Insight Evaluation on Traditional and CNN Features

Truong-Minh Le[1], Nguyen-Phan-Long Le[2], Nhat-Gia-Nghi Hua[2], Nhien-Loc Bui[2], and Vinh Quang Dinh[2]

[1]Faculty of Computer Science, University of Information Technology, Ho Chi Minh, Vietnam
[2]Faculty of Engineering, Vietnamese-Germany University, Binh Duong, Vietnam

***Abstract*—Feature extraction serves as the prerequisite for any intelligent-based applications. There are various methods to extract features from the initial data sets, namely traditional filters and deep learning components. However, current traditional feature extraction methods still have troubles in dealing with automatically updating the weights. This paper will propose a new feature representation method that shows more promising results than any previous methods. This proposed method investigates the benefits of features based on Convolutional Neural Network (CNN) to solve the above problems. CNN's purpose is to process multi-dimensional data, such as image and time-series data and crucial features are noticed and picked automatically without the need of being supervised by human-being. The proposed method has demonstrated significant improvement in terms of accuracy and stability compared to traditional methods of feature extraction. From that, we can easily understand why CNN-based algorithms have become a part of state-of-the-art algorithms these days.**

***Keywords*—** Convolutional Neural Network, Deep Learning, Feature Extraction, Local Patterns, Traditional Filters

I. Introduction

Computer vision applications based on videos often require the thorough feature selection from the input images as their first step. This makes selecting the correct features one of the most investigated fields in computer vision, which has led to the development of multiple feature extraction methods, with new methods coming out continuously. Recently, we have multiple options between extracting features via traditional filters or feature representation via Deep Learning neural network.[1].

One of the ongoing problems faced by any method of feature extraction is selecting features efficiently and updating weights correctly in the backward-propagation process. In this work, we leverage Local Binary Pattern[2], Local Ternary Pattern[3] and its variants to enhance robustness to illumination changes, shadows and noise, and produce more impressive results. Then, we extract features via Convolutional Neural Network and compare the final results under the same conditions.

Through our research, in different situations, diverse methods can produce good results. By combining the findings and methods of traditional filters and CNN, we have proposed our own methodology and approach to the problem. The algorithm and the results of evaluating its performance on the MNIST [4] and CIFAR-10 [5] datasets and are presented in this paper. This research presents several new and major contributions, as summarized below:

- This paper introduced the traditional powerful feature extraction methods to compare with the new method, Deep Learning, which is clearly defined in the Related Works section.
- Give a visual comparison between CNN and traditional methods on the highly trusted datasets and metrics.
- Give a clear explanation for the superior results from CNN compared to traditional methods after having experimental results gained from the experiments.

This paper argues that applying this method will generate more promising results. It firstly shows the previous works to demonstrate the background context of our research and describe the current limitations in Section II. Section III discusses the datasets and the proposed methods. Section IV presents the results collected from our proposed experiments. Finally, section V summarizes what we did, and suggest the future plan on scientific grounds.

II. Related Works

With the advent of deep learning, many authors have attempted a comparative review of deep learning versus more traditional methods over a wide range of applications. In the year 2016, Bo et al. [6] compared several methods - both deep learning-based and traditional-based - on their precision in detecting network intrusions. Experimental results show that Support Vector Machine (SVM) - Restricted Boltzmann Machine has the best performance which is approximately 75%, surpassing pure SVM and Naive Bayes by an average of 20%.

In 2017, Nikolaos et al. [7] showed that Multilayer Perceptron (MLP) can predict energy demands more accurately with higher reliability than other common traditional based approaches, namely Support Vector Machines, Gaussian Processes, Regression Trees, Ensemble Boosting and Linear Regression. In the same year of 2017, Gregory et al. [8] used Long Short Term Memory (LSTM) neural networks, Multinomial Naive Bayes, and GDA for authorship attribution and concluded that LSTM achieves the best performance with almost 90% in test accuracy.

In 2018, Adriana et al. [9] processed Romanian text using DNNs and Decision Tree coupled with MLP. Although Decision Tree came close to DNN, overall, DNN gave the best performance, especially in predicting entire words. In regard to the task of gender identification, [10] showed that the model based on fully connected layers of CNN produced the best F-score of 88%$\pm$3%, outperforming SVM, Gradient Boosting and 2 other deep learning models.

Lai et al. [11] has tried to evaluate the performance of CNN on the MNIST dataset in 2019. As opposed to our work, the traditional machine learning based method in comparison here is Support Vector Machine (SVM). Furthermore, accuracy is the only metric surveyed. Like [6] as above, [12] performed by Mohsin et al. in 2019 tested deep networks' performance in detecting anomalies for streaming data by comparing a total of 12 methods, divided into 4 categories: distance, density, kernel, and deep learning. Their analysis establishes deep learning methods' superiority in three out of four metrics, suffering from only high inference time.

Sucheta et al. [13] compared 4 different methods, including CNN and a CNN-based hybrid model in how well they can predict cognitive performance from MRI lesion images. In terms of predictive accuracy, the hybrid model ranked the highest overall with r2 value of 0.675, compared to CNN with just 0.627. However, CNN is better at predicting language deficit severity. In 2020, Shaik et al. [14] has carried out a very comprehensive review of two traditional methods vs three different deep learning networks (coupled with three different classifiers) over their accuracy in detecting facial expressions of the elderly. Amongst the deep learning networks, VGG-16 using an RF classifier achieves the highest accuracy, with 84% on the CIFE data set. Experiments also showed that said deep architecture outperforms other traditional methods.

In 2021, Lewis et al. [15] provided a comparative analysis of non-sequential machine learning models, sequential deep learning

TABLE I
PARTITIONS OF THE MNIST AND CIFAR-10 DATASETS

	Number of samples	
	MNIST Dataset	**CIFAR-10 Dataset**
Training set	60000	50000
Test set	10000	10000

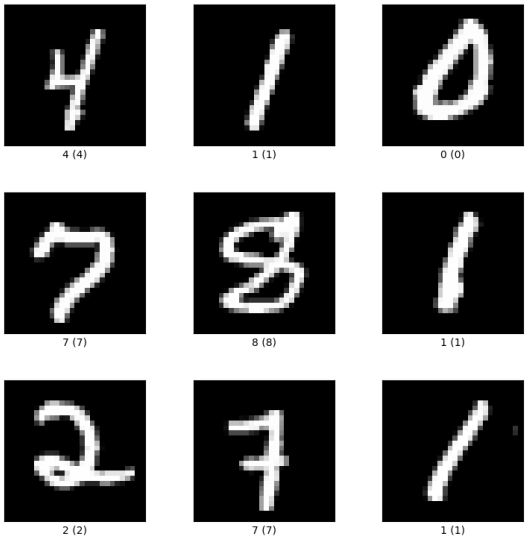


Fig. 1. Sample images belonging to MNIST dataset

models and logistic regression models over how well they can predict preventable utilization for heart failure patients. Results showed a close win of deep learning models over non-sequential ML models, both outperformed logistic regression by a wide margin. Right in the year of 2021, Abdullah Sheneamer [16] tested deep learning's ability in filtering spam emails by comparing LSTM and CNN (with and without GloVe model) and a set of 5 traditional classifiers. The analysis was carried out over 4 data sets in 3 different metrics: Accuracy, Precision and Recall. They discovered that deep learning methods outperform others by an average of 10-14%. The study also found that CNN, compared to LSTM, has better results in terms of accuracy, loss of training and validation curve.

Our work offers a comparison of multiple local patterns versus CNN by measuring their performance in two data sets - MNIST and CIFAR. In doing so, we attempt to establish CNN's superiority in both accuracy and recall levels.

III. METHODOLOGY

A. Datasets

In our current research, we utilize two well-known datasets which have been previously published, namely MNIST [4] and CIFAR-10 [5]. Both of the mentioned datasets have ten distinctive classes with different features. We highly appreciate the specificity and generality of the data sets and believe that the methods we apply can be extended to many more applicable problems in the future.

1) MNIST Dataset: The MNIST dataset is the collection of handwritten digits, having open access for everyone. It contains 70000 images whose size is 28x28 in total, containing the ten classes, ranging from 0 to 9. It has been used to benchmark the efficiency of many machine learning-based algorithms. After loading the dataset, we divide the original one into two possible partitions, namely the training set and the test set. For more details, training set comprises 60000 examples, and test set comprises 10000 examples. This dataset has been integrated into Keras [1] for efficient use. Table 1 represents how the MNIST dataset is divided for conducted experiments.

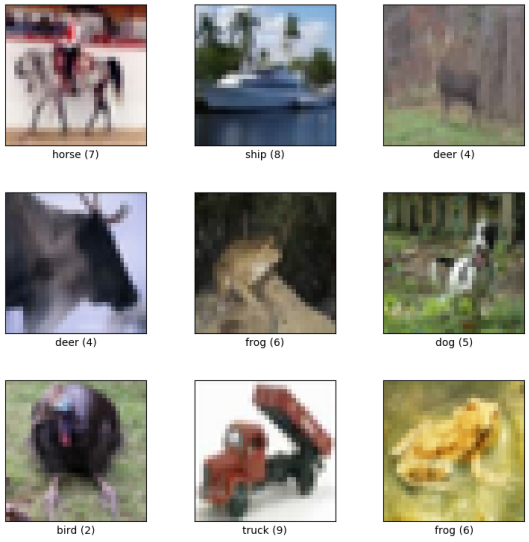


Fig. 2. Sample images belonging to CIFAR-10 dataset

2) CIFAR-10 Dataset: On the other hand, CIFAR-10 [5] is also one famous dataset whose size is 32x32 with 60000 images in total. In particular, we also split the original dataset into two subsets, including 50000 images for the training set and 10000 images for the test set. CIFAR-10 has ten specific classes, namely aeroplane, automobile, bird, car, deer, dog, frog, horse, ship, truck. Fortunately, this dataset has also been integrated into Keras [2] so that we could leverage it more effectively. Table I also represents how the CIFAR-10 dataset is divided for conducted experiments.

B. Proposed Methods

1) Feature Extractor: When dealing with practical machine learning-based algorithms, in general, we have the raw data which has not been processed yet. Transformation steps are required to get rid of noise data not contributing to the training phase and affecting negatively the final results. Then, we process and normalize the whole raw data to one consistent format, which is ready for different training purposes.

The newly transformed data must retain the general features of the initial raw data. Moreover, we need to design the typical transformations according to the properties of the problem. This important stage is called the **Feature Extraction**. After the output of Feature Extractor process, we gain the extracted features for raw input data. We use it for training deep learning-based tasks, such as classification, regression.

The important components of feature extractor are shown as below:

- Input of feature extractor component: the raw data which is all what we know about the data, including pixel values when coping with images, text length, words or sentences when coping with natural language processing, WAV-encoded audio

[1]https://keras.io/api/datasets/mnist/
[2]https://keras.io/api/datasets/cifar10/

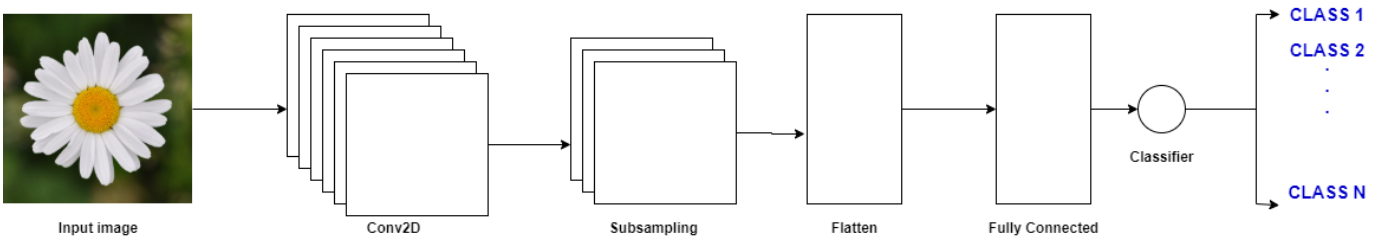


Fig. 3. General process of CNN neural network architecture

when coping with audio recognition. These types of data have different shapes and are in inconsistent formats. We cannot work with it directly but applying specific techniques to gain useful information from that.

- Initial knowledge about the selected dataset: some practical understanding of data also helps us know which type of this data. From the initial survey and particular discovery of the dataset, we are able to know the dispersion of each data point if it is in linearly separable or not.
- Output component: we only have this component in the supervised learning tasks. When it comes to the unsupervised learning tasks, the output is not defined because there are no ground truths for this type of task.

Feature extractor is used for both the training and test phases. But, it is simpler in the test phase as we can re-use the created feature extractor from the training phase - but not leverage the output as it is what we are finding - to find out the corresponding feature vectors.

2) Traditional filters: To effectively extract spatial information, we use algorithms to detect both the edge and local texture, which are figured in table II. To be clearer, for the CIFAR-10 dataset containing colour images with three channels (RGB), we use the whole 14 traditional filters to apply the feature extraction step, including two filters which define the filters according to two axes horizontal and vertical.

However, for the MNIST dataset which contains only grey-scale images, it is not the case. We get rid of two filters requiring colour channels, including SCBP and MC-LBP. From that, we combined the efficiency of 12 different traditional filters to extract features as demonstrated below:

- Sobel filter: this is an algorithm supporting edge detection in two directions: horizontal and vertical, where one filter is simply the rotation of the other filter by an angle of 90 degrees. The operator comprises two corresponding 3x3 kernels as below, in which G_x captures the horizontal changes, while G_y follows the changes in the vertical direction:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \tag{1}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \tag{2}$$

- Prewitt filter: this filter has the same idea as the Sobel filter but uses two kernels with different values in each kernel as demonstrated below:

$$G_x = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix} \tag{3}$$

$$G_y = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} \tag{4}$$

- Laplacian filter: it looks for zero values when taking the quadratic derivative of the images in the datasets. Regarding the mathematic aspect, Laplacian of an image at a point I(x, y) is calculated as below:

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \tag{5}$$

In this filter, we have one disadvantage which is not having determined accurately the specific direction of changes. Finally, the input image is filtered by using the 3x3 kernel as below:

$$\begin{bmatrix} 0 & +1 & 0 \\ +1 & -4 & +1 \\ 0 & +1 & 0 \end{bmatrix} \tag{6}$$

- Gaussian Blur filter: the final goal of this filter is to make an input image smoother. The input is an image in the dataset and it will be filtered by 3x3 kernels to get the output images. We consider the Gaussian operation in 2-dimension space with the math formula as below:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-}(\frac{x^2+y^2}{2\sigma^2}) \tag{7}$$

- Local Binary Pattern (LBP) filter: the classic approach for reflecting the texture of images is using Local Binary Patterns (LBP). This is a texture descriptor introduced as early as 1994 in [17]. LBP is constructed by comparing the centre pixel with its surrounding neighbourhood of pixels. Information extracted from LBP is invariant to local illumination variations and can handle well in changing illumination conditions. The binary string generated from LBP at a given location (x_c, y_c) can be calculated from the following formula:

$$LBP(x_c, y_c) = \sum_{p=0}^{p-1} s(i_p, i_c) 2^p, \tag{8}$$

where i_c and i_p are the central pixel and the N-neighborhood pixel gray-scale intensity respectively, function s for LBP is defined as follows:

$$s_{LBP}(i_p, i_c) = \begin{cases} 1, & if \quad i_p > i_c \\ 0, & otherwise \end{cases} \tag{9}$$

- Center-Symmetric LBP (CS-LBP) filter: an improved version of LBP as mentioned earlier in computational efficiency, first coined by [18]. Instead of comparing surrounding neighbours to the central pixel like LBP, CS-LBP provides a better interest region descriptor by comparing center-symmetric pairs of pixels, which also reduces the pixel couples compared. However, both LBP and CS-LBP binary representation can be affected heavily by noise.
- Local Ternary Pattern (LTP) filter: it is first introduced by [19], which will compare the neighboring pixels to a constant threshold, giving each pixel three possible values but not two. Because of this change, LTP is less sensitive to small noise compared to LBP. Having considered k as the threshold constant, the result of threshold is:

$$s_{LTP}(i_p, i_c) = \begin{cases} 1, if & i_p \geq i_c + k \\ 0, if & i_p > i_c - k \quad and \quad i_p < i_c + k \\ -1, if & i_p \leq i_c - k \end{cases} \tag{10}$$

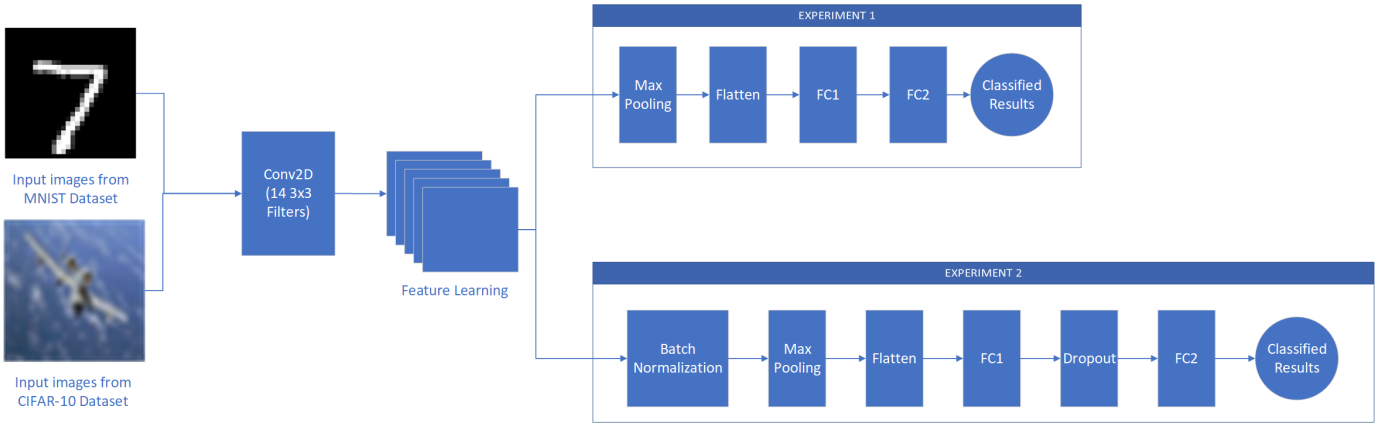


Fig. 4. Extract features via CNN filters and perform ten-class classification task on the datasets

- eXtended Center-Symmetric Local Ternary Pattern (XCS-LTP) filter: it is inspired by XCS-LBP in [20]. It is not deeply on the parameters and works as well as its original version, LTP.
- Scale Invariant Local Ternary Pattern (SILTP) filter: [21] introduced SILTP as the significantly improved version of LTP, derived from the below function:

$$s_{SILTP}(i_p, i_c) = \begin{cases} 01, if & i_p > (1+a)i_c \\ 10, if & i_p < (1-a)i_c \\ 00, if & otherwise \end{cases} \quad (11)$$

  SILTP makes use of the adaptive parameter to extract local features, depending on the grey-scale intensity of the central pixel.
- Scale-invariant Center-symmetric Local Ternary Pattern (SCS-LTP) filter: this is a center-symmetric version of SILTP, is used as a traditional feature extraction algorithm as [22]. Both LTP and SILTP have their own advantages, but they cannot handle features effectively under certain circumstances. LTP cannot handle noise under significant illumination changes, whereas, SILTP will be only effective if noise varies based on pixel intensities.
- Generalised Local Ternary Pattern (GLTP) filter: it is first coined in [14], showing the ability to resolve both problems above at the same time, which is proposed in the following function:

$$s_{GLTP}(i_p, i_c) = \begin{cases} 1, if & i_p - ai_c \geq \tau \\ 0, if & |i_p - ai_c| < \tau \\ -1, if & i_p - ai_c \leq -\tau \end{cases} \quad (12)$$

  where a is the scale factor and τ is the constant threshold.
- Completed Local Ternary Pattern (CLTP) filter: it is another version of LTP proposed in [23], achieving both the insensitiveness to noise and can perform great discriminating properties.

In general, all algorithms above can extract features only from gray-scale information. As a result, we make use of two more patterns to extract features from color intensities, namely Spatial Color Binary Pattern (SCBP) in [24] and Multi-Channel Local Binary Pattern (MC-LBP), inspired by Multi-Channel Scale Invariant Local Ternary Pattern (MC-SILTP) in [25]. MC-LBP applies original LBP respectively to red, green and blue channels of the input images, while SCBP combines three-channel intensities into gray-scale textures as the equation below:

$$\begin{aligned} SCBP(i_p, i_c) = LBP(i_p, i_c) + 2^{N+1} f(R_c, G_c|\gamma) \\ +2^{N+2} f(G_c, B_c|\gamma) + 2^{N+3} f(B_c, R_c|\gamma) \end{aligned} \quad (13)$$

$$f(a, b|\gamma) = \begin{cases} 1, if & a > \gamma b \\ 0, if & otherwise \end{cases} \quad (14)$$

where R_c, G_c, and B_c are the three-channel intensities of the central pixel, γ is the scale factor.

We limit to use the above commonly used local pattern in our experiments, although there are several variants or improvements of those local patterns such as rank transform and fuzzy local patterns [26] [27].

3) Convolutional Neural Network filters: Convolutional Neural Network (CNN) was first introduced in the paper [28] published in 1998 with the initial purpose is to resolve the variability of 2D shapes. After that, it was clearly optimized with advanced architectures by proposing the new architectures on a regular basis and now we can use it for various problems in Computer Vision with the competitively impressive results compared to other traditional machine learning-based methods. CNN is a deep learning algorithm receiving input as an image matrix and continuously converting it via hidden layers. Finally, there is an output layer at the end of the network playing the role of usual tasks such as classification.

From the currently proposed architectures, we mainly make use of ConvNet layers to exploit the computational and classification efficiency of feature extraction methods by using Convolutional layers. From that, we will acquire the objective results to surely prove that extracting features via CNN based filters will return better results than the traditional methods.

In the paper, the ConvNet layers we have used are as follows:

- Convolutional layer: This is one of the most important blocks which are built in CNN. Its idea comes from the convolution operations when sliding a predefined-size kernel (the common size is 3x3) over the input data. After multiple dot product operations, we can produce the feature maps which are the output of the layers. After this step, the height and width of the image will be decreased, but the depth of the image will increase instead. One advantage of Convolutional layer is that the spatial distribution of the input image matrix will be totally kept, therefore, the computational performance will be better.
- Pooling layer: There are many var to define pooling layer, including Max Pooling, Average Pooling. The main objective of this layer is to down-sampling the output of the image, eliminating the noise data and still keeping the main features of the images.
- Batch Normalization layer [29]: In our study, we have set up one experiment relating to the Batch Normalization layer. In practice, this layer will normalize the input image matrices, preserving the general contributions of each feature. This layer is usually used right after the Convolutional layer to improve the generalization of the network.
- Dropout layer [30]: As we have already known, overfitting is a bad issue while training a neural network. When it comes to the main idea, we will randomly drop some units in some layers along with the network trained. This will prevent effectively

overfitting by breaking some rules that the model has learnt from data previously, then it can learn data without depending heavily on the data.
- Fully-Connected layer [31]: This layer will create connections to all neurons in the adjacent layers. We usually use this layer at the end of CNN neural network architectures.

We have conducted two experiments on two datasets. Table III points out the detailed number of parameters we use for each data collection. For the first experiment, we input the images to Convolutional layers for feature extraction in case of using CNN filters or traditional layers in case of using traditional filters, followed by Max Pooling layer. Two Fully-Connected layers are applied right after flattening the whole neurons in the network. Finally, we perform the classification tasks based on each dataset. When it comes to the second experiment, we supply two additional layers, one is Batch Normalization layer after extracting the features, and another layer is Dropout layer after the first Fully-Connected layer.

C. Data Preparation

Data is the initial resource we need to face when finding a solution for any problems. For the CNN filters, we load the data sets from Keras with MNIST [3] and CIFAR-10 [4] and use it directly without storing at a local environment. However, for the traditional filters, we save the dataset at the local environment and process for each image by applying extracting features via presented traditional filters. Then, we concatenate all images along one dimension to have a consistent data source before inputting them to the networks.

D. Evaluation Metrics

Prior to our metrics used to evaluate the methods, initial understanding of these four below definitions is vital:

- True Positive (TP): An input sample is predicted by the model as POSITIVE and it is consistent with the ground truth.
- True Negative (TN): An input sample is predicted by the model as NEGATIVE and it is consistent with the ground truth.
- False Positive (FP): An input sample is predicted by the model as POSITIVE, but it is inconsistent with the ground truth. The actual ground truth is NEGATIVE.
- False Negative (FN): An input sample is predicted by the model as NEGATIVE, but it is inconsistent with the ground truth. The actual ground truth is POSITIVE.

To effectively building any systems, methods evaluating it equally to get practical insights should be defined clearly. From that, we can gain the rules and easily apply them for more general problems. In our research with the ten-class classification problem, to evaluate the performance of classification task using different filters, we use two metrics as follows:

- Accuracy score: it is the fraction between the truely correct predictions over sum of all predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{15}$$

, where TP is the number of True Positive samples, TN is the number of True Negative samples, FP is the number of False Positive samples, FN is the number of False Negative samples.
- Recall score: it is defined as the fraction between truely correct predictions over sum of all positive predictions that possibly guess.

$$Recall = \frac{TP}{TP + FP} \tag{16}$$

, where TP is the number of True Positive samples, and FP is the number of False Positive samples.

[3]https://www.tensorflow.org/api_docs/python/tf/keras/datasets/mnist/load_data

[4]https://www.tensorflow.org/api_docs/python/tf/keras/datasets/cifar10/load_data

IV. Experimental Results

A. Train Accuracy and Train Loss

Besides the previous metrics, we also examine the pattern how the model learnt from the training data in the training phase in two metrics: train loss and train accuracy as shown in these below graphs respectively.

From the above graphs, although both traditional filters and CNN filters can support the model to learn features from the inputted data correctly by minimizing the loss ratio and increasing the training accuracy after each epoch, traditional filters somehow affect negatively the training phase when doing experiments on datasets. It is easily observed train accuracy has wavered without a rule around the range from 9.5% to 10% in the first experiment when extracting features via traditional filters from the CIFAR-10 dataset.

B. Evaluation on test set

Table III shows the detailed proportions that we performed on two datasets by using the mentioned techniques. In experiment 1, performing classification task by initially extracting features from images in the MNIST dataset via CNN filters shows the accuracy score (98.95%) and recall score (98.94%) which are nearly nine times more than when using traditional filters (accuracy score is 11.35% and recall score is 10%). The same behavior occurs on the CIFAR-10 dataset. In the second experiment, the classification task performed by using CNN filters in the MNIST dataset demonstrates the accuracy score (98.83%) and recall score (98.82%) which are roughly ten times higher than using the traditional filters (9.99% and 9.89% respectively).

After all, we can see that in two experiments for all datasets, feature extraction via CNN filters always presents better results and stability compared to traditional filters.

V. Further Discussion and Conclusion

After what we have presented, we have compared the efficiency of using traditional filters and CNN filters for classification problems. Despite the complexity of datasets and models, experiments have shown that CNN filters have performed overwhelmingly, compared to their counterparts.

In regard to the reason behind the superiority of CNN, it makes use of the learnable filters to capture the relevant features from the images or videos at different levels to mimic what the human brain thinks. Although using traditional filters to detect features for Computer Vision-based tasks is famous prior to the emergence of CNN, it would become more and more cumbersome when the number of classes needed to classify increases. However, it would be the case that features from traditional filters can also work well if the problem is more simple and best-fit for them.

In the going forward research, we are firstly going to leverage more different datasets to see whether traditional filters can perform better than CNN filters under various conditions such as changes in brightness, random rotation, and also figure out the inefficiency of CNN filters when being given a limited amount of iterations. Next, improvements for our work can be implemented by applying the latest deep learning models. Besides, we also use the result from this paper to apply to the real problems to have objective views in an industrial environment.

References

[1] "Background subtraction in real applications: Challenges, current models and future directions," vol. 35, Feb. 2020.
[2] "A texture-based method for modeling the background and detecting moving objects," vol. 28, 2006.
[3] "Enhanced local texture feature sets for face recognition under difficult lighting conditions," vol. 19.
[4] "MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges."

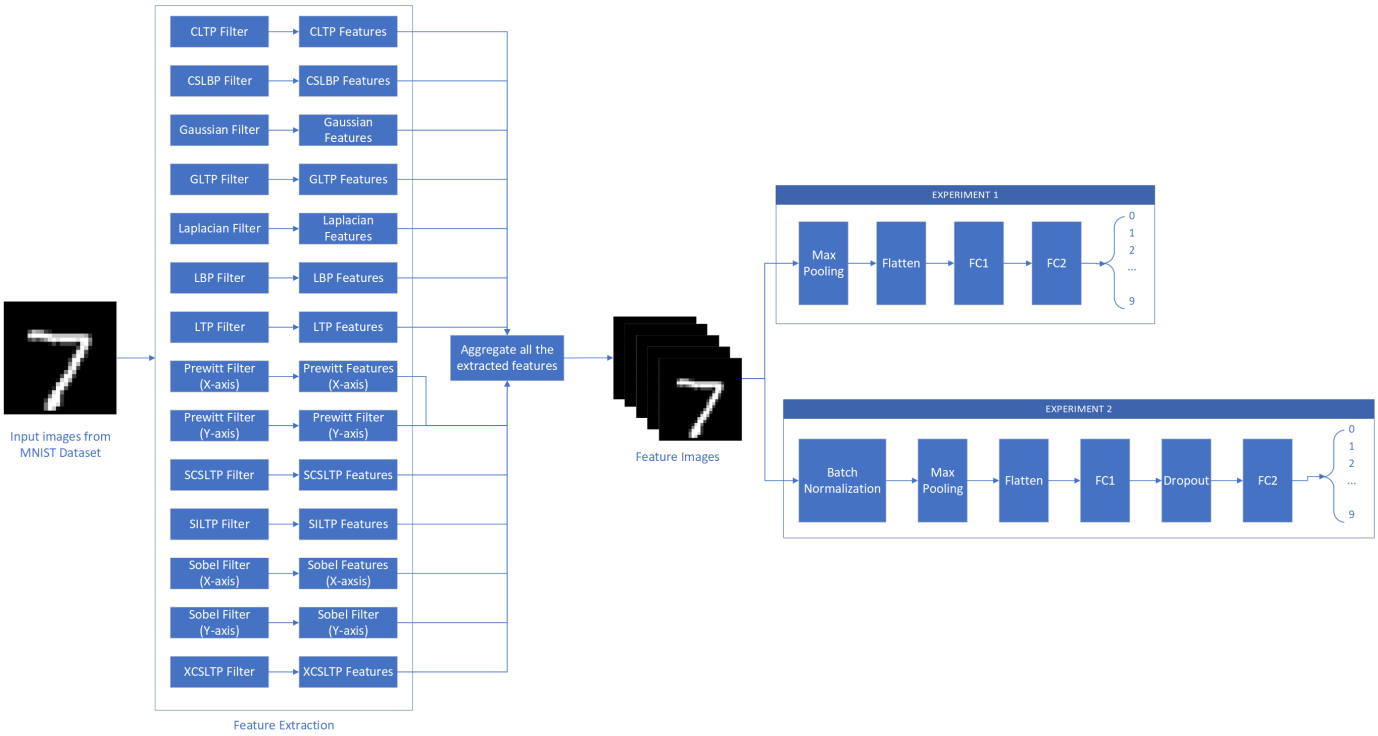


Fig. 5. Extract features via traditional filters and perform ten-class classification task on the MNIST datasets

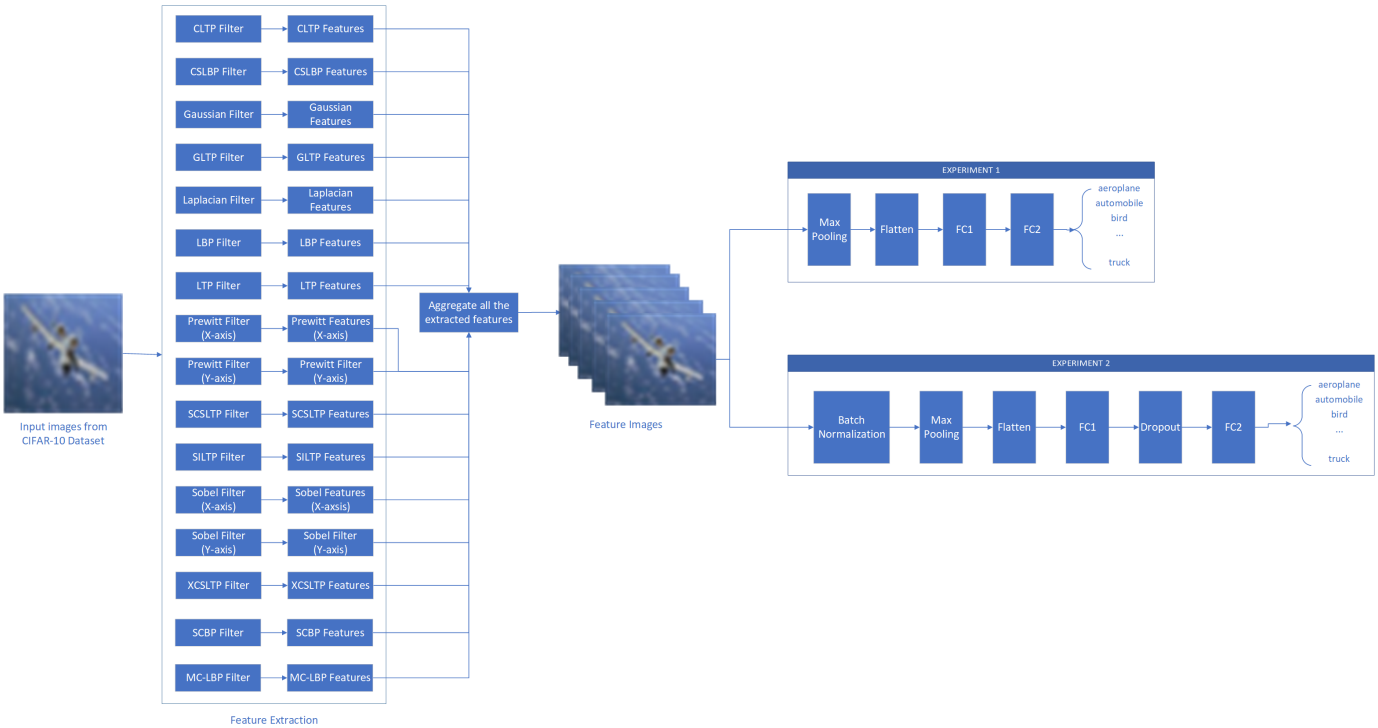
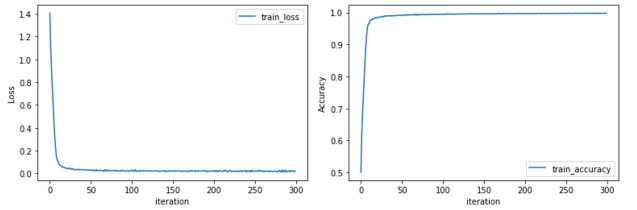


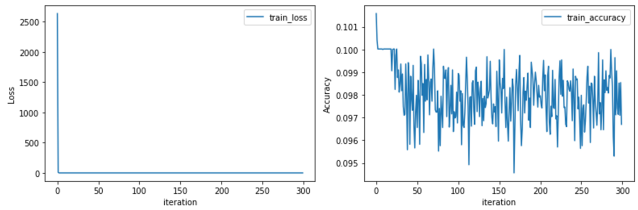
Fig. 6. Extract features via traditional filters and perform ten-class classification task on the CIFAR-10 datasets

TABLE II
NUMBERS OF FILTERS USED IN TRADITIONAL FILTERS AND CNN FILTERS

Filter Name	Axis	CIFAR-10		MNIST	
		CNN Filters	Traditional Filters	CNN Filters	Traditional Filters
Prewitt	**Prewitt X**	1	1	1	1
	Prewitt Y	1	1	1	1
Sobel	**Sobel X**	1	1	1	1
	Sobel Y	1	1	1	1
Laplacian	**Laplacian**	1	1	1	1
Gaussian Blur	**Gaussian Blur**	1	1	1	1
Local Binary Pattern (LBP)	**LBP**	1	1	1	1
Center-symmetric LBP (CS-LBP)	**CS-LBP**	1	1	1	1
Spatial Color Binary Pattern (SCBP)	**SCBP**	1	1	X	X
Multi-channel LBP (MC-LBP)	**MC-LBP**	1	1	X	X
Local Ternary Pattern (LTP)	**LTP**	1	1	1	1
Completed LTP (CLTP)	**CLTP**	1	1	1	1
Scale Invariant LTP	**SILTP**	1	1	1	1
Scale-invariant Center Symmetric LTP	**SCS-LTP**	1	1	1	1
Extended Center-symmetric LTP	**XCS-LTP**	1	1	1	1
Generalized LTP	**GLTP**	1	1	1	1
Summary		16	16	14	14

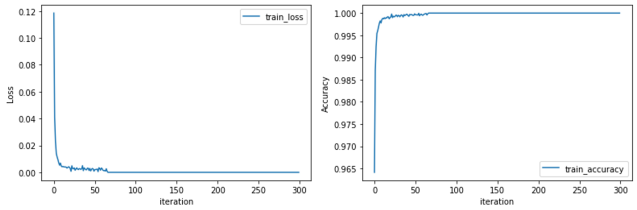


Experiment 1 – CIFAR-10 – CNN Filters

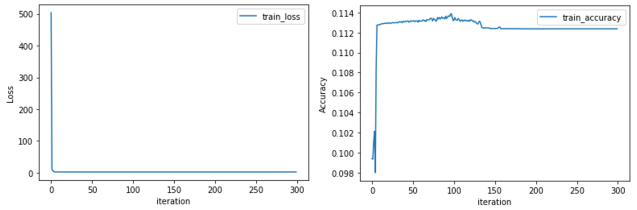


Experiment 1 – CIFAR-10 – Traditional Filters

Fig. 7. Experiment 1 - Train Accuracy and Train Loss are shown respectively when applying CNN filters and traditional filters on CIFAR-10 dataset



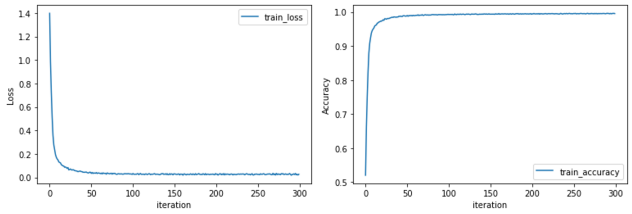
Experiment 1 – MNIST – CNN Filters



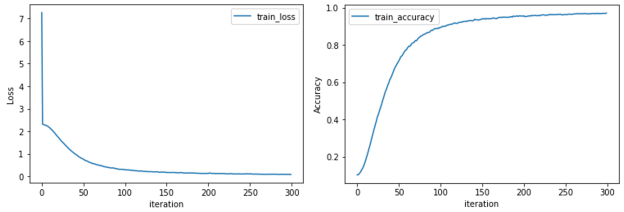
Experiment 1 – MNIST – Traditional Filters

Fig. 8. Experiment 1 - Train Accuracy and Train Loss are shown respectively when applying CNN filters and traditional filters on MNIST dataset

[5] "CIFAR-10 and CIFAR-100 datasets."
[6] B. Dong and X. Wang, "Comparison deep learning method to traditional methods using for network intrusion detection," in *2016 8th IEEE International Conference on Communication Software and Networks (ICCSN)*, (Beijing, China), pp. 581–585, IEEE, June 2016.
[7] N. G. Paterakis, E. Mocanu, M. Gibescu, B. Stappers, and W. van Alst, "Deep learning versus traditional machine learning methods for aggregated energy demand prediction," in *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, (Torino), pp. 1–6, IEEE, Sept. 2017.
[8] G. Luppescu, "Comparing Deep Learning and Conventional Machine Learning for Authorship Attribution and Text Generation," 2017.
[9] A. Stan and G. Mircea, "A Comparison Between Traditional Machine Learning Approaches And Deep Neural Networks For Text Processing In Romanian," Nov. 2018.
[10] A. Sboev, I. Moloshnikov, D. Gudovskikh, A. Selivanov, R. Rybka, and T. Litvinova, "Deep Learning neural nets versus traditional machine learning in gender identification of authors of RusProfiling texts," *Procedia Computer Science*, vol. 123, pp. 424–431, 2018.
[11] Y. Lai, "A Comparison of Traditional Machine Learning and Deep Learning in Image Recognition," *Journal of Physics:*

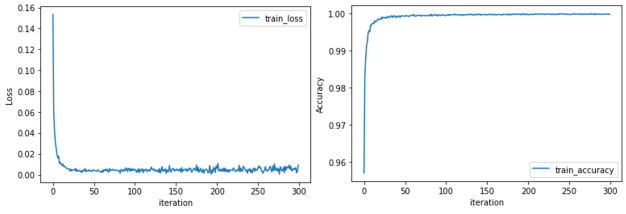


Experiment 2 – CIFAR-10 – CNN Filters

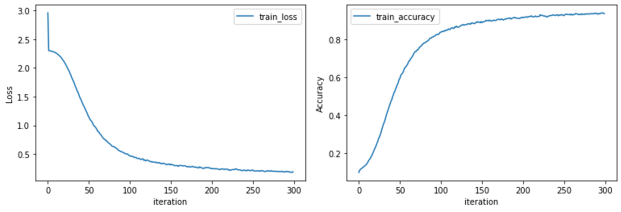


Experiment 2 – CIFAR-10 – Traditional Filters

Fig. 9. Experiment 2 - Train Accuracy and Train Loss are shown respectively when applying CNN filters and traditional filters on CIFAR-10 dataset



Experiment 2 – MNIST – CNN Filters



Experiment 2 – MNIST – Traditional Filters

Fig. 10. Experiment 2 - Train Accuracy and Train Loss are shown respectively when applying CNN filters and traditional filters on CIFAR-10 dataset

TABLE III
DETAILED EXPERIMENTAL RESULTS FOR EACH METHOD APPLYING INTO THE DATASETS

Experiment No.	Types of Filters	Dataset	Metrics (Evaluated on the test set)	
			Accuracy Score	Recall Score
1	Traditional Filters	MNIST	11.35%	10%
		CIFAR-10	10.01%	10.01%
	CNN Filters	MNIST	**98.95%**	**98.94%**
		CIFAR-10	**63.49%**	**63.49%**
2	Traditional Filters	MNIST	9.99%	9.89%
		CIFAR-10	9.43%	9.43%
	CNN Filters	MNIST	**98.83%**	**98.82%**
		CIFAR-10	**64.92%**	**64.92%**

Conference Series, vol. 1314, p. 012148, Oct. 2019.

[12] M. Munir, M. A. Chattha, A. Dengel, and S. Ahmed, "A Comparative Analysis of Traditional and Deep Learning-Based Anomaly Detection Methods for Streaming Data," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, (Boca Raton, FL, USA), pp. 561–566, IEEE, Dec. 2019.

[13] S. Chauhan, L. Vig, M. De Filippo De Grazia, M. Corbetta, S. Ahmad, and M. Zorzi, "A Comparison of Shallow and Deep Learning Methods for Predicting Cognitive Performance of Stroke Patients From MRI Lesion Images," *Frontiers in Neuroinformatics*, vol. 13, p. 53, July 2019.

[14] M. Shaik Taj and R. S Narayana, "Comparative performance analysis of LBP and LTP based facial expression recognition," *International Journal of Applied Engineering Research*, vol. 12, no. 17, pp. 6897–6900.

[15] M. Lewis, G. Elad, M. Beladev, G. Maor, K. Radinsky, D. Hermann, Y. Litani, T. Geller, J. M. Pines, N. l. Shapiro, and J. F. Figueroa, "Comparison of deep learning with traditional models to predict preventable acute care use and spending among heart failure patients," *Scientific Reports*, vol. 11, p. 1164, Dec. 2021.

[16] A. Sheneamer, "Comparison of Deep and Traditional Learning Methods for Email Spam Filtering," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 1, 2021.

[17] T. Ojala, M. Pietikainen, and D. Harwood, "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions," in *Proceedings of 12th International Conference on Pattern Recognition*, vol. 1, (Jerusalem, Israel), pp. 582–585, IEEE Comput. Soc. Press, 1994.

[18] M. Heikkilä, M. Pietikäinen, and C. Schmid, "Description of interest regions with local binary patterns," *Pattern Recognition*, vol. 42, pp. 425–436, Mar. 2009.

[19] Xiaoyang Tan and B. Triggs, "Enhanced Local Texture Feature Sets for Face Recognition Under Difficult Lighting Conditions," *IEEE Transactions on Image Processing*, vol. 19, pp. 1635–1650, June 2010.

[20] C. Silva, T. Bouwmans, and C. Frélicot, "An eXtended Center-Symmetric Local Binary Pattern for Background Modeling and Subtraction in Videos:," in *Proceedings of the 10th International Conference on Computer Vision Theory and Applications*, (Berlin, Germany), pp. 395–402, SCITEPRESS - Science and and Technology Publications, 2015.

[21] S. Liao, G. Zhao, V. Kellokumpu, M. Pietikäinen, and S. Z. Li, "Modeling pixel process with scale invariant local patterns for background subtraction in complex scenes," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1301–1306, June 2010. ISSN: 1063-6919.

[22] Z. Zhang, B. Xiao, C. Wang, W. Zhou, and S. Liu, "Background modeling by exploring multi-scale fusion of texture and intensity in complex scenes," in *The First Asian Conference on Pattern Recognition*, pp. 402–406, Nov. 2011. ISSN: 0730-6512.

[23] T. H. Rassem and B. E. Khoo, "Completed Local Ternary Pattern for Rotation Invariant Texture Classification," *The Scientific World Journal*, vol. 2014, pp. 1–10, 2014.

[24] W. Zhou, Y. Liu, W. Zhang, L. Zhuang, and N. Yu, "Dynamic Background Subtraction Using Spatial-Color Binary Patterns," in *2011 Sixth International Conference on Image and Graphics*, pp. 314–319, Aug. 2011.

[25] F. Ma and N. Sang, "Background Subtraction Based on Multi-channel SILTP," in *Computer Vision - ACCV 2012 Workshops* (J.-I. Park and J. Kim, eds.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 73–84, Springer, 2013.

[26] V. Q. Dinh, C. C. Pham, and J. W. Jeon, "Matching cost function using robust soft rank transformations," *IET Image Process.*, vol. 10, no. 7, pp. 561–569, 2016.

[27] V. Q. Dinh, V. D. Nguyen, H. V. Nguyen, and J. W. Jeon, "Fuzzy encoding pattern for stereo matching cost," *IEEE Trans. Circuits*

Syst. Video Technol., vol. 26, no. 7, pp. 1215–1228, 2016.

[28] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov. 1998.

[29] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *International Conference on Machine Learning*, pp. 448–456, PMLR, June 2015.

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[31] "Impact of fully connected layers on performance of convolutional neural networks for image classification," *Neurocomputing*, vol. 378, pp. 112–119, Feb. 2020.