iteratively minimizing the error. Intuitively, when the number of layers grow, the performances will also improve along with it. When DCNN gets deeper, the parameter count will also increase exponentially and those parameters could be able to explain almost every important feature about the dataset. However, to extract meaning full features, DCNN needs a vast amount of sample data. The ImageNet dataset can be recognize as a large-scale dataset for image recognition [19]. Many DCNN architectures were introduced for image classification and detection using ImageNet's Large Scale Visual Recognition Challenge 2012 (ILSVRC2012) dataset. AlexNet [20], VGG [21], ResNet [22] and Inception-V3 [23] are some notable DCNN architectures trained using ImageNet dataset.

2.2 Image Classification

The most efficient method of image classification is to extract features of the images and classify them based on their features using a particular classifier. This pipeline is used in BoF and DCNN based image classifications.

BoF depicts the image as histograms of discrete quantized features and uses SVM as a classifier to perform the classification. Spatial pyramid approach introduced by Lazebnik et al. [12], which count the features inside a sub-region without focusing the image as a whole. Authors in [24] improved this classification approach by incorporating sparse coding and dictionary learning. This proposed implementation achieved the best performances in ILSVRC 2010 classification challenge. Thereafter, Perronnin et al. [25] employed fisher kernel to introduce higher order statistics for image classification and their approach achieved the state-of-the-art image classification results in ILSVRC 2011. These higher order statistics can be considered as the baseline idea for deep learning.

After ILSVRC 2011, higher order statistical approaches for image classification got attention. As a result, AlexNet was introduced [20]. AlexNet consists of five convolutional layers followed by two fully connected and a softmax output layer. This architecture achieved the top-5 error rate of 15.3% and won the ILSVRC 2012. As mentioned before, the same pipeline approach is used for AlexNet, which means extracting the features by employing a DCNN and classify them using a soft-max classifier. From this point onwards, all the winning architectures were based on DCNNs with various modifications. Introduction of a Spatial Pyramid Pooling (SPP-net [26]), batch-normalized networks, inception modules, and residual networks can be noted as some iconic modifications in DCNN architectures. Some of the state-of-the-art DCNN architectures are described below.

VGG-16

The VGG network architecture was introduced by Simonya et al. [21] for the ILSVRC 2014 challenge. This architecture uses only 3×3 convolutional layers stacked together to increase the depth, while reducing the volume size by using max pooling. Two fully-connected layers, each with 4,096 nodes are then followed by a softmax classifier which includes 1000 classes, see Fig 2a. This architecture demonstrates astonishing results in 2014, and it is frequently being used in many deep learning applications.

ResNet-50

Sometimes deep architecutres performs worse, when they become deeper. Kaiming He et al [22] have proposed a solution for this issue by introducing residual networks into DCNNs see Fig 2b. Intuitively, if a DCNN is capable of extracting meaningful features, then its accuracy should not be degraded by adding one layer of network, this is because the last layer of the original network should be able to learn an identity feature mapping to the newly added layer. However, in practice, deep neural nets suffer from a vanishing gradient problem, which means the gradient signal could go to zero quickly when back propagating through layers, while making a gradient descent unbearably slow or impossible. Bringing in the residual models, the gradient signal could travel

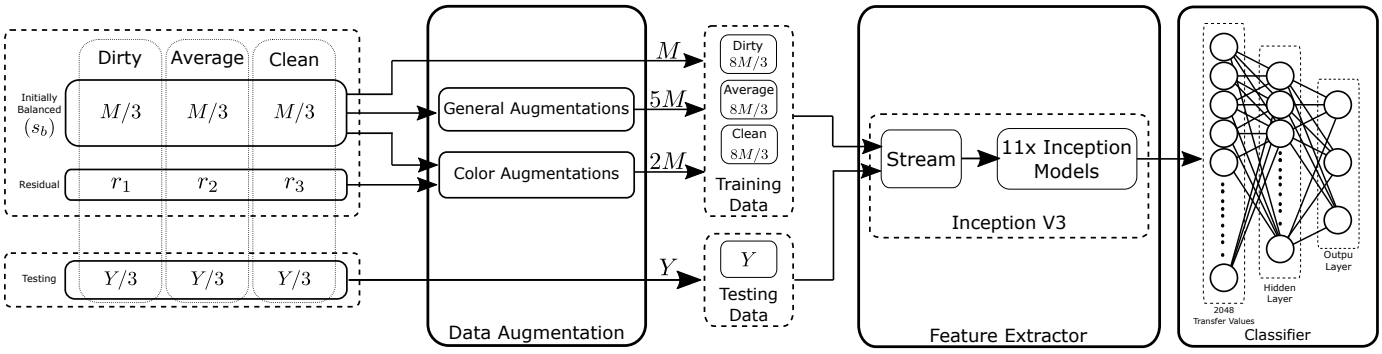


Fig. 3: The proposed architecture for the cleanliness classification.

backward via skip connections in residual nets. Therefore, one can suddenly build 50-layer or even 1,000 layer nets using residual blocks.

Inception-V3

Though the ResNet is about going deeper, the Inception architecture is all about going wider. Inception-V3 is a very deep network whose architecture is inspired by Network-in-Network concept [27] and sparse networks. The architecture consists of an image stream, eleven inception modules, and a fully connected neural network classifier, see Fig. 2c. Inception-V3 focuses on developing a deeper network without increasing much computational cost. The whole architecture contains 54 neural layers with 25 million parameters. The input image stream consists of five convolutional layers along with two max-pooling layers. The most crucial part of feature extraction in Inception-V3 model is the stack of eleven inception modules. Inception stack is made of four types of different inception modules, namely, basic inception module, size reduction module, and two types of filter bank expansion modules. The basic inception module is designed to approximate the optimal local sparse structure of features. Size reduction modules are responsible for reducing model dimension otherwise the computational requirements would be too complex. The two types of filter bank expansion modules assist to increase the dimensional representations for the soft-max classifier.

3 Proposed Architecture

As shown in Fig 3, the proposed architecture consists of data augmentation, feature extraction, and a classifier. The collected dataset is limited and unbalanced, where $M = 273$, $N = 323$, $r_1 = 4$, $r_2 = 7$, and $r_3 = 39$. Therefore, the dataset needs to be artificially enlarged by using data augmentation techniques. The proposed data augmentation increases the size of a given dataset by seven times and collectively the training data set becomes eight times larger than the initially balanced dataset see Fig. 3.

The dataset consists of 4032×3024 pixels sized RGB urinal bowl images taken from a restroom facility. The image dataset is labeled by an experience building manager into three classes: namely dirty, average, and clean see Fig. 1. Since urinal bowls not frequently getting dirty, the number of images in dirty and average categories are significantly less than the images in the Clean category. In [28], the authors investigated that the algorithms trained with balanced datasets usually surpass those trained with imbalanced. Hence, we prepared an initially balanced dataset $\boldsymbol{s_b}$ from the collected data $\boldsymbol{s}$, and kept the remaining data as residuals to accommodate in color augmentation and validation tasks in the training process (explained in subsection 3.1 and 3.3). For the testing phase, we acquired a new unbiased dataset from the same restroom facility environment to ensure that there are no overlaps between training and testing data.

Thereafter, a pre-trained Inception-V3 [23] DCNN model was selected to extract features from the data. Subsection 3.2 describe the selection procedure of the Inception-V3 deep feature extractor. Finally, the extracted features are classified by using a three-layered MLP-Neural network classifier. The proposed training, and testing methods for the classifier are explained in the subsection 3.3.

3.1 Data augmentation

Though there are many techniques to generate data, a mere and plausible way could be random crops. Since the dataset consists of rather high-resolution images, the cropped frame needs to incorporate a considerable part of the urinal bowl. Because of that, we included 50% and 75% crops from the original image, which leads to 2016×1512 and 3024×2268 pixel resolutions respectively.

According to the Fig. 1, it can be clearly see that the collected dataset is not uniform. This means that they accommodate a variety of inconsistencies between data due to the variations of camera angles and lighting conditions. For instance, images show various scales of urinal bowl sizes, some images are flipped compared to others and lighting conditions are fairly inconsistent between data.

Furthermore, these disparities could influence the decision-making process. Subsequently, the classifier will learn these differences as the features for classification rather than learning the dirtiness feature included in the data, this phenomena is know as over-fit in deep learning. Thus, the solution would be to eliminate these dissimilarities by introducing additional data with the same types of disparities, then the classifier can be trained to be robust to variants during implementation. Therefore, we apply random scaling, horizontal flipping, and histogram equalizations. These techniques are referred as general augmentations in Fig 3. Implementations have been done using the libraries opencv-python-3.4 and TensorFlow.

The other form of data augmentation consists of altering the intensities of RGB channels probably around the dirty regions (outliers). The motivation is to increase the dirtiness by enhancing the pixels around the dirty region, see Table. 1. It may or may not be visible to the naked eye, but numerically it is clearly visible to the learning model.

Algorithm 1 Color Augmentation.

Input: $\boldsymbol{s} = [S^1, \ldots, S^N] \ \forall S^i \in \mathbb{R}^{W \times H \times 3}, i \in \{1, \ldots, N\}$
Input: $I_{W \times H \times 3} \in \boldsymbol{s_b} = \left[S_b^1, \ldots, S_b^M\right] \ and \ \boldsymbol{s_b} \subset \boldsymbol{s_b}$
Output: Im_1, Im_2

1: $Initialisation : \boldsymbol{r_0}, \boldsymbol{g_0}, \boldsymbol{b_0} \ \rightarrow \ 0$
2: $\boldsymbol{s} = \left[\left[(s^1)_{w,h,c}\right], ..., \left[(s^N)_{w,h,c}\right]\right] \ \forall \ w \in \{1, ..., W\}, h \in \{1, ..., H\}, c \in \{1, 2, 3\}$
3: **for** $i = 1$ to N **do**
4: $\quad S' = resize(S^i)$
$\quad$ *Now* $W \rightarrow W', H \rightarrow H' \therefore S' = [(s')_{w',h',c}]_{W' \times H' \times 3}$
5: $\quad \boldsymbol{r_i} = vec([(s')_{w',h',1}])_{W'H' \times 1} \parallel \boldsymbol{r_{i-1}}$
6: $\quad \boldsymbol{g_i} = vec([(s')_{w',h',2}])_{W'H' \times 1} \parallel \boldsymbol{g_{i-1}}$
7: $\quad \boldsymbol{b_i} = vec([(s')_{w',h',3}])_{W'H' \times 1} \parallel \boldsymbol{b_{i-1}}$
8: **end for**
9: $\boldsymbol{i} = [\boldsymbol{r_N}, \boldsymbol{g_N}, \boldsymbol{b_N}] \ \forall \ \boldsymbol{r_N}, \boldsymbol{g_N}, \boldsymbol{b_N} \in \mathbb{R}^{W'H'N \times 1}$
10: *Calculating mean :* $\bar{\boldsymbol{i}} = [\overline{\boldsymbol{r_N}}, \overline{\boldsymbol{g_N}}, \overline{\boldsymbol{b_N}}] \ and \ \boldsymbol{i_m} = \boldsymbol{i} - \bar{\boldsymbol{i}}$
11: *Performing PCA :* $[\boldsymbol{\lambda}, \boldsymbol{u}] = PCA(\boldsymbol{i_m})$
$\quad$ *Now* $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \lambda_3]_{1 \times 3}, \boldsymbol{u} = [\boldsymbol{u_1}, \boldsymbol{u_2}, \boldsymbol{u_3}]_{3 \times 3}$
12: $\boldsymbol{\lambda_s} = [\alpha_1\lambda_1, \alpha_2\lambda_2, \alpha_3\lambda_3] \ \forall \alpha_i \sim \mathcal{N}(\mu, \sigma^2)$
13: $\hat{I} = Normalization(I) \ and \ \boldsymbol{v} = \boldsymbol{u}\boldsymbol{\lambda_s}^\top$
$\quad$ *Defined :* $\hat{\boldsymbol{I}} = \left[\hat{I}_r, \hat{I}_g, \hat{I}_b\right] \ \forall \hat{I}_r, \hat{I}_g, \hat{I}_b \in \mathbb{R}^{W \times H}, \ \boldsymbol{v} = [v_1, v_2, v_3]^\top \ and I = [I_r, I_g, I_b]$
14: $\boldsymbol{I^m} = \left[\hat{I}_r + v_1, \hat{I}_g + v_2, \hat{I}_b + v_3\right]$
15: $Im_1 = I + \beta I^m \ \forall \beta \sim \mathcal{N}(\mu, \sigma^2)$
16: $I_{BW} = \gamma[BinaryThresholding(Normalization(\beta \boldsymbol{I^m}))]$
17: $Im_2 = \left[\hat{I}_r - I_{BW}, \hat{I}_g - I_{BW}, \hat{I}_b - I_{BW}\right]$
18: **return** Im_1, Im_2

The color augmentation is shown in Algorithm 1, where it needs collected data $\boldsymbol{s} = [S^1, \ldots, S^N] \ \forall S^i \in \mathbb{R}^{W \times H \times 3}, i \in \{1, \ldots, N\}$, where N is the size of collected dataset and W, H are scalars indicating the hight and width of a single RGB image. The next input is the image $I_{W \times H \times 3}$ from the initially balanced dataset $\boldsymbol{s_b} = \left[S_b^1, \ldots, S_b^M\right]$, where M represents the number of data contains in $\boldsymbol{s_b}$, $\boldsymbol{s_b} \subset \boldsymbol{s}$. The i^{th} image S^i in the $\boldsymbol{s}$ dataset denoted as $\left[(s^i)_{w,h,c}\right]$ matrix where $(s^i)_{w,h,c}$ represents the pixel element at w, h, c position, $\left[(s^i)_{w,h,1}\right], \left[(s^i)_{w,h,2}\right], \left[(s^i)_{w,h,3}\right]$ denoted as red, green and blue channels of the image S^i respectively, $w \in \{1, ..., W\}, h \in \{1, ..., H\}, c \in \{1, 2, 3\}$. Due to the high-resolution (W and H) of a single image $S^i \in \boldsymbol{s}$, a resizing operation performed using first order spline interpolation as depicted in,

$$S' = resize(S^i) \tag{1}$$

. This helps to increase the computational efficiency by reducing the image S^i dimensions to W' and H', hence downscale image denoted as $S' \in \mathbb{R}^{W' \times H' \times 3}$ and $W' \leq W, \ H' \leq H$. Since c represents red(r), green(g) and blue(b) channels, we perform matrix vectorization on the channel sliced two dimensional matrix of resized image S' according to

$$\begin{aligned} vec([(s')_{w',h',c}]) = [&s'_{1,1,c}, \ldots, s'_{W',1,c}, s'_{1,2,c}, \ldots, \\ &s'_{W',2,c}, \ldots, s'_{1,H',c}, \ldots, s'_{W',H',c}]^\top \end{aligned} \tag{2}$$

and as depicted in Algorithm 1, line 5 to 7 concatenate the vectorized result with the previous image S^{i-1} results, according to the color channel c respectively. The motivation of this adjustment is to obtain eigenvalues λ_1,λ_2,λ_3 and eigenvectors $\boldsymbol{u}$ that describes details about the feature distribution of the complete dataset $\boldsymbol{s}$, and use these values in the color augmentation of the image I. The $\boldsymbol{\lambda}$ and $\boldsymbol{u}$ are calculated by performing the PCA [29] on mean centered concatenated values, considering them as a three columned matrix consist of $W'H'N \times 3$ dimensions (see Algorithm 1 lines 9 to 11). Similarly to [20] we added α_i scaler from a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ and obtained $\boldsymbol{u}\boldsymbol{\lambda}$ product values $\boldsymbol{v}$. We empirically decided $\mu = 0$ and $\sigma = 0.15$ based on the dataset and results. Color channels wise image normalization is applied to the input image I according to

$$\hat{I}_c = (I_c - I_{c_{Min}}) \frac{Max_{new} - Min_{new}}{I_{c_{Max}} - I_{c_{Min}}} \tag{3}$$

where I_c denotes the c^{th} color channel of the given image I, $I_{c_{Min}}$ represents the minimum value of the c^{th} channel, maximum value represents $I_{c_{Max}}$, Max_{new} and Min_{new} are the maximum and minimum values of the expected normalized image $\hat{I}$. In this work, we used $Max_{new} = 1$ and $Min_{new} = 0$ as normalization boundaries. Next, perform a column wise addition of $\hat{I}$ and $\boldsymbol{v}^\top$. The result I^m is multiply by a constant β, extracted from a standard normal distribution with a purpose of providing a randomness and to control the intensity of color augmentation. This color augmented matrix βI^m (shown in Table. 1-col 2), is added to the given image I (shown in Table. 1-col 1) and results depicted as I_{m_1} (shown in Table. 1-col 3). Thereafter, βI^m normalized back to the 0 to 255 range, which is the input image I value range using eq. 3 and then converted it into a binary image I_{BW}, see Table. 1-col 4. The control variable γ is between 0 and 1. The binary thresholding matrix, I_{BW} is then subtracted from the channels of the given input image I, the results stored as I_{m_2}, as shown in Table. 1-col 5.

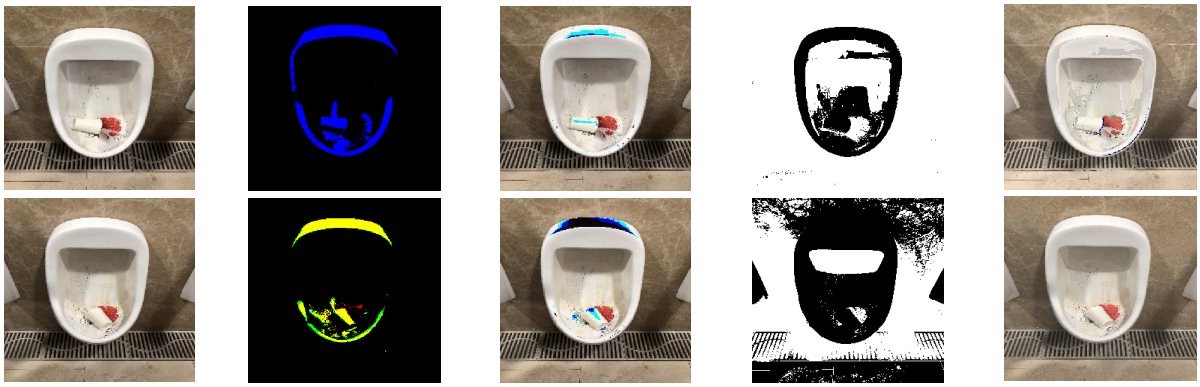
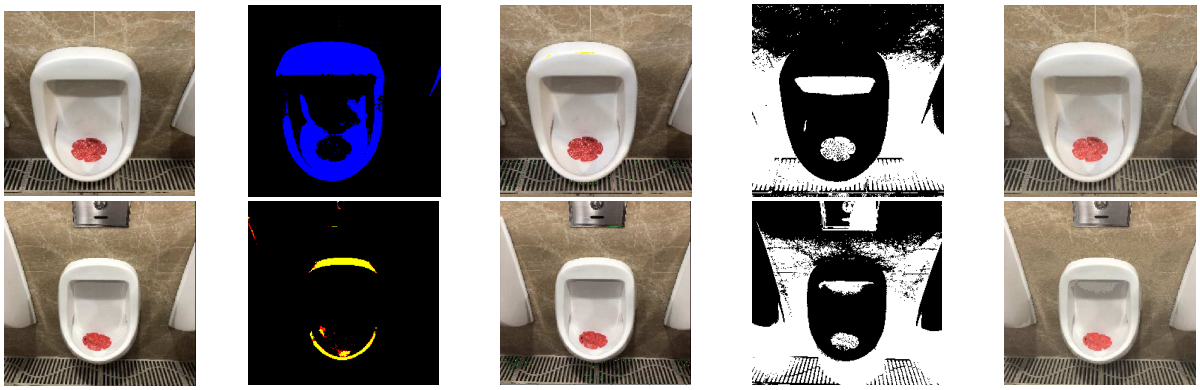
Results and Discussion

Table. 1 show the color augmentation results with respect to the dirty, average, and clean categories. It's evident after observing the color augmentation matrix images, that the proposed algorithm can identify the dirtiness (outliers) exist in urinal bowl. Examining the color augmented images closely, it's apparently visible that the color of the dirty part of images has been enhanced. In the clean category, there hasn't been too much change because it does not contain any dirtiness (outliers). But if we look closely at the color augmented image in the clean category, some of the sewer line pixels have been augmented, but the overall image remains almost same as the original image I. The binary thresholding matrix I_{BW} is able to distinguish the difference between dirty and non-dirty pixels clearly. For instance, the binary thresholding matrices of the average category were able to identify the dirtiness very precisely when compared to color augmented matrices of them. Noise suppressed images I_{m_2} were produced by subtracting the binary thresholding matrix I_{BW} by the original image I. Upon comparing the noise-suppressed image and the original image, its clearly visible that the details of the non-dirty parts are suppressed, even the wall textures and shadows are suppressed to some extent.

Since we have a biased dataset $\boldsymbol{s}$ towards clean data, the PCA act as an outliers (dirtiness) detector. This phenomenon is used in the above method. Therefore, the addition of βI^m and the input image I will enhance the pixels around dirtiness, while subtraction of I_{BW} from I will increase the attention to dirtiness. This could also be considered as a noise suppression.

A method of balancing an unbalanced dataset without wasting labeled data and a solution to the problem of limited sized datasets

TABLE 1: Color augmentation results for three categories.

Category	Original image: I	Color augmentation matrix: βI^m	Color augmented image: I_{m_1}	Binary thresholding matrix: I_{BW}	Noise suppressed image: I_{m_2}
Dirty					
Average					
Clean					

has been proposed in this section. Next subsection focuses on a method of selecting a suitable DCNN model to extract features.

3.2 Feature extraction

Over the years, many DCNNs were introduced for image classification task, and all these models were proven to give an acceptable range of image classification accuracy. Even though these deep architectures perform well in their tested datasets, training such an architecture from scratch requires millions of images and significant computational power. Since our dataset is limited, a suitable pre-trained model needed to be selected in order to fine-tune it using the training dataset. Then arise a question, how to select a pre-trained model suitable for a given dataset. In this section, we propose a novel solution to this challenge, which utilizes a visualization technique in assisting us to determine the model that is best suitable for our dataset.

Most of the deep architectures designed and trained to tackle ImageNet's dataset [19]. It contains 1.2 million images in 1,000 categories, but our specific categories are not one of them. However, since these deep architectures have been well studied in the literature and with a proven track record, it will be effective if we can leverage on these deep architectures in solving our problem. We have addressed this problem by exploiting PCA on the extracted features from unique models and different hidden layers of those models. We used VGG-16, ResNet-50, and Inception-V3 as the pre-trained DCNN models for our experiment.

According to the Fig. 5, whole dataset is used in the evaluation process. We removed some specific layers from the bottom (the layer associated with soft-max classifier) to the top (input

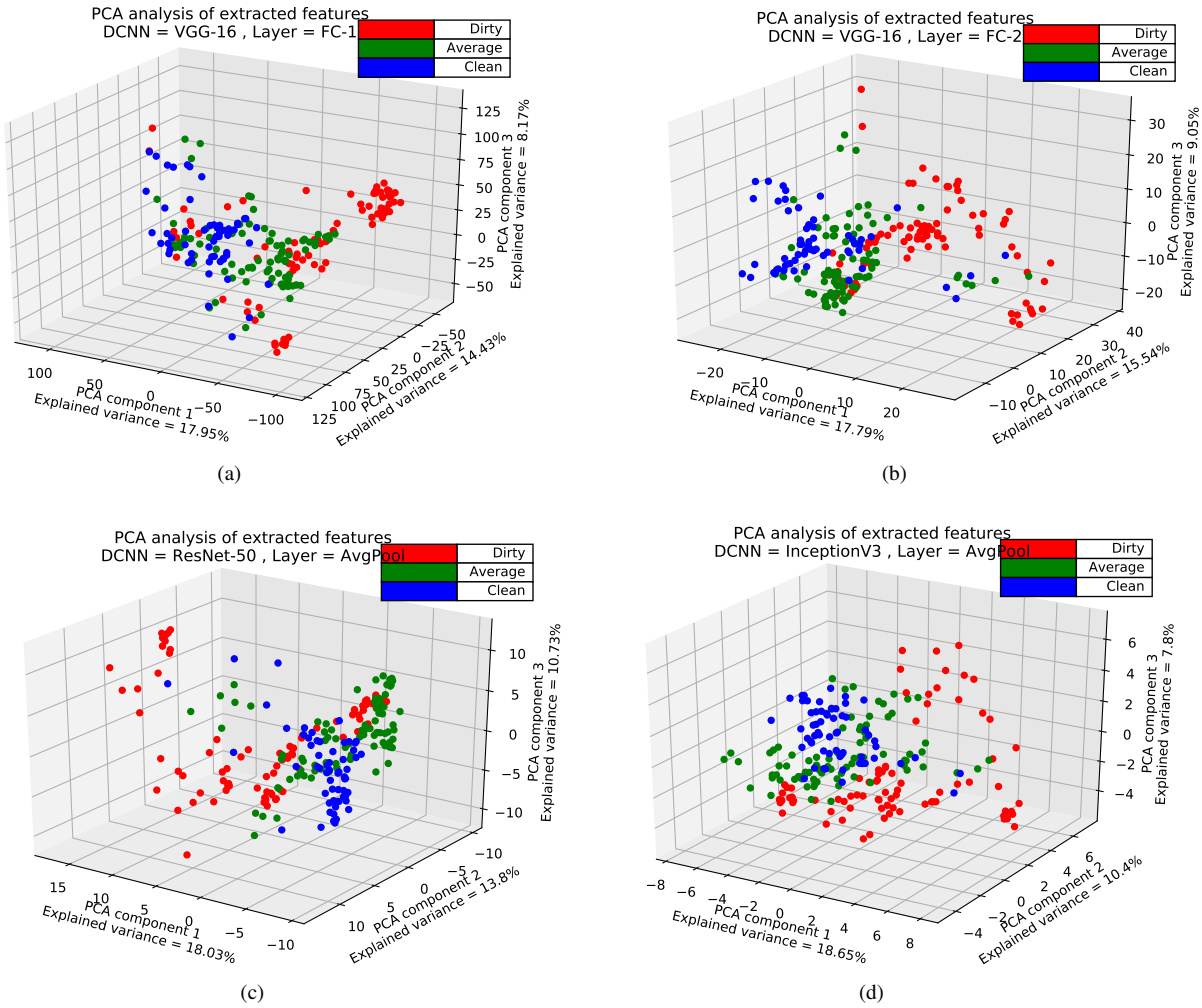


Fig. 4: PCA cluster analysis for feature extractors are presented. (a), (b) results relate to VGG-16 feature extractors and (c), (d) results relate to ResNet-50 and Inception-V3 feature extractors respectively

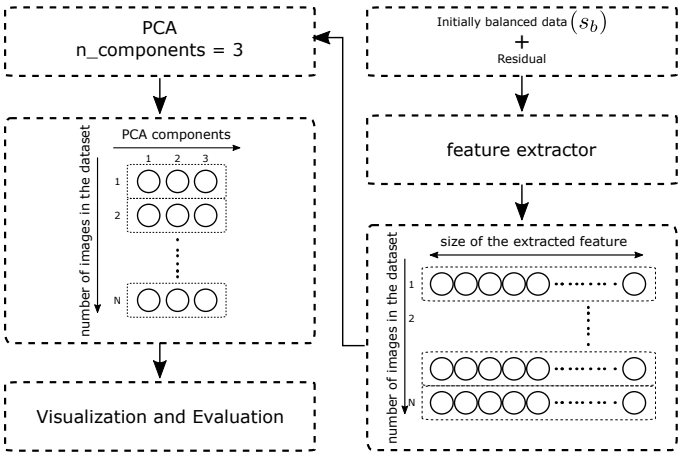


Fig. 5: The proposed evaluation process of DCNN feature extractors.

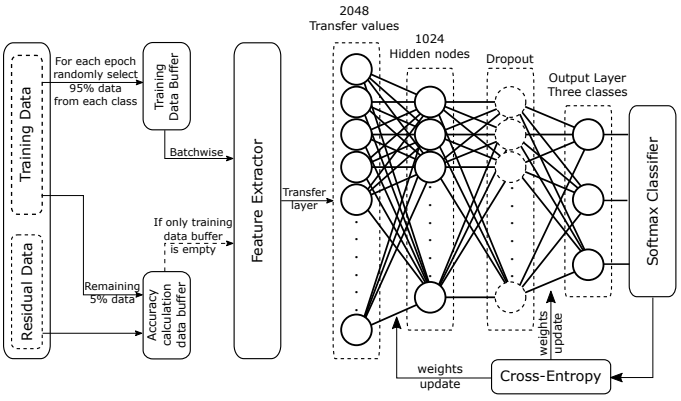


Fig. 6: Proposed classifier and its training procedure is shown here. At the end of each epoch, training accuracy is calculated using reserved accuracy calculation data. End of an epoch is marked by when training data buffer becomes empty.

layer) in the pre-trained models and created four different feature extractors. The first feature extractor is created by removing the last two fully connected layers (FC-3 and FC-2) from the VGG-16 and left it only with one fully connected layer (FC-1). Second we used the same VGG-16 and removed just the last fully connected layer (FC-3). Finally in both ResNet-50 and Inception-V3, we

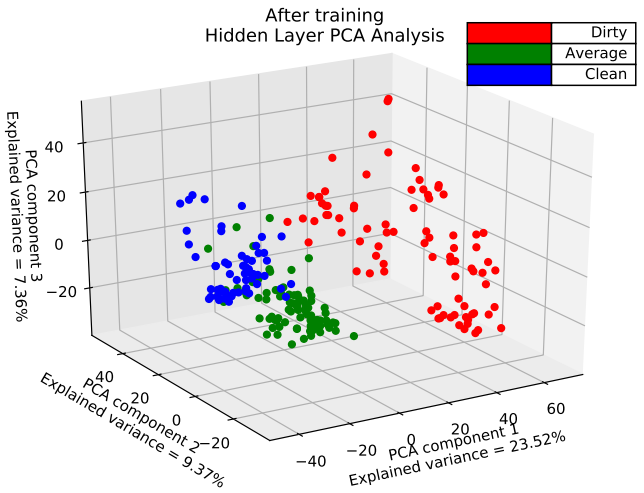


Fig. 7: PCA cluster analysis for extracted features of the classifier hidden layer.

TABLE 2: Confusion Matrix for the Classifier Results.

	Dirty	Average	Clean
Dirty	**0.99**	0.00	0.08
Average	0.02	**0.83**	0.15
Clean	0.00	0.04	**0.95**

dropped layers up to the latter average pooling layer and created the third and fourth feature extractors respectively. Features are extracted by employing one of the feature extractors mentioned above, and perform the PCA calculation using extracted features up to three components. Thereafter, features are visualized on the calculated principal component space. This process repeated for all four feature extractors.

Results and Discussion

Since PCA can be considered as a clustering method [30], the results from a compatible feature extractor (model) need to demonstrate considerable clusters in PCA space. If PCA results are not showing any form of a cluster or they are completely random, then feature extractor has failed in extracting features of the given dataset. Therefore, no use of utilizing it as a feature extractor as it only introduces a randomness, which can also achieve by a random weight initializer.

The results for each four feature extractors are depicted in Fig. 4. After a thorough observation, we can observe that, except Inception-V3 feature extractor results as shown in Fig. 4d, all others have either mixed extracted features or features related to dirty category be randomly distributed throughout the space. However, Inception-V3 extracted features related to the dirty category are almost completely separated from the features of clean and average categories. This indicates that Inception-V3 feature extractor can effectively differentiate dirty and non-dirty features as compared with the other three feature extractors. We selected the Inception-V3 up to its average pooling layer as our DCNN feature extractor.

The typical method is to build four different end-to-end solutions using the four different feature extractors with the existing classifier or employing a new classifier and compare the results at the end, which probably consume lots of resources and computational time. Whereas, using our method Fig 5, we were able to select the most qualified feature extractor for our dataset at the middle of the process and saved lot of computational time and effort. Next section focuses on classifying the data, utilizing the Inception-V3 average pooling layer result as the extracted feature.

3.3 The Classifier and Training Procedure

In general, when there is a large training dataset, latter layers of the DCNN pre-trained models could be fine-tuned without much over-fitting [31]. Since we have only a limited dataset, it's not plausible to fine-tune existing layers without over-fitting, instead, we used only the pre-trained Inception-V3 DCNN feature extractor as mentioned in subsection 3.2, and introduced a separate MLP-Neural network classifier to perform the classification task. The numerical values of the flattened average pooling layer of the Inception-V3 feature extractor is taken as the input of the proposed classifier. As depicted in Fig. 6, the flattened average pooling layer is denoted as *transfer layer* and its numerical values are *transfer values*. However, extracted features or transfer values haven't demonstrated the expected clusters in the PCA space, hence the model requires further target specific feature extraction [31]. Therefore, we used another hidden layer with dropout regularization before the soft-max classifier layer see Fig. 6.

During the training stage, 95% of training data from each category buffered in a *training data buffer*, the remaining 5% and the residual data loaded into a *accuracy calculation data buffer*. This data buffering always occurs before every epoch. In Fig. 6, at the end of every epoch, the accuracy is calculated using the reserved data in accuracy calculation buffer. Completion of an epoch is denoted when the training data buffer becomes empty. The focus of this arrangement is to enforce the model, not to learn noisy features like backgrounds, translations, etc. The training data buffer feeds the data batch wise, then the soft-max with cross-entropy loss is calculated and minimized using Adam-optimizer in the training process. We used polynomial decaying learning rate for our experiment, starting from 0.1 learning rate to 0.0001 learning rate within 100000. After model started to show compelling results, we terminated the training process and employed the test data Y to evaluate the proposed architecture.

Results and discussion

After training the classifier using extracted features from Inception-V3 feature extractor, PCA cluster analysis has been performed on the extracted features from the classifier hidden layer Fig. 7. After comparing Fig. 4d and Fig. 7, it's clear that the proposed classifier has been succeeded in extracting unique features of training data and distinguish between them.

Our experiment setups are implemented using TensorFlow framework [32] and python 3.5, on an HP Z840 workstation with NVIDIA GeForce GTX 1080Ti graphics card. The size of the collected dataset $\boldsymbol{s}$ is $N = 323$ and the size of the initially balanced dataset $\boldsymbol{s_b}$ is $M = 273$. Even though an initial dataset is created, deep learning techniques could not be applied, because the dataset is significantly very small. However, proposed architecture introduces data augmentation techniques, therefore, using data augmentation and eq. 4, we were able to enlarge the dataset

$$Dataset_i = Dataset_{i-1} + DataAugmentation(Dataset_{i-1}) \quad (4)$$

TABLE 3: The results of the trained model adaptability without fine-tuning it on other restroom images. Each column represents the classifier output probabilities for the given input image, and the class relevant to the highest probability is taken as the category for the input image.

						
Dirty	0.923	0.981	0.435	0.172	0.482	0.00
Average	0.002	0.003	0.005	0.009	0.517	0.001
Clean	0.074	0.014	0.559	0.733	0.007	0.997
Result	Dirty (Accurate)	Dirty (Accurate)	Clean (Inaccurate)	Clean (Accurate)	Average (Accurate)	Clean (Accurate)

where $i \in \{1, \ldots, 4\}$, when $i = 0$ $Dataset_0$ represent the initial balanced dataset $\boldsymbol{s_b}$ and $DataAugmentation$ represent the proposed data augmentation as a function, which enlarge a given dataset by eight times.

Table 2 shows the classification performances of our proposed classifier. Confusion matrix (CM) calculated as

$$CM(i,j) = \frac{1}{|Y_{class_i}|} \sum\nolimits_{y \in Y_{class_i}} P(class(\hat{y}) = class_j | y) \tag{5}$$

where $CM(i,j)$ denotes the prediction of j^{th} class when an i^{th} class element given, y represent an element of i^{th} class and $class(\hat{y})$ denotes the predicted class label. For some data, the classifier yields accurate class, but with a low probability, therefore even the results are accurate, the decision uncertainty could be high. Evaluation performances using Eq. 5 will reflect those uncertainties in the results. According to Table 2, it's clear that the average category has the lowest prediction accuracy. This result can be validated using Fig. 7, some features of average category seems to merge with the clean category features as well as the dirty category boundary.

Next, we gathered images from some other restroom facility and some online urinal bowl images as our new test dataset Y. Then we utilized this new test dataset to check our trained model adaptability to a new restroom environment. Table 3 shows the results reported for various types of restroom conditions and urinal bowl shapes. After observing the images closely, the dirty images were identified except the third image, but the inaccurate result doesn't have a higher prediction value. Even though these images contain various types of backgrounds, shapes, lighting conditions, elevations, etc., our proposed architecture was able to classify them accurately up to an expectable level. This result verifies that our classifier is not much over-fit, and our architecture has the potential to adapt into a new environment. However, further study is required to ensure such robustness.

4 Conclusion

In summary, this paper has developed a novel solution for cleanliness classification in active restroom facilities by learning deep features using the proposed architecture. We have devised methodologies to enhance the dirtiness by color augmentation, a method to identify a qualified DCNN model and transfer layer for feature extraction, and a competent classifier implementation along with its training procedure, which can classify an application specific dataset up to an acceptable accuracy. Finally, we have verified that our classifier is not much over-fit to the training data and our architecture has the ability to perform in different other restrooms environments. The limitations of this method could be highlighted as the difficulty of collecting data, sometimes cameras could have blind spots, image quality depends on lighting conditions and camera quality. However, despite these limitations, our approach demonstrated acceptable results. Therefore, as our future work, these limitations will be investigated and further experiments will be carried out to evaluate the model adaptability in various types of restrooms.

References

[1] National Environment Agency, "Cleaning service industry Cleaning performance for commercial premises SS499." [online]. Available: http://www.nea.gov.sg/public-health/public-cleanliness/cleaning-industry. [Accessed on 19 February 2018].

[2] "Restroom visitlizer system." [online]. Available: http://smarttechnologies.sg/. [Accessed on 19 February 2018].

[3] "Cleaner, smarter toilets for your employees and visitors." [online]. Available: https://www.ncs.com.sg/documents/20184/73669/NCS+Smart+Toilet+Analytics.pdf/ab34c9ab-5e0f-4fe5-b794-fcd6546e4f24. [Accessed on 19 February 2018].

[4] SmartClean, "Virtual cleaning supervisor." [online]. Available: http://www.smartclean.sg/. [Accessed on 19 February 2018].

[5] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[6] J. Yang, Y.-G. Jiang, A. G. Hauptmann, and C.-W. Ngo, "Evaluating bag-of-visual-words representations in scene classification," in *Proceedings of the international workshop on Workshop on multimedia information retrieval*, pp. 197–206, ACM, 2007.

[7] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[8] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893, IEEE, 2005.

[9] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[10] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient sparse coding algorithms," in *Advances in neural information processing systems*, pp. 801–808, 2007.

[11] S. Gao, I. W.-H. Tsang, and L.-T. Chia, "Kernel sparse representation for image classification and face recognition," in *European Conference on Computer Vision*, pp. 1–14, Springer, 2010.

[12] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, vol. 2, pp. 2169–2178, IEEE, 2006.

[13] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce, "Learning mid-level features for recognition," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 2559–2566, IEEE, 2010.

[14] B. A. Draper, K. Baek, M. S. Bartlett, and J. R. Beveridge, "Recognizing faces with pca and ica," *Computer vision and image understanding*, vol. 91, no. 1-2, pp. 115–137, 2003.

[15] Y. Ke and R. Sukthankar, "Pca-sift: A more distinctive representation for local image descriptors," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 2, pp. II–II, IEEE, 2004.

[16] A. Rakhlin and et al, "Deep convolutional neural networks for breast cancer histology image analysis," *arXiv preprint arXiv:1802.00752*, 2018.

[17] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[18] L. Deng and et al, "Recent advances in deep learning for speech research at microsoft," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 8604–8608, IEEE, 2013.

[19] O. Russakovsky and et al, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, pp. 1097–1105, 2012.

[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[23] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016.

[24] Y. Lin and et al, "Large-scale image classification: fast feature extraction and svm training," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 1689–1696, IEEE, 2011.

[25] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *European conference on computer vision*, pp. 143–156, Springer, 2010.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *european conference on computer vision*, pp. 346–361, Springer, 2014.

[27] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[28] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[29] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[30] S. Raychaudhuri, J. M. Stuart, and R. B. Altman, "Principal components analysis to summarize microarray experiments: application to sporulation time series," in *Biocomputing 2000*, pp. 455–466, World Scientific, 1999.

[31] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in neural information processing systems*, pp. 3320–3328, 2014.

[32] M. Abadi and et al, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.