Interpretable Convolutional Learning Classifier System (C-LCS) for higher dimensional datasets

Jelani Owens[1], Kishor Datta Gupta[2], Xuyang Yan[1], Lydia Asrat Zeleke[1], and Abdollah Homaifar[1,*]

***Abstract*— The purpose of this paper is to devise an interpretable hybrid classification model for Convolutional Neural Networks (CNN) and a Learning Classifier System (LCS). The presented hybrid system integrates the fundamental attributes from both types of these classifiers. In the proposed hybrid model CNN works as an automatic feature extractor, and LCS works to provide interpretable rule-based classification results. Although LCS has limitations working on higher dimensional datasets, we resolve this limitation by using CNN as a feature extractor. The other concept of the non-interpretability of CNN is addressed by using the LCS rule. Furthermore, our experiment with higher dimensional datasets like CIFAR-10 and Fashion-MNIST shows that extended LCS provides comparable performance to the standard neural network model while also providing interpretable results. We named this extended LCS method Convolutional Learning Classifier Cystem (C-LCS).**

I. Introduction

Although effective, deep learning models (usually with millions of model parameters) are challenging to interpret. We address this challenge in this research with the approached explained as follows.

Because, an interpretable predictive model results in faster adaptability use within some industries, (especially the healthcare domain) [1], [2] is prevalent. The application of decision trees [3], because of their ease of interpretation, have been noted as successfully employed in the health care domain [4]–[6]. However, decision trees can easily overfit and may function poorly on large heterogeneous datasets. Convolutional Neural Networks (CNN) are widely used for high dimensional data sets like image classification tasks. CNNs are able to achieve high-quality image classification performance with their ability to extract concrete features from image datasets [7]. The main limitation with these deep learning models revolves around the limited interpretability they provide during the classification process [8]. Interpretability describes how clear the inner workings of a machine learning model are to the interpetation of a human user [9]. Therefore, the question is how can we develop a novel data-driven learning model which can attain state-of-the-art performance like deep learning models and in conjunction be effortlessly interpreted by human users.

In recent studies, Artificial Intelligence researchers have attempted to interpret learned features from deep neural nets. An early work [10] explored visualizing the hierarchical structures learned by neural nets. Simultaneously, another research team [11] explored feature generalization in CNN. Additionally, [12] claimed that interpreting individual units of deep models could result in misleading causality. This research demonstrates that it is possible to interpret deep learning. However, the behavior of deep neural models still remains more complicated than previously assumed, motivating researchers to find alternative approaches to interpreting how deep models work.

Meanwhile, [13] demonstrated empirically that shallow neural networks are capable of conducting similar prediction performance as deep neural networks. First training a state-of-the-art deep model, and then training shallow neural networks, predictions made by the deep model as target labels can then be used. Similarly, Hinton et al. [14] presented an efficient knowledge distillation approach to transfer (deep) knowledge from model ensembles into a single model following the concept of model compression [15]. Another work [16] brings a Bayesian approach to distill knowledge from a deep neural network to a shallow neural network. Furthermore, mimic learning has also been successfully used for multitask learning, reinforcement learning, and speech processing applications [17]–[19]. In figure 1, an interpretable approach by [20] is presented, where a learning model is used to mimic blackbox models. In mimic learning, the teacher usually is a deep model, while the student is typically deployed as a shallow model resulting in a more effortless interpretation. The general process of mimic learning can be viewed as a distillation procedure from teacher to student. The interpretable student model provides a global perspective in a post-hoc manner for the deep teacher model [21]. Surrogate Strategies is a similar approach explored by [22]. In Surrogate Strategy an inherently interpretable model is trained using the same data as a black-box model to serve as the black-box model's surrogate. These works inspired us to study the possibility of employing mimic learning to comprehend an interpretable model or find a suitable surrogate model for deep neural net and, in parallel, achieve a similar performance as a deep neural network.

In this paper, we propose that Learning Classifier Systems (LCS) may be a suitable contender for a mimic learning model, or as surrogate model. In a low-dimensional data format, LCS yields two advantages over Neural Networks. First, LCS is considerably more flexible when training multiple types of datasets based on how LCS creates the machine learning model. Second, LCS provides interpretability to the

[1]North Carolina Agricultural and Technical State University, Greensboro, North Carolina, US, 27411. Emails: jowens2@aggies.ncat.edu;xyan@aggies.ncat.edu; lzeleke@aggies.ncat.edu; homaifar@ncat.edu

[2]Clark Atlanta University, Atlanta, GA, US. Email: kgupta@cau.edu

*Corresponding Author

user based on the rules that the system creates and is also used for the prediction model. Most common mimic learning model decision trees are susceptible to small perturbation in input data. A small change in the data can cause a large change in the structure of the decision tree causing instability, but LCS has low vulnerability for perturbations compared to decision trees models.

While the LCS shows significant strength over the Neural Network in a non-image classification problem, image classification for LCS becomes more complicated. The LCS normally struggles with image dataset because of the high dimensionality of features and the inability of LCS to extract the location values of each pixel in the image dataset. To address these issues, our work proposes a new machine learning model that not only produces high performance results but, also provides better interpretability. The new model, namely Convolutional-Learning Classifier System (C-LCS), uses a pretrained Convolutional Neural Network to extract the features from an image dataset and then shares the new feature space with the LCS to train. The convolutional feature extraction layer properly allows the LCS to create a robust prediction model.

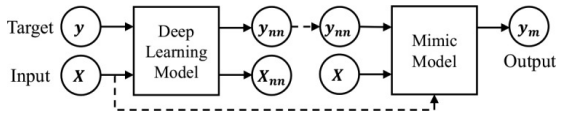


Fig. 1: Mimic performance of Blackbox model with opaque models [20].

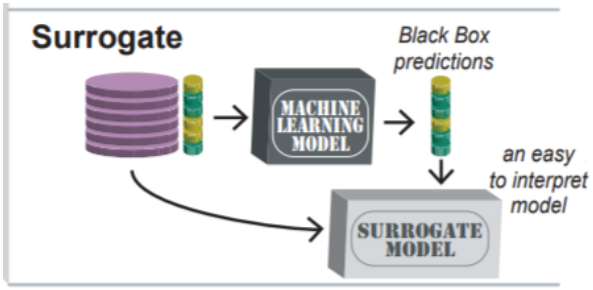


Fig. 2: Surrogate strategy for interpretable Machine Learning [22].

Our literature study suggested that if considering a learning model as an explainer for the BlackBox, the learning model needs to be a self-interpretable model with comparable accuracy to the BlackBox model. LCS properties make it self-interpretable, and shows comparable accuracy in low-dimensional datasets. This paper illustrates that LCS can also be close to state-of-the-art learning models for high-dimensional dataset with a static feature extraction process. Given static features extraction processes are self-interpretable the extended LCS remains interpretable. We used a static extraction method which was generated by different datasets (example: Imagenet) for a state-of-the-art ML methods like Vgg-19. Experimental results show that the

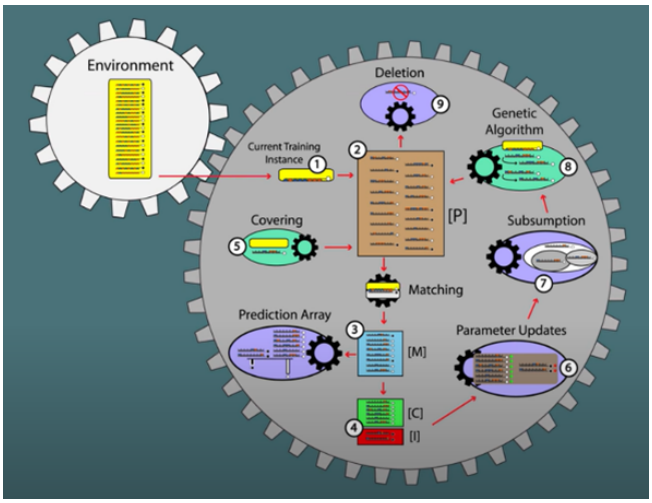


Fig. 3: Learning Classifier System (LCS) [23].

extended LCS (C-LCS) model gives comparable results with deep learning models (example: resnet) in different datasets (example: CIFAR) when a dimension reduction method is employed.

The contribution of our work is summarized as follows:

- Extend the LCS method with a static feature extraction process and dimension reduction layer for image classification tasks.
- Experimental data suggestings the pooling layer performs better than PCA as a dimension reduction method and Vgg-19 trained convolutional layer performs better for C-LCS.
- The experiments with the high-dimensional image datasets exhibit C-LCS can be an alternative to mimic the black-box machine learning method as an interpretable approach, while providing comparable performance with the deep learning method.

The layout of this paper is structured as follows: section II provides background on the Learning Classifier System and discusses the details of the interpretability of LCS. Section III presents the proposed C-LCS technique in terms of feature extraction, feature transformation, and prediction model. Section IV illustrates the experiments and related results with detailed discussions on the results and provides possible reasons for those experimental results. Finally, Section V concludes our paper and outlines future work.

II. Preliminaries

A. Learning Classifier System

A Learning Classifier System is an evolutionary machine learning algorithm that creates its prediction model through a set of rules. The rules are structured in a IF...THEN format where the IF conditions are based on the features in the dataset while the THEN result statement is based on the classes within the dataset. In figure 3 the components of the LCS are discussed. The LCS is composed of three sections: Environment, Learning Cycle, and the Post-Processing Stage. The Environment contains the dataset for

the machine learning algorithm. To improve the performance of the LCS, the user can analyze patterns between features and classes to make the rule generation procedure run more smoothly. One example is to add a Rule Specificity Limit (RSL) for the environment. The RSL puts a restriction on the number of features used to make rules for LCS. Putting a focus on selecting the most important features optimizes the time it takes for the LCS to create rules for the prediction model. Regardless of the decision to perform analysis on the Environment or not, the instances located in the Environment play a vital role in initializing the creation of a LCS.

B. Interpretable Machine Learning

Interpretability in machine learning can be defined as the capacity where a machine learning based model provides insight into its behavior or decision-making process to an interested end-user. Interpretability can be achieved intrinsically or by using post-hoc techniques for trained machine learning models [24]. Models built from algorithms that inherently enforce their interpretability are known as transparent models. Transparent models include linear regressions, logistic regressions, decision trees, RuleFit algorithm, attention models and LCS. Deep neural networks are complex non-linear black-box models that, over the years, have been tasked to perform high stake decisions in sensitive areas such as healthcare, the justice system, and autonomous driving. Shortcomings of their application in those fields [25]–[27] has recently raised a fair amount of public distrust and has generated sufficient incentive to call for increased accountability and transparency in application of those models.

C. LCS as surrogate model for Interpretable ML

Our work demonstrates that interpretability can be incorporated into a complex deep learning model without necessarily undermining the accuracy. As stated previously, the purpose of interpretability is to help data scientists and engineers identify biases within a system and allow users to fix the problems with the models in a timely manner [20]. The rules from the LCS are a key factor in the interpretability of the system. When the Rules are created in the LCS, the algorithm gives a clear view on which features or attributes should used to create a prediction model. The conditions for the attributes are either given in a numerical format or a true or false format shown to the user as the LCS model is being trained. These rules provide the idea behind the algorithms thought process and how the prediction model was created for best application and use. Another factor that lends more clarity is the significance of each attribute within the dataset. Attributes that are not used in the LCS are given # or don't care symbols meaning that they do not have an effect on whether a program predicts an instance or is a certain label or not. This information allows the user to push the algorithm to focus on specific attributes which make the LCS more efficient.

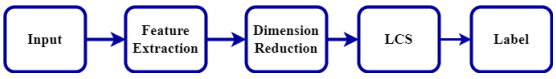


Fig. 4: Simplified steps of the proposed method

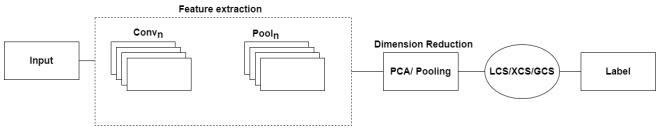


Fig. 5: Simplified steps of Convolutional Learning Classifier System (C-LCS).

III. Proposed method

The C-LCS consists of four different phases: Feature Extraction, Dimension Reduction, Hyperparameter Tuning, Classifier. In the figure 4 these steps are visually presented. The first two components contribute to the setup of the new environment for the Learning Classifier System. A CNN in combination with a Dimension Reduction layer is used as the new Environment for the LCS where it can use a new feature space to properly construct rules for the prediction for the system.

A. Feature Extraction

The Feature Extraction component of the C-LCS uses a static pretrained Convolutional Neural Network to extract the features from the original dataset. This process is static in training and testing time to allow the interpretability of LCS. In traditional CNN this layer is dynamically updated in training time. The combination of Convolutional layers and Max Pooling layers as illustrated in figure 5. These layers adjust the dimension size of the feature space then properly put through a classifier while also drawing out crucial information from the feature space. After the dataset has passed through the Convolutional Neural Network, the new feature space is placed into the next component of the C-LCS. This layer is static and pretrained using a well-established dataset. In our experiment we used Imagenet pretrained convolutional layers.

B. Dimension Reduction

The next portion of the C-LCS features the Dimension Reduction component. Dimension Reduction reduces the size of the feature space of the original dataset while attempting to maintain as much information from the original dataset as possible. This component is vital to the C-LCS because LCS tends to produce high-quality results with low dimensional feature spaces. For this experiment, an additional pooling layer was used to reduce the dimensionality of the feature space along with PCA to compare the effectiveness of both methods.

C. Hyperparameter Tuning and Classifier

The final three components of the C-LCS are composed of the LCS itself. The first component involves Hyperparameter Tuning. Hyperparameter Tuning is critical for finding

the proper values to create a Classifier that will produce the best classification results. The details on the specific Hyperparameter Tuning procedure are further discussed in the Experimental Setup and Results section. After finding the best parameters, the Classifier portion of the C-LCS is established where it outputs the final portion of the C-LCS: the Label space. This section contains all of the predictions made by the Classifier.

IV. Experiment and Result analysis

A. Experimental Setup

The goal of our experiment was to prove that LCS with convolutional layer has comparable performance with CNN state-of-the-art deep learning model for higher dimensional benchmark datasets and to demonstrate how the LCS adds interpretability to the neural network model. As the first goal of our experiment, we compared the classification performance of a prediction model generated from a Convolutional Neural Network and a C-LCS. The performance of models generated from the Neural Network and LCS in both an image and non-image format were contained here as well. Each of these algorithms was run using the Google Colab API, a cloud programming platform that allows the user to execute multiple cells of python code in a notebook environment. The features extracted from the pretrained Convolution Neural Network are compiled as a table and exported as an excel file for analysis. Finally, Receiver Operating Characteristics (ROC) and Precision Recall Curves(PRC) were used to evaluate the performance of the prediction model.

For the second portion of the experiment, we compare the classification performance of a prediction model generated from a Neural Network, Decision Tree, LCS, and XCS. Like the previous portion, ROC and PRC are used to evaluate the performance of the prediction model. In addition to the display of performance metrics, rules generated from the Decision Tree, LCS, and XCS are shown as a graph to explain the classification patterns in the Neural Network.

To evaluate the performance of each algorithm, we use the winequality-white, winequality-red, Fashion MNIST, MNIST and CIFAR10 datasets. This approach is used to see how each algorithm responds to different types of datasets based on feature size, image vs non-image, and color dataset vs non-color datasets. The wine quality sets are used to demonstrate the rule creation process for the LCS, XCS, and Decision Tree because they are non-image datasets and contain a small amount of features.The winequality-white and winequality-red datasets are based on user reviews on the taste of wine based on different factors. The main difference between the two winequality datasets is the number of sample for each dataset. To simplify the datasets, we convert the winequality datasets to a binary classification problem where reviews above 7 are good while anything below 7 is bad. The MNIST dataset is a 28x28 grayscale image dataset based on numerical digits. The Fashion MNIST dataset is similar to the MNIST set however, it uses grayscale images of clothes instead of numbers. The CIFAR10 dataset is a 32x32 colored image dataset based on real-world objects such as cats, dogs, and other subjects. For winequality-white, the training set consists of 3672 samples and the testing set consists of 1225 samples. For winequality-red, the training set consists of 1198 samples and testing set consists of 400 samples. For MNIST, the training set consists of 60,000 images and the testing set consisted of 10,000 images. For CIFAR10, the training set has 50,000 images while the testing set has 10,000 images.

B. Evaluation Metrics

The performance of the prediction models from both Convolutional Neural Network and the Convolutional Neural Network-Learning Classifier System are measured using a Receiver Operating Characteristic (ROC) and a Precision Recall Curve. The ROC measures the area under the curve of a graph that maps the true positive rate over the false positive rate.

C. Hyperparameter Tuning

The parameters of the Convolutional Neural Networks are tuned through a two step process: Random Search and Grid Search. The first step goes through a simulation which tests the performance of randomly generated combinations from a list of parameters generated by the user. At the end of this process, the combination of parameters with the best classification performance is stored for later use. Since Random Search only goes through a selected number of parameter combinations per test, it is repeated multiple times throughout the experiment to produce a new set of parameters for the subsequent Grid Search. After compiling a new set of parameters, the new list of conditions are passed through a Grid Search. The Grid Search goes through every possible case of combinations of parameters and evaluates which combination produces the best classification result. The Random Search is usually implemented before Grid Search to reduce the amount of time spent on the Grid Search. After the Grid Search, the user takes the combination of parameters generated by procedure and uses these values to train the Neural Network to make the prediction model.

To tune the Convolutional Learning Classifier System, the learning iterations of the system are constantly increased throughout the experiment. As stated in section II, increasing the number of learning iterations provides more time for the C-LCS to further evaluate properties of the dataset and to improve on the quality of the rules created throughout the system. The initial number of learning iterations for the experiment is 50. This number derives from a previous test with non-image dataset for the comparison between the Neural Network and Learning Classifier System. As the hyperparameter tuning process continues, the amount of learning iterations increases to 1000 and is increased by 1000 for the remaining tests. Eventually, for both datasets, the parameter tuning process ends when the number of learning iterations reaches 10000. This value provides favorable results for the experiment. The increase of the learning iterations beyond this point considerably increases the run time

of creating prediction model with very little improvement on the classification results.

TABLE I: Wine Quality-Red Results

Machine Learning Algorithm	**LCS**	***Neural Network***	***XCS***	***Decision Tree***
PRC	0.432	0.455	0.465	0.499
ROC	0.853	0.880	0.815	0.733

TABLE II: Wine Quality-White Results

Machine Learning Algorithm	**LCS**	***Neural Network***	***XCS***	***Decision Tree***
PRC	0.430	0.499	0.414	0.608
ROC	0.748	0.793	0.702	0.746

TABLE III: MNIST PCA results

PCA Features	**Metrics**	***LCS***	***Neural Network***	***C-LCS***	***CNN***
500	Average PRC	0.500	0.193	0.107	0.530
	Average ROC	0.500	0.549	0.529	0.864
200	Average PRC	0.384	0.188	0.236	0.553
	Average ROC	0.501	0.551	0.716	0.872
100	Average PRC	0.104	0.193	0.416	0.5339
	Average ROC	0.504	0.550	0.842	0.899
50	Average PRC	0.119	0.203	0.511	0.570
	Average ROC	0.497	0.553	0.876	0.887

TABLE IV: PCA CIFAR10 results

PCA Features	**Metrics**	***LCS***	***Neural Network***	***C-LCS***	***CNN***
500	Average PRC	0.146	0.191	0.117	0.379
	Average ROC	0.504	0.661	0.530	0.809
200	Average PRC	0.116	0.190	0.210	0.425
	Average ROC	0.503	0.662	0.699	0.837
100	Average PRC	0.114	0.193	0.338	0.470
	Average ROC	0.518	0.665	0.817	0.848
50	Average PRC	0.112	0.196	0.347	0.451
	Average ROC	0.518	0.669	0.798	0.853

D. Result analysis

Table I displays the performance of the Neural Network compared to 3 other interpretable classifiers. The four classifiers produced similar results to each other with PRC scores ranging from 0.432 to 0.499 and ROC scores ranging from 0.7330 to 0.880. Out of the four classifiers, the Decision Tree had the highest PRC score out of the group. However, it also had the lowest ROC score. The low ROC is explained by the overfitting nature of the Decision Tree. Figure 8 helps outline how overfitting occurs within the Decision Tree. Also, note that figure 8 only display a portion of the entire Decision When a condition is created for the Decision Tree, the algorithm only looks at the relationship of one numerical value for one feature. The limited search scope makes the Decision Tree more susceptible to change based on the iteration of fed to the Decision Tree. One small cLCS: range of values for one feature. Same pattern occurs with XCS. The #numbers in figure 9 indicate when certain features can be ignored for a rule in the prediction model. Figure 10 provides a more condensed form of figure 9 giving a clearer picture of which features have an impact on this particular portion of the dataset. This conversion is done by eliminating the features that contain the # within their conditions. In conclusion, establishing a range of values for a condition to occur provides more flexibility in terms of prediction. The high flexibility provides better scores for the prediction model.

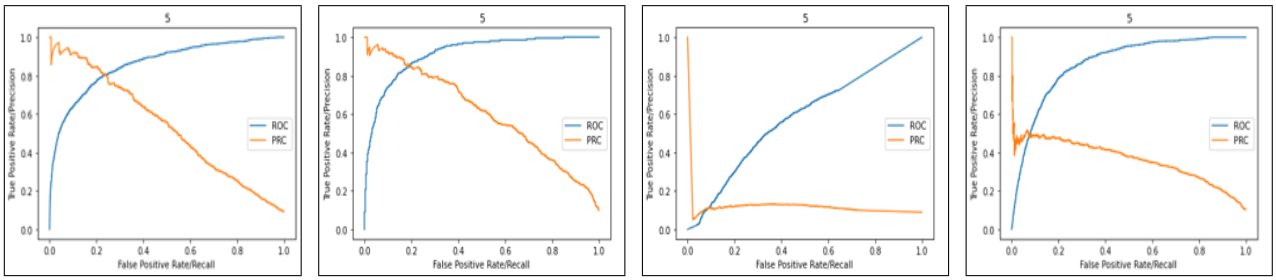
TABLE V: Convolutional Layer CIFAR10 results

Number of Covolutional Features	**Metrics**	***Vgg 19 CNN***	***Resnet CNN***	***Vgg 19 C-LCS***	***Resnet C-LCS***
500	Average PRC	0.9164	0.8437	0.2587	0.5501
	Average ROC	0.9848	0.9681	0.6236	0.5001
200	Average PRC	0.9031	0.812	0.4083	0.2536
	Average ROC	0.9818	0.9613	0.8262	0.5101
100	Average PRC	0.8509	0.719	0.4507	0.2585
	Average ROC	0.9712	0.9352	0.8602	0.7472
50	Average PRC	0.7646	0.5893	0.4062	0.2665
	Average ROC	0.9511	0.8998	0.8456	0.7574

For table II, The unusual results can be a result of the difference in size of the wine datasets. The white wine dataset is roughly 3 times the size of the red wine dataset. The increase in size results in more patterns for the LCS, XCS, and Decision Tree to process. The results, table II will contain worse results compared to table I because the second dataset is more complex.

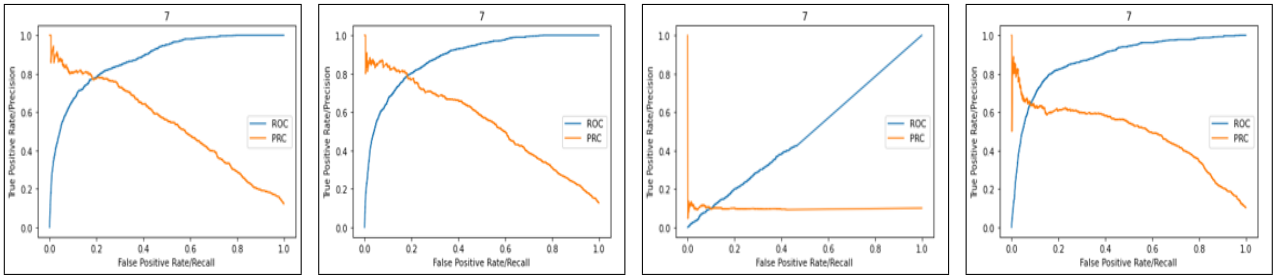
Table III presents the performance of the non-Convolutional classifiers and Convolutional classifiers when PCA is used to reduce the dimensionality of the feature space for the MNIST dataset. For these results, both the LCS and Neural Network showed poor performance. The ROC and PRC scores of the LCS model range from 0.119 to 0.504 meaning that results were inconclusive. A similar pattern occurred with the Neural Network however the scores are slightly better than the LCS scores. This result can be explained by the Neural Network's ability to perform better than the LCS when the dataset contains a high feature count. However, due to the low ROC and PRC scores, it is difficult to reach a conclusion solely using this information. It can be concluded that the low classification scores are attributed to the poor feature extraction performance. The LCS and Neural Network did not properly extract the locational information from the pixels and thus there is information that is considered in the prediction stage.

Both the CNN and C-LCS achieved much more favorable scores compared to their non-convolutional counterparts when exposed to PCA. The ROC scores for the CNN particularly reaches values over 0.8 and it indicates that the CNN model did a great job in classifying the different labels in the test set. The C-LCS starts to display less favorable



(a) CNN 500 PCA MNIST Features (b) CNN 50 PCA MNIST features (c) C-LCS 500 PCA MNIST features (d) C-LCS 50 PCA MNIST features

Fig. 6: CNN and C-LCS comparison on MNIST dataset



(a) CNN 500 PCA CIFAR10 Features (b) CNN 50 PCA CIFAR10 features (c) C-LCS 500 PCA CIFAR10 features (d) C-LCS 50 PCA CIFAR10 features

Fig. 7: CNN and C-LCS comparison on CIFAR10 dataset

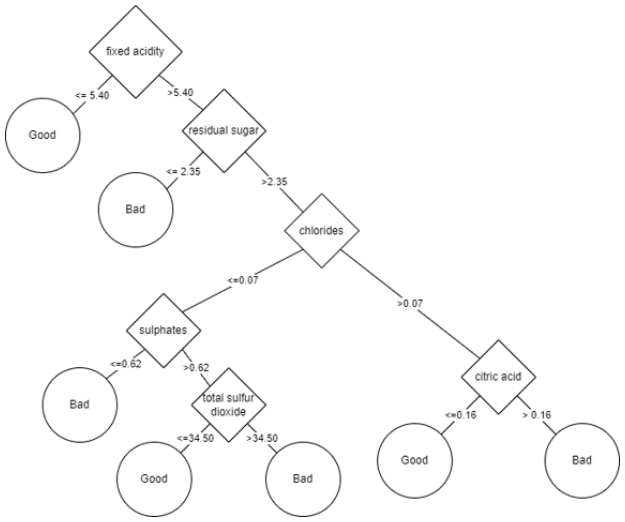


Fig. 8: Decision Tree Rule for Red Wine Quality Flowchart

ROC scores in test set with a higher feature count initially. As the dimensionality of the feature space starts to decrease, particularly around 100 features, the C-LCS produces better scores. Although the scores never match or exceed the scores from the CNN, the scores of the C-LCS with less than 100 PCA features hover around 0.8, implying that the prediction models generates from the C-LCS performed with adequate execution. The performance improvement can be visualized in the lower graphs in Figure 6. When the feature count was 500, the ROC curve appears as diagonal line and provides a

Fixed acidity	Volatile acidity	Citric acid	Residual sugar	chlorides	Free sulfur dioxide	Total sulfur dioxide	density	pH	Sulphates	alcohol	quality
6.184-13.42	-0.013-0.893	#	0.602-4.398	-0.105-0.231	-17.27-35.27	#	0.996-0.999	2.883-3.597	#	#	Bad

Fig. 9: LCS Rule 1 for Red Wine Quality Traditional Form

Specified Values	Specified Attribute Name	quality
[6.184 – 13.42]	Fixed acidity	Bad
[-0.013 – 0.893]	Volatile acidity	
[0.602 – 4.398]	Residual sugar	
[-0.105 – 0.231]	Chlorides	
[-17.27 – 35.27]	Free sulfur dioxide	
[0.996 – 0.999]	Density	
[2.883 – 3.597]	pH	

Fig. 10: LCS Rule 1 for Red Wine Quality Discrete Continuous Attribute List (DCAL) Form

Fixed acidity	Volatile acidity	Citric acid	Residual sugar	chlorides	Free sulfur dioxide	Total sulfur dioxide	Density	pH	Sulphates	alcohol	quality
#	#	#	#	-0.108-0.181	-4.55-24.555	-59.64-141.64	0.996-1.000	#	#	#	Bad

Fig. 11: XCS Rule 1 for Red Wine Quality Traditional Form

Specified Values	Specified Attribute Name	quality
[-0.108-0.181]	Chlorides	Bad
[-4.55-24.555]	Free sulfur dioxide	
[-59.64-141.64]	Total sulfur dioxide	
[0.996-1.000]	Density	

Fig. 12: XCS Rule 1 for Red Wine Quality Discrete Continuous Attribute List (DCAL) Form

small area under the curve. However, when 50 PCA features are used, the ROC curve starts to round out more which increases the area under the curve. It should be noted that the PRC scores for all four machine learning algorithms are extremely low. Typically, PRC scores are low in a dataset with balanced class distributions. Under this situation, the ROC scores tend to be better than the PRC scores.

Table IV displays similar patterns to Table III with minor differences. The main difference is the classification scores for the C-LCS and CNN are lower for Table IV than Table III. Based on this result, it shows that the C-LCS and CNN had a harder time labeling the color image dataset, CIFAR10 compared to the grayscale image dataset, MNIST. Despite the decrease in classification score, the C-LCS and CNN still produce similar results in terms of labeling images. For the non convolutional classifiers, the LCS scores for both tables are very similar to each other. While the classification scores for the Neural Network in Table IV are better than the Neural Network scores in Table III, the scores are still too low to make a conclusive assessment based on the fact that they are under the value 0.7.

Transitioning to the next CIFAR10 classification test, Table IV evaluates the classification performance of the CNN and C-LCS when the Dimension Reduction method uses an additional Convolutional Layer. The unique part of this test involves using two different feature extraction CNN: Resnet and Vgg19. The purpose of using different feature extraction is to determine the effect, if any, of using different CNNs on performance of the CNN and C-LCS. Looking at the table, the Vgg19 classifier performs better in terms of classification compared Resnet. The classification scores when adding a Convolutional Layer are significantly better than using PCA. Both the PRC and ROC scores for the CNN hover around 0.8-0.9. These scores are much closer to each other compared to scores obtained from using PCA. This signifys that the Convolutional Layer method is not as affected by the balance of quantity of class label compared to the PCA. The CNN scores also decrease as the feature count starts to decrease during the test. The opposite effect occurs with the C-LCS. The scores start to improve as the feature space shrinks and the C-LCS starts to produce more favorable results (earlier) compared to the results using the PCA method. From a previous test, the C-LCS produced a favorable result with at least 100 features. For this test, the C-LCS produces a favorable result at 200 features. While the ROC scores for the C-LCS never reach the heights of the CNN, the C-LCS still produce scores of at least 0.8 signifying that the C-LCS perform an adequate job of classifying CIFAR10 images. It should be noted however that the PRC scores for C-LCS still remain low and due to these results it is recommended that the ROC score should be used to measure the classification performance, for now.

Our exhibited result shows that if we use a static pre-trained feature extractor for one model in example Resnet, we can use that pre-trained feature extraction method to mimic another model which is using vgg16. We also note that, C-LCS provided similar result using static feature extractor pre-trained by different dataset, while resnet and vgg-16 retrained convolutional layer for CIFAR and F-MNIST. This proves that CNN performance depends on its ability to customize the convolutional layers for specific datasets to get better features, but still with a static feature extractor LCS results were similar when a dimensionality reduction layer was used.

Centered on the literature study and other experiments conducted prior to this paper, the LCS displays comparable classification scores to the deep learning models. Transitioning from a non-image dataset to a image dataset, the C-LCS displays similar ROC scores to the CNN when the feature count for the dataset is low. This result shows that C-LCS can be a viable alternative for image classification when the features are reduced to a small quantity. In addition to this observation, three additional patterns can be extracted from these tests. When comparing the performance of the image datasets, the tests with the MNIST dataset had better results than the tests with the CIFAR10 datasets. This result implies that it is easier to perform image classification with a grayscaled image dataset than a colored image dataset. When comparing feature extraction methods, the Vgg19 CNN provides better results compared to the Resnet CNN. In terms of dimension reduction for this experiment, the best method to use for the process is to add a Convolutional layer to reduce the dimensionality of the feature without losing crucial information.

V. Conclusion and Future work

We developed an interpretable Convolutional-based Learning Classifier Systems (C-LCS) model as an alternative to the CNN model in image classification problems. C-LCS extends the traditional LCS models to handle high-dimensional image data and overcomes the limitations of CNN models in terms of the model's interpretability. According to the experimental results, the C-LCS produced comparable ROC scores to the CNN when the Convolutional Layer was used to decrease the dimensionality of the feature space. This shows that C-LCS can be used as a surrogate model for the deep learning approach. Additionally, C-LCS improves LCS capability to work with higher-dimensional datasets. The recent rise of interpretable ML demand offers LCS an opportunity to increase its use for different applications. We believe our attempt to make LCS applicable for higher-dimensional datasets using the feature extraction and reduction process will increase the applicability of LCS as an interpretable model. As a result, we expect an upcoming increase of research interest on LCS as interpretable ML model.

ACKNOWLEDGMENT

This work is supported by Lockheed Martin and National Institute of Aerospace's Langley Distinguished Professor Program under agreement number 234363 and C16-2B00-NCAT, respectively. Also, this work is partially funded by the NASA University Leadership Initiative (ULI) under grant number 80NSSC20M0161. The authors would like to thank them for their support.

REFERENCES

[1] M. Peleg, S. Tu, J. Bury, P. Ciccarese, J. Fox, R. A. Greenes, R. Hall, P. D. Johnson, N. Jones, A. Kumar *et al.*, "Comparing computer-interpretable guideline models: a case-study approach," *Journal of the American Medical Informatics Association*, vol. 10, no. 1, pp. 52–68, 2003.

[2] K. F. Kerr, A. Bansal, and M. S. Pepe, "Further insight into the incremental value of new markers: the interpretation of performance measures and the importance of clinical context," *American journal of epidemiology*, vol. 176, no. 6, pp. 482–487, 2012.

[3] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[4] G. Bonner, "Decision making for health care professionals: use of decision trees within the community mental health setting," *Journal of Advanced Nursing*, vol. 35, no. 3, pp. 349–356, 2001.

[5] Z. Yao, P. Liu, L. Lei, and J. Yin, "R-c4. 5 decision tree model and its applications to health care dataset," in *Proceedings of ICSSSM'05. 2005 International Conference on Services Systems and Services Management, 2005.*, vol. 2. IEEE, 2005, pp. 1099–1103.

[6] C.-Y. Fan, P.-C. Chang, J.-J. Lin, and J. Hsieh, "A hybrid model combining case-based reasoning and fuzzy decision tree for medical data classification," *Applied Soft Computing*, vol. 11, no. 1, pp. 632–644, 2011.

[7] A. Dhillon and G. K. Verma, "Convolutional neural network: a review of models, methodologies and applications to object detection," *Progress in Artificial Intelligence*, vol. 9, no. 2, pp. 85–112, 2020.

[8] G. Montavon, W. Samek, and K.-R. Müller, "Methods for interpreting and understanding deep neural networks," *Digital Signal Processing*, vol. 73, pp. 1–15, 2018.

[9] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, 2018, pp. 80–89.

[10] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *University of Montreal*, vol. 1341, no. 3, p. 1, 2009.

[11] M. D. Zeiler, R. Fergus, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, "Computer vision–eccv 2014," *Lecture Notes in Computer Science*, vol. 8689, pp. 818–833, 2014.

[12] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[13] J. Ba and R. Caruana, "Do deep nets really need to be deep?" *Advances in neural information processing systems*, vol. 27, 2014.

[14] G. Hinton, O. Vinyals, J. Dean *et al.*, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, vol. 2, no. 7, 2015.

[15] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 535–541.

[16] A. Korattikara Balan, V. Rathod, K. P. Murphy, and M. Welling, "Bayesian dark knowledge," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[17] E. Parisotto, J. L. Ba, and R. Salakhutdinov, "Actor-mimic: Deep multitask and transfer reinforcement learning," *arXiv preprint arXiv:1511.06342*, 2015.

[18] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell, "Policy distillation," *arXiv preprint arXiv:1511.06295*, 2015.

[19] E. Parisotto, J. L. Ba, and R. Salakhutdinov, "Actor-mimic: Deep multitask and transfer reinforcement learning," *arXiv preprint arXiv:1511.06342*, 2015.

[20] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu, "Interpretable machine learning: definitions, methods, and applications," *arXiv preprint arXiv:1901.04592*, 2019.

[21] F. Yang, M. Du, and X. Hu, "Evaluating explanation without ground truth in interpretable machine learning," *ArXiv*, vol. abs/1907.06831, 2019.

[22] C. B. Azodi, J. Tang, and S.-H. Shiu, "Opening the black box: interpretable machine learning for geneticists," *Trends in genetics*, vol. 36, no. 6, pp. 442–455, 2020.

[23] R. Urbanowicz, "Urbanowicz lab." [Online]. Available: https://www.med.upenn.edu/urbslab/lcs.html

[24] M. Du, N. Liu, and X. Hu, "Techniques for interpretable machine learning," *Communications of the ACM*, vol. 63, no. 1, pp. 68–77, 2019.

[25] G. Saposnik, D. Redelmeier, C. C. Ruff, and P. N. Tobler, "Cognitive biases associated with medical decisions: a systematic review," *BMC medical informatics and decision making*, vol. 16, no. 1, pp. 1–14, 2016.

[26] J. Angwin, J. Larson, S. Mattu, and L. Kirchner, "Machine bias," *ProPublica, May*, vol. 23, p. 2016, 2016.

[27] M. Harris, "Ntsb investigation into deadly uber self-driving car crash reveals lax attitude toward safety," *IEEE Spectrum*, 2019.