A Hybrid Model for Classification of Biomedical Data using Feature Filtering and a Convolutional Neural Network

Sadegh Salesi
School of Science and Technology
Nottingham Trent University
Nottingham, UK
sadegh.salesi@my.ntu.ac.uk

Ali A. Alani
Computer Science Department
College of Science
University of Diyala Diyala, Iraq
alialani@sciences.uodiyala.edu.iq

Georgina Cosma
School of Science and Technology
Nottingham Trent University
Nottingham, UK
georgina.cosma@ntu.ac.uk

***Abstract*—Deep learning is known for its capabilities in analysing large and complex sets of data, without the need of applying noise reduction methods, which is a necessary step for improving the performance of conventional machine learning models. Indeed, the superiority of deep learning over conventional machine learning models resides in their capabilities of analysing large sets of data to learn features directly from the data without the need for manual feature extraction. However, this paper aims to evaluate the hypothesis that by using feature filtering as a pre-processing step prior to feeding the data into the deep learning model, the quality of the data is improved which also leads to a better performing deep learning model. Two complex biomedical datasets which contain a large number of features and sufficient number of patient cases for deep learning were selected for the evaluations. A selection of feature filtering methods were applied to identify the most important features (i.e. top 20$\%$ ranked features) at the input level, prior to the data being fed into a deep learning classifier. Once the most important features are selected, these are fed into a deep learning algorithm, and in particular the Convolutional Neural Network, which has been tuned for the particular task. Experiment results demonstrate that applying feature filtering at the input level improves the performance of the deep Convolutional Neural Network, even for the most complex biomedical data such as those utilised in this paper. In particular, for the first dataset, PANCAN, an improvement of 20% was reported in Accuracy, whereas for the second dataset GAMETES Epistasis, an improvement of 10.63% was reported in Accuracy. The results are promising and demonstrate the benefits of filter filtering when deep learning methods are adopted for biomedical classification tasks.**

***Index Terms*—Deep learning, feature filtering, noise reduction, feature selection, classification**

I. Introduction

Advances in high throughput technologies have contributed significantly to the rapid growth in biomedical data obtained from different sources including gene sequencing, medical images. Such data are highly variable and multidimensional. The task of transforming large quantities of data into valuable information is becoming increasingly important in bioinformatics [1]. Biomedical data includes omics, image, and signal data, which is generated and analysed to discover its potential for applications in biological and healthcare research. Machine learning is a technique which has been widely used to extract underlying patterns and knowledge from complex data including biomedical and healthcare data. In biomedical and healthcare applications, machine learning algorithms are adopted to build models which can be used to predict future patient outcomes. Some well-known conventional machine learning algorithms include Artificial Neural Networks; Support Vector Machines, Ensemble Learning methods, and Bayesian networks [2]. In order to build efficient predictive models, all features are analysed and a suitable set of features are selected as predictors which are used to build the prediction model [3]. Removing redundant features is also known as the task of noise reduction, or feature selection. A new family of machine learning, deep learning, has emerged as a solution to the limited capabilities of conventional machine learning algorithms to process and analyse big and very large data. Indeed, one of the benefits of deep learning algorithms over conventional machine learning algorithms is their ability to perform automatic feature extraction from raw data, via feature learning.

Feature learning or representation learning is a set of techniques that allows a system to automatically discover the representations needed for feature detection (i.e. discovering features or representations through examination) or classification from raw data. Feature learning replaces manual feature engineering and allows a machine to both learn the features and use them to perform a specific task. The majority of deep learning methods use neural network architectures, and for this reason deep learning models are often referred to as deep neural networks. The term *deep* usually refers to the number of hidden layers in the neural network. Conventional neural networks only contain 2-3 hidden layers, whilst deep networks have many more hidden layers. Most deep learning models are trained by using large sets of labelled data and learn features directly from the data without the need for manual feature extraction. Thus, deep learning provides the capabilities to analyse large and complex sets of features with little to no pre-processing compared when using traditional machine learning methods, where feature selection with large data is a necessary step for improving the performance of the machine learning model.

This paper hypothesises that by improving the quality of the data which is input into the deep learning model, the performance of the model will be improved. To test the hypothesis this paper presents a deep learning architecture which comprises two main components: Feature filtering and classification. Evaluations are carried out with and without the feature filtering component in order to evaluate the impact of feature filtering on classification performance. The features are reduced by applying feature filtering methods to identify the most important features (or highly ranked features) at the input level, prior to the data being fed into the classification model. Once the most important features are selected, these are fed into a deep learning classifier, and in particular the Convolutional Neural Network, which has been tuned for the particular task. The filtering methods which were used in the first component are: ReliefF [4], Fisher's score [5], and Local Learning-Based Clustering feature selection (LLC) [6]. The performance of Fisher score and its robustness to noise for different applications have widely been discussed in the literature [5], [7], [8]. In addition, theoretical and empirical performance analysis of Relief family algorithms as successful attribute estimators which are especially good in detecting conditional dependencies can be found in [4]. LLC is an unsupervised feature selection approach developed by Zeng and Cheung [6] which works within the framework of the Local Learning-Based Clustering [9].

The remaining of this paper is structured as follows. Section II discusses related methods; Section III provides a description of the proposed Convolutional Neural Network Architecture; Section IV explains the experiment methodology and results; and Section V provides a conclusion and future work.

II. Related methods

This section describes the methods which have been used to devise and experimentally evaluate the proposed architecture (see Section III). This section gives an insight into 1) the feature filtering methods used for reducing the features which are input into the deep learning classifier; and 2) the Convolutional Neural Network method which is the deep learning method proposed in this paper.

A. ReliefF Feature Filtering Algorithm

The original Relief algorithm first was developed by Kira and Rendell [10] in 1992. The Relief algorithm estimates the quality of attributes based on how well their values distinguish between instances that are near to each other. The first version of Relief was limited to binary classification problems, however two years after it was initially introduced, Kononenko [11] proposed an extension for the algorithm called ReliefF which is not limited to two class problems. ReliefF is more robust than Relief and can deal with incomplete and noisy data. The ReliefF algorithm randomly selects a sample x and then searches for k of its nearest neighbours from the same class called nearest hits and also k nearest neighbours from each of the different classes, called nearest misses. The quality estimation (i.e. weight) for each feature is updated by comparing within-class distance and between-class distance from neighbour samples. This procedure is repeated m times to obtain feature weighting vector. The formula by which the ReliefF algorithm updates the weight of features is:

$$W_f^i = W_f^{i-1} + \sum_{c \notin class(x)} \Big(\frac{\frac{p(x)}{1-p(class(x))} \sum_{j=1}^{k} diff_f(x, M_j(x)))}{m \times k} - \frac{\sum_{j=1}^{k} diff_f(x, H_j(x)}{m \times k} \Big) \tag{1}$$

where $diff_f(.)$ and $p(.)$ respectively denotes distance calculation and probability functions, $H_j(x)$ is the jth neighbour from the same class of sample x, and $M_j(x)$ shows jth neighbour from the different class of sample x. the pseudocode of RreliefF algorithm is presented in Algorithm (1).

Algorithm 1: ReliefF Algorithm

```
1  begin
2      for all samples do
3          Select a random sample $x$
4          Find $k$ nearest hits, $H_j$
5          for classes $C \notin$ class $x$ do
6              Find $k$ nearest misses for each class, $M_j(c)$
7          end
8          for all features do
9              Update weight $W_f^{(i+1)}$ using eq. 1
10         end
11     end
12 end
```

B. Fisher Score Feature Filtering Algorithm

Fisher discrimination analysis constructs a subset of features such that in the data space spanned by the features in the subset, the distances between samples from different classes are as large as possible, while the distances between samples from the same class are as small as possible. In particular, when m features are selected, the original data matrix $X \in R^{(d \times n)}$ will be represented by $Z \in R^{(m \times n)}$. Then, the Fisher score is computed as follows:

$$f(Z) = \frac{tr(S_b)}{tr(S_t)} \tag{2}$$

where $tr(.)$ denotes the trace of a matrix; S_b is the between-class scatter matrix; and S_t is the within-class scatter matrix, which is defined as:

$$S_b = \sum_{k=1}^{c} n_k(\mu_k - \mu)(\mu_k - \mu)^T \tag{3}$$

$$S_t = \sum_{i=1}^{n} (z_i - \mu)(z_i - \mu)^T \tag{4}$$

where μ_k and n_k are the mean and sample number of the kth class, respectively, in the reduced data space and $\mu = \sum_{k=1}^{c} n_k \mu_k$ is the overall mean vector of the reduced data. The number of candidate subsets is combination of $\binom{m}{d}$ so the optimal feature subset selection problem can be solved by combination optimisation, but this is highly challenging and computationally prohibitive for high dimensional data. To reduce the difficulty, a heuristic strategy is often used to calculate a score for each feature independently using some criterion f. Specifically, let μ_k^j and σ_k^j be the mean and deviation of samples from the kth class, corresponding to the jth feature. Let μ^j and σ^j denote the mean and deviation of the whole samples dataset corresponding to the jth feature. Then, the Fisher Score of the jth feature is calculated as follows:

$$f(j) = \frac{\sum_{k=1}^{c} n_k (\mu_k^j - \mu^j)^2}{\sum_{k=1}^{c} n_k (\sigma^j)^2} \tag{5}$$

where $(\sigma^j)^2 = \sum_{k=1}^{c} n_k (\sigma_k^j)^2$. After obtaining the Fisher Score for all features, m first features with highest scores are selected to construct the final feature subset. This procedure is shown in Algorithm (2).

Algorithm 2: Fisher Score

```
1 begin
2     for all features do
3         Calculate mean and deviation of the samples
          from each class
4         Calculate mean and deviation of all samples
5         Calculate $FisherScore$ using Eq. 5
6     end
7     Arrange features in descending order based on their
      $FisherScore$
8     Select the $m$ first features
9 end
```

C. Local Learning-Based Clustering Feature Filtering

The most important component of the Local Learning Clustering (LLC) is to learn the local regression model only trained with neighbourhood points [9]. LLC makes sure that the data points cluster labels are close to the predicted labels using the local regression model with their neighbouring points and their cluster labels [6]. For LLC feature selection, a weight is assigned to each feature and is taken into the built-in regularization of the LLC algorithm to consider the relevance of each feature for the clustering task. Each feature weight is determined by the magnitude of its corresponding element in the regression coefficients for all of the clusters locally solved at each point. If the feature's corresponding element in the regression coefficients has negligible magnitude for all clusters at each point, it indicates that the corresponding feature is likely unimportant while predicting the confidence of the cluster that the point belongs to. Accordingly, in the clustering process, the weights are iteratively estimated and updated until the optimal cluster indicator matrix is found via minimizing the overall prediction error.

D. Convolutional Neural Network

The Convolutional Neural Network (denoted as CNN, or ConvNet) is a variant of multi-layer neural networks that have been applied successfully in image classification and numerous other pattern recognition tasks [12]. A CNN is a feed forward network that can extract topological properties from data, and it is trained with a version of the back-propagation algorithm. CNNs are designed to recognise data patterns directly from raw data with little-to-none pre-processing. In practice, the design of a Convolutional Neural Network (CNN) is motivated by the discovery of the visual cortex in the brain. The visual cortex contains cells which are responsible for detecting light in the receptive fields which are small, overlapping sub-regions of the visual field. These cells act as local filters over the input space, and the more complex cells have larger receptive fields [13]. The past few years have shown pioneering results of CNN in many different domains such as image classification [12], [14], [15], object and face detection [16] on big image datasets and benchmark datasets.

A typical CNN is composed of many layers of hierarchy with some layers for feature representation (or feature maps) and others as a type of conventional neural networks for classification. It often starts with two altering types of layers called convolutional layers and pooling layers. In the hierarchical structure of a CNN, the convolutional layers perform convolution operations with several filter maps of equal size, while sub sampling layers reduce the sizes of proceeding layers by averaging pixels within a small neighborhood or by max-pooling.

III. Proposed Convolutional Neural Network-based architecture for biomedical data classification

This paper proposes a deep learning architecture based on the CNN for biomedical data classification. The proposed architecture consists of two components:

- **Component 1:** A feature filtering step to reduce the dimension and the complexity of the data and thus to contain a subset of data which only contains the most important features.
- **Component 2:** A deep CNN step for performing the feature extraction and classification task using the features identified after applying feature filtering.

The hypothesis is that by using feature filtering, the quality of the data which is input into the deep learning model is improved which also leads to a better performing deep learning model. The features are reduced by applying feature filtering methods to identify the most important ones (or highly ranked features) at the input level, prior to the data being fed into the prediction model. Once the most important features are selected, these are fed into a deep learning algorithm, and in particular the Convolutional Neural Network, which has been tuned for the particular task. The outcome will be a trained prediction model. The two components are described in the subsections that follow.

A. Component 1: Feature Filtering

The main purpose of the feature filtering step in the proposed approach is to reduce the complexity of the data and to identify most relevant features. Biomedical datasets (particularly gene expression and microarray datasets) mostly consist of thousands of features of which many of them could be highly correlated with others and consequently, the correlated features provide no extra information about the classes and thus add noise to the learning algorithm. Therefore, a feature filtering step is applied, before feeding the data into a CNN, to select the most discriminating features which can help CNN improve classification performance, reduce the complexity of classification model which can impact computational processing power. For this, a feature scoring algorithm is applied to rank all the features according to their assigned features. Next, based on a user-defined reduction rate, the most irrelevant, noisy and redundant features are removed and a new reduced dataset is formed. This new dataset is fed to the CNN algorithm for further processing. The reduction rate in this experiment is set to 80% meaning that 20% of the entire features in the dataset are kept as elite features, and the rest of the features are removed.

B. Component 2: Deep Convolutional Neural Network

The CNN used in the second component of the proposed architecture. The particular CNN was developed for the specific task, and it is composed of: two convolutional layers, one pooling layers and two fully connected layers with ReLU (Rectified Linear Unit). Four dropout performances are in the network. There are two dropout procedures after the two convolutional layers, and the other two dropouts are performed after the first and second fully connected layer. The size of the input layer is specified based on the number of features which are fed to the CNN from the first step. The first convolutional layer maps the input to the next layer with 64 feature maps. There are 64 filters in the first convolutional layer, and these filters map the input to 64 channels or filter maps. The length of the convolutional window of each kernel or filter is set to 1. Zero padding is used to have the same length of the original input in the outputs of the convolutional layer. The Rectified Linear Units (ReLU) approach is used as the activation function for the convolutional layer. The next layer is a regularization layer (Dropout) that is configured to randomly exclude 20 percent of neurons to reduce over-fitting. The Max-Pooling layer uses the maximum value to progressively reduce the spatial size of the representation to reduce the number of parameters and computation in the network, and also controls over-fitting. The length of the max pooling window is set to 2, keeps their maximum value and discards all other values. The next hidden layer is a convolutional layer that has a kernel size of 1 and contains 32 feature maps with a ReLU activation function. This layer is followed by another regularization layer (Dropout) that is the same as the previous layer. Afterwards, a layer namely the Flatten layer, converts the data to a vector, and that allowing the final output to be processed by standard fully connected layers to obtain the next layers. The first fully connected layer with the ReLU activation function contains 128 neurons. This is followed by a dropout layer to exclude 50% of neurons to reduce over-fitting. The second fully connected layer containing 32 neurons with the ReLU activation function also this layer followed by a dropout layer to exclude 50% of neurons. The final part of the CNN structure is the output layer which comprises a Softmax activation function and contains enough neurons to cover all class targets of the datasets. Fig. 1 shows the architecture of the proposed approach, and the parameter values set in this experiment are summarised in Table I.

TABLE I: The proposed deep CNN layers configuration

Layer Operation	Layer Configuration
Feature filtering	80% reduction Rate
Convolution	64 filters , 1x1 kernel and ReLU
Dropout	20%
Max-Pooling	2x2 kernel
Dropout	20%
Convolution	32 filters , 1x1 kernel and ReLU
Dropout	20%
Fully Connected	128 Neurons
Dropout	50%
Fully Connected	32 Neurons
Dropout	50%

IV. Experiment Methodology and Results

This section provides a description of the datasets and their properties; describes the evaluation measure used to evaluate the performance of the deep learning model when adopting the various feature filtering methods; and the results of the evaluations.

A. Datasets

Two datasets are used in this experiment namely: PANCAN and GAMETES Epistasis. Some properties of both datasets are available in Table II.

- **PANCAN:** The PANCAN dataset was created as part of the Pan-Cancer project, which aimed to assemble and analyse The Cancer Genome Atlas's of data across 12 tumor types. Until now, research has mostly focused on individual cancer types. The Pan-Cancer project is concerned with approaches to examine cancers based on their organ of origin and their genomic profiles. Analysis of this integrated dataset can provide a comprehensive information and insight of the molecular biology of multiple cancers, their similarities and differences in their genomic changes. The PANCAN dataset can be found in UCI machine learning repository [17] and is part of the RNA-Seq (HiSeq) PANCAN dataset. The frequency of the class labels of the PANCAN dataset is provided in Table III.
- **GAMETES Epistatis:** The GAMETES Epistatis dataset is a dataset which is simulated by a genetic-data simulation software called GAMETES. This software package generates epistatic patterns of association in 'mock' single nucleotide polymorphism (SNP) data [18]. The GAMETES Epistasis dataset is known to be of most

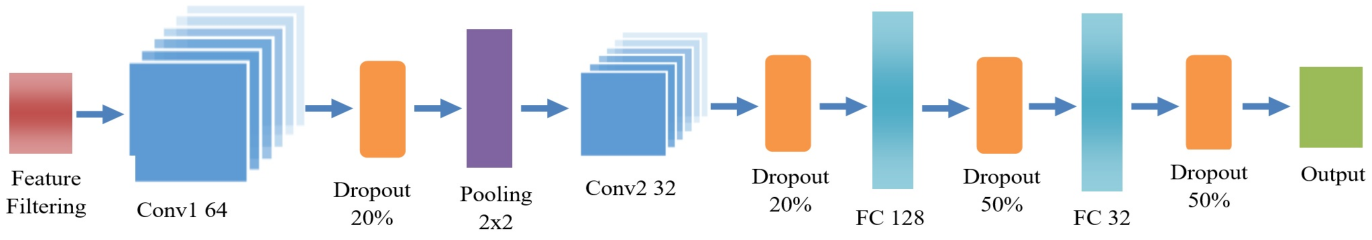

Fig. 1: Architecture of the proposed deep CNN

TABLE II: Dataset properties

Dataset Name	Type	# Features	# Instances	# Classes
PANCAN	Biomedical	20531	800	5
GAMETES	Biomedical	4100	1600	2

Class	# Samples	Percentage
1	136	16.98%
2	141	17.60%
3	300	37.45%
4	146	18.23%
5	78	9.74%

TABLE III: PANCAN dataset class frequency

difficult datasets due to amount of the noise which exists in the dataset as well as lack of univariate correlations between features and classes [19]. Table IV presents the frequency of the class labels in GAMETES dataset.

B. Evaluation Measure

The classification Accuracy is adopted to evaluate the performance of the proposed approach. Accuracy is simply the ratio of correctly predicted observations to the total observations. The function for calculating Accuracy is found in Equation (6).

$$Accuracy = \frac{TCC}{TCC + FCC} \times 100, \qquad (6)$$

where TCC, is the total number of cases which are correctly classified; and FCC, is the total number of cases which are incorrectly classified.

C. Experimental Results

The proposed deep CNN approach is trained and tested against the PANCAN and GAMETES datasets, and the models are implemented in the Python programming language using Theano and Keras libraries. After initial feature filtering using a feature filtering algorithm in the first layer (i.e. first component of the proposed architecture) (Fig. 1), the top 20% of the features are kept to construct the reduced dataset and the rest are removed. It is worth highlighting that the feature filtering methods are very fast, and each took a couple of seconds on average running time to return the top ranked 20% of features. To unambiguously refer to a reduced dataset, each dataset is denoted by the filtering method + dataset name, eg., FisherScore+PANCAN which refers to the reduced PANCAN dataset after applying the Fisher score filtering method. The reduced dataset is then fed to the CNN. The reduced dataset is split into a 70:30 ratio to form the training and testing sets, respectively.

Class	# Samples	Percentage
0	800	50.00%
1	800	50.00%

TABLE IV: GAMETES dataset class frequency

PANCAN dataset results: Table V presents the results of the proposed CNN architecture applied to the PANCAN dataset. The experiment results show that the CNN architecture achieved 100.00% accuracy when using the ReliefF feature filtering technique, which was 20% higher than the performance obtained by applying the CNN architecture on the original dataset (i.e. 80.00%). The confusion matrix of the best performing CNN model, i.e. FisherScore+PANCAN, trained with 30 epochs is shown in Fig. 2. The diagonal elements of the confusion matrices represent the number of data which were correctly classified, while off-diagonal elements are those that are mislabelled by the classifier. The higher the diagonal values of the confusion matrix, the better the classification performance. Fig. 3 and Fig. 4 depict the model accuracy and training loss when adopting the FisherScore+PANCAN which was the best CNN model. These figures show that training and testing performance were close together during different epochs, which indicates that the FisherScore+PANCAN CNN model was not over-fitting the data. The training times have also been reported in Table V which show that the reduction in features at the input level has resulted in reduced training time of approximately 5 seconds.

GAMETES dataset results: Table VI presents the results of the proposed CNN architecture applied to the GAMETES dataset. The experiment results show that the CNN architecture achieved 58.96% accuracy when using the Fisher feature filtering technique, which was 10.63% higher than the performance obtained by applying the CNN architecture on the original dataset (i.e. 48.33%). The confusion matrix of the FisherScore+GAMETES CNN model trained with 30 epochs is shown in Fig. 5. Fig. 6 and Fig. 7 depict the model accuracy and training loss when adopting FisherScore+GAMETES CNN proposed model. These figures show that training and testing performance were close together up to epoch 10 and after that the CNN started to overfit the data, and hence the gap in the training and testing accuracies. The training times can also be found in the last column of Table VI. The main aim is to investigate whether filter filtering improves the performance

Dataset Name	#Features	#Instances	#Classes	#Training set cases	#Test set cases	Accuracy	Epochs	Training time (s)
PANCAN Original	20531	800	5	560	240	80%	10	12.17
FisherScore+PANCAN	4100	800	5	560	240	98.5%	30	7.44
LLC+PANCAN	4100	800	5	560	240	91.17%	30	7.11
ReliefF+PANCAN	4100	800	5	560	240	100%	30	7.06

TABLE V: PANCAN dataset classification results

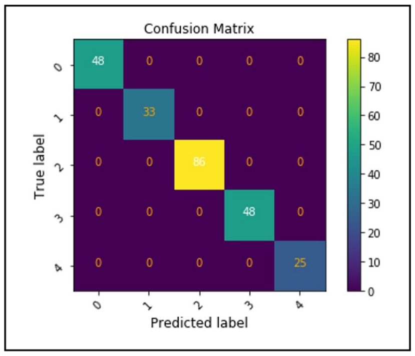

Fig. 2: Confusion matrix on the FisherScore+PANCAN dataset

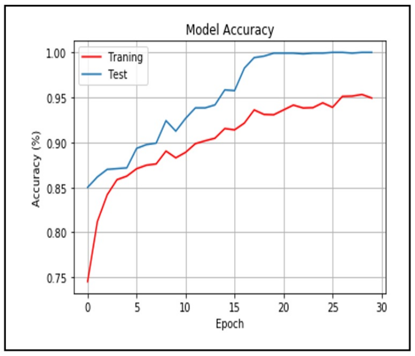

Fig. 3: FisherScore+PANCAN training and testing Accuracies

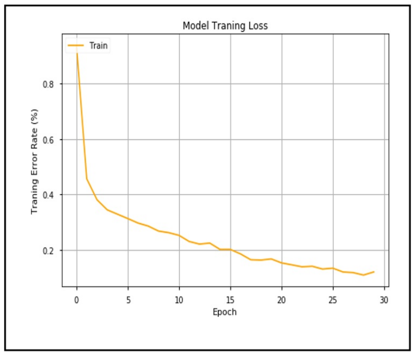

Fig. 4: FisherScore+PANCAN dataset loss during the CNN training process

of the Convolutional Neural Network and experiments with the GAMETES dataset show clearly demonstrate the benefit of applying feature filtering. The training times have also been reported in Table VI which show that the reduction in features at the input level has resulted in reduced training time of approximately 2 seconds.

V. Conclusion and Future Work

The paper evaluates the hypothesis that by using feature filtering, the quality of the data which is input into the deep learning model is improved which also leads to a better performing deep learning model. Two biomedical datasets, namely the PANCAN and GAMETES Epistasis datasets, which contain a large number of biological features but also large enough number of patient cases were selected for the evaluations. It was challenging to identify datasets suitable for the task, as most datasets contained either a low number of features and many patient cases or a large number of features and few patient cases. A selection of feature filtering methods were applied to identify the most important features (i.e. 20% of most highly ranked features extracted from each dataset) at the input level, prior to the data being fed into the prediction model. Once the most important features were selected, these were fed into a deep learning algorithm, and in particular the Convolutional Neural Network, which has been tuned for the particular task. Experiment results demonstrate that applying feature filtering at the input level improves the performance of deep learning models, even for the most complex biomedical data such as those utilised in this paper. In particular, for the first dataset, PANCAN, an improvement of 20% was reported in Accuracy, whereas for the second dataset GAMETES Epistasis, an improvement of 10.63% was reported in Accuracy. The results are promising and demonstrate the benefits of filter filtering when deep learning methods are adopted for biomedical classification tasks.

Future work includes performing experiments with more biomedical datasets to further evaluate the impact of feature filtering on the performance of the Convolutional Neural Network and other deep learning models.

VI. Acknowledgement

Dr Cosma acknowledges the financial support of The Leverhulme Trust (Research Project Grant RPG- 2016-252).

References

[1] B. D. McKinsey, “The next frontier for innovation, competition, and productivity,” *McKinsey Global Institute Report*, 2011.

[2] P. Larranaga, B. Calvo, R. Santana, C. Bielza, J. Galdiano, I. Inza, J. A. Lozano, R. Armananzas, G. Santafé, A. Pérez *et al.*, “Machine learning in bioinformatics,” *Briefings in bioinformatics*, pp. 86–112, 2006.

[3] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[4] M. Robnik-Šikonja and I. Kononenko, “Theoretical and empirical analysis of relieff and rrelieff,” *Machine learning*, vol. 53, no. 1-2, pp. 23–69, 2003.

[5] W. Malina, “On an extended fisher criterion for feature selection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 5, pp. 611–614, 1981.

[6] H. Zeng and Y.-m. Cheung, “Feature selection and kernel learning for local learning-based clustering,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 8, pp. 1532–1547, 2011.

[7] S. Olyaee, Z. Dashtban, and M. H. Dashtban, “Design and implementation of super-heterodyne nano-metrology circuits,” *Frontiers of Optoelectronics*, vol. 6, no. 3, pp. 318–326, 2013.

[8] J. Xuan, Y. Wang, Y. Dong, Y. Feng, B. Wang, J. Khan, M. Bakay, Z. Wang, L. Pachman, S. Winokur *et al.*, “Gene selection for multiclass prediction by weighted fisher criterion,” *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 2007, no. 1, p. 64628, 2007.

[9] M. Wu and B. Schölkopf, “A local learning approach for clustering,” in *Advances in neural information processing systems*, 2007, pp. 1529–1536.

[10] K. Kira and L. A. Rendell, “A practical approach to feature selection,” in *Proceedings of the ninth international workshop on Machine learning*, 1992, pp. 249–256.

[11] I. Kononenko, “Estimating attributes: analysis and extensions of relief,” in *European conference on machine learning*. Springer, 1994, pp. 171–182.

Dataset Name	#Features	#Instances	#Classes	#Training set cases	#Test set cases	Accuracy	Epochs	Training time(s)
GAMETES Original	1000	1600	2	1120	480	48.33%	10	8.52
FisherScore+GAMETES	200	1600	2	1120	480	58.96%	30	6.53
LLC+GAMETES	200	1600	2	1120	480	51.04%	30	6.47
ReliefF+GAMETES	200	1600	2	1120	480	49.38%	30	6.39

TABLE VI: GAMETES dataset classification results

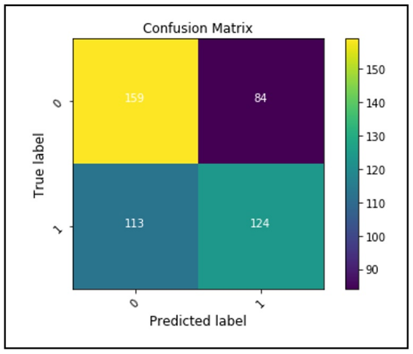

Fig. 5: Confusion matrix on the FisherScore+GAMETES dataset

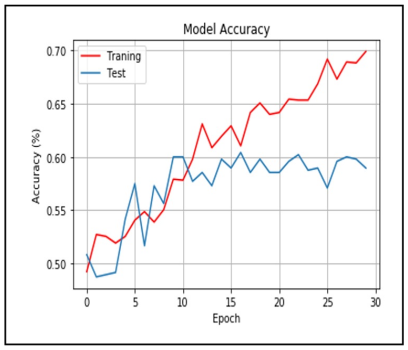

Fig. 6: FisherScore+GAMETES training and testing Accuracies

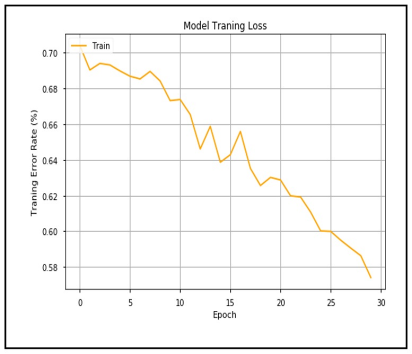

Fig. 7: FisherScore+GAMETES loss during the CNN training process

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
[13] S. Hijazi, R. Kumar, and C. Rowen, "Using convolutional neural networks for image recognition," *Cadence Design Systems Inc.: San Jose, CA, USA*, 2015.
[14] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.
[15] A. A. Alani, "Arabic handwritten digit recognition based on restricted boltzmann machine and convolutional neural networks," *Information*, vol. 8, no. 4, p. 142, 2017.
[16] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," in *Advances in neural information processing systems*, 2013, pp. 2553–2561.
[17] A. Asuncion and D. Newman, "Uci machine learning repository," 2007.
[18] R. J. Urbanowicz, J. Kiralis, N. A. Sinnott-Armstrong, T. Heberling, J. M. Fisher, and J. H. Moore, "Gametes: a fast, direct algorithm for generating pure, strict, epistatic models with random architectures," *BioData mining*, vol. 5, no. 1, p. 16, 2012.
[19] R. S. Olson, W. La Cava, P. Orzechowski, R. J. Urbanowicz, and J. H. Moore, "Pmlb: a large benchmark suite for machine learning evaluation and comparison," *BioData mining*, vol. 10, no. 1, p. 36, 2017.