"Lymphoma Subtype Classification using Convolutional Neural Networks"

Filippo Canderle[†], Davide Colussi[‡]

***Abstract*—Deep learning and Neural networks represent one of the most powerful tools that we have to fight and prevent diseases. These tools will be increasingly used in precision medicine and will represent an aid and support to the healthcare personnel's decisions. In this work we have considered a dataset containing images related to three types of lymphomas. In the contest of supervised learning, we trained a model to automatically recognize the correct class among the three. In particular, we created four different convolutional neural networks with different characteristics to evaluate the different classification performances on the dataset. After choosing the best network hyperparameters for our purposes, our best model obtained an accuracy of 0,93.**

***Index Terms*—Convolutional Neural Network, Multiclass Classification, Lymphoma classification, Histology.**

I. Introduction

Lymphomas are types of cancer that begin in the lymphatic system when abnormal white blood cells grow. Lymphomas are the sixth most common form of cancer overall (excluding non-melanoma skin cancer). The treatment and the chance of a cure depend on the lymphoma subtype and the stage. The three most common subtypes are chronic lymphocytic leukemia (CLL), follicular lymphoma (FL) and mantle cell lymphoma (MCL).

Some different approaches in automating pathology lymphoma classification using digital images already exists in the literature. In [1] some set of transform are applied to the image, then a classification is done by involving different color planes, using Weighted Neighbor Distance (WND), naïve Bayes network (BBN) and radial basis functions (RBF) classifiers. In [2] deep learning is applied to digital pathology (DP), with the aim of creating a domain agnostic approach combining both feature discovery and implementation. In this paper an already pre-existing neural network model have been used: "AlexNet" [3]. Also in [4] a deep learning approach is used to discriminate between the three main lymphoma subtypes.

In this paper different models are explored to evaluate which one performs better in classifying the three main lymphoma subtype. Instead of using statistical classifiers, as in [1], we we use neural networks (NNs) to accomplish the classification task. In particular, some different flavors of convolutional neural networks (CNNs) will be employed, since in the current state of the art they turn out to be one of the best strategies when dealing with image classification. Instead of [2], which uses a pre-existing NN model to accomplish the task, we created and evaluated four different CNN models from scratch, by tackling various hyperparameters and layers structure. This work is meant to be an example where different models have been created and evaluated to discover which one gets the highest classification accuracy. This to show that we can always use new models to solve the problem. In the future our work could undergo new improvements and our approach could be used as a starting point for new researches.

This report is structured as follows: In Section II the state of art is described; in Section III the main processing pipeline is presented from a high-level perspective, while in Section IV the data, preprocessing and features are described; In Section V the learning framework its shown in detail, comparing the different models used; The performance evaluation of the four models is explained in Section VI, while the conclusions are reported in Section VII.

II. Related Work

In the last years different solutions were implemented for lymphoma subtype classification. In the work of [1] there is shown a way to classify the three main types of Lymphomas (CLL, FL and MCL) by first extracting relevant features from the lymphoma image by transforming the original signal with a set of transforms into spectral planes. Simple (Fourier, Chebyshev, and wavelets) and compound transforms (Chebyshev of Fourier and wavelets of Fourier) have been applied. In this approach no segmentation of the image is done before, but different color planes are involved: RGB, gray, Lab, H&E (hematoxylin and eosin stains). The classification is done using three classifiers: Weighted Neighbor Distance (WND), naïve Bayes network (BBN) and radial basis functions (RBF). They found that the strongest signal is contained in a histological (HE) color scheme, with the original (RGB) scheme giving measurably worse performance, indicating that classification is sensitive to biologically relevant morphologies. The HE set of channels results in the best classification accuracy on three lymphoma subtypes (88% for BBN and 98% for the other two classifiers), the RGB set gave the second-best result (90%). They found that the WND classifier demonstrated the best overall performance with RBF performing very similarly, and with BBN never reporting the best accuracy. While [1] focuses its work on comparing the performance using different statistical classifier, our work will focus on the use of neural netwrorks to address the same classification problem.

[†]Department of Information Engineering, University of Padova, email: {filippo.canderle}@studenti.unipd.it

[‡]Department of Information Engineering, University of Padova, email: {davide.colussi}@studenti.unipd.it

In [2] a deep learning approach is involved in the following tasks: nuclei segmentation, epithelium segmentation, tubule segmentation, lymphocyte detection, mitosis detection, invasive ductal carcinoma detection and lymphoma classification. The cited paper uses a dataset of 374 images where three types of lymphoma are present (CLL, FL and MCL). In the paper they show the use of an already existing CNN model called "AlexNet", developed in [3], which is found in the deep learning framework "Caffe", for the lymphoma classification. This is a 8 layer convolutional neural network, in which the first two layer are convolutional layer with 32 filters, the first one followed by a max pooling layer and the second one followed by a mean pooling layer. The third layer is a convolutional layer with filter size of 64, followed by two dense layers and a softmax activation function. The images are split into sub-patches, so that the model could classify each patch indipendently, in a "winner-takes-all" fashion: the class with the highest number of vote became the designated class for the entire image. The accuracy of their model on the lymphoma classification task is 96.58%. Typically, one needs to make various decisions to design an appropriate network such as input patch size, number of layers, and convolutional attributes. In [2] they attempt to mitigate this dependency by instead opting to leverage the popular and successful "AlexNet" network. In our paper, instead, we extend [2] by trying different CNNs configurations. We also applied patch splitting to our dataset images, taking inspiration by [2], as it was a nice technique to increase the training set size.

In [4] they still address the problem of lymphoma subtype classification among the three main subtypes. The method proposed in their work, which yields an accuracy of 97.33 percent on the NIA curated dataset, shows that deep learning has the potential to aid an automated diagnosis for lymphoma. An "Inception V3 network" is used, which is an image recognition model that has been shown to attain greater than 78.1% accuracy on the "ImageNet" dataset. The model is the culmination of many ideas developed by multiple researchers over the years. Also, several branches are used to determine the appropriate type of convolution to be made at each layer. [4], as [2], still makes use of a pre-existing neural network model, we have instead tried to implement by ourselves new models to accomplish the lymphoma classification task.

III. Processing Pipeline

In this paper we use a dataset with lymphoma digital images, each one labeled with the lymphoma subtype. The images, with their labels, are used to teach a machine learning algorithm to distinguish between the various classes. Each image is first preprocessed, then transformed into a feature vector. The feature vectors contains the caratheristics of an image, which will serve the algorithm to be able to learn to distinguish which characteristics are associated with each lymphoma subtype. This allows us to create a predictive model which, given a new image that was never seen by the algorithm, is able to predict to what class it belongs to.

In the preprocessing step we normalize and process the original data to make the algorithm work better. The specific preprocessing steps will be described more specifically in IV. This is a case of supervised learning. The processing pipeline is summarized in the Fig. 1.

IV. Signals and Features

A dataset with 374 labeled RGB images (113 for CLL, 139 for FL and 122 for MCL) of 1040x1388 pixels was used.

A. Preprocessing - Patches extraction and color channel resclaing

Since neural networks needs several training examples to perform good, instead of working with the whole images directly we divide them into sub-images which we will call patches, of size a x b. This allows us to have more training samples and better results. The total number of patches extracted from the dataset is obtained as follows:

$$N = \lfloor (m - a + 1) * (n - b + 1) \rfloor rK \tag{1}$$

where m is image height, n is image width, a is patch height, b is patch width, $r \in]0, 1]$ is a proportion of the total possible maximum number of patches, K is the total number of images in the dataset. By changing r we are able to select less patches, by taking them randomly across the all possible ones.

Each color channel (red/green/blue) ranges from 0 to 255, a rescaling by dividing by 255 has been done, obtaining a color value which goes from 0 to 1. In this way, the numbers will be small and the computation done by the framework becomes easier and faster.

B. Training and test vectors

In the contest of supervised learning we need a feature vector X and a label vector Y, so that the network can be trained on recognizing the class based on the features. In our case X has a shape of N x a x b x 3 (where 3 represents the number of color channels) and Y has a shape of N x 3 where each entry is a "one-hot" encoded lable: $(1, 0, 0)$ for CLL, $(0, 1, 0)$ for FL and $(0, 0, 1)$ for MCL.

A typical approach in machine learning is not to evaluate the performance of the model just on the training data, instead accuracy on some new data, which the network has never seen, is usually used. This type of data is called the test data. By following a typical approach, by just using the original dataset we can split it into training and test data. In our case X and Y have been split into training and test vectors with a proportion respectively of 80% and 20% of the total number of patches N, obtaining four vectors: $XTrain$, $YTrain$, $XTest$, $YTest$.

For the patches, we found out that 32x32 pixels was a suitable size to work on, as it behaved well on the neural network models first tests.

In order to keep an affordable computational complexity we can set, empirically, different values for r in (1) for the four CNNs, leting us set the total number of patches N used for the network models. We used a fairly low percentage of the

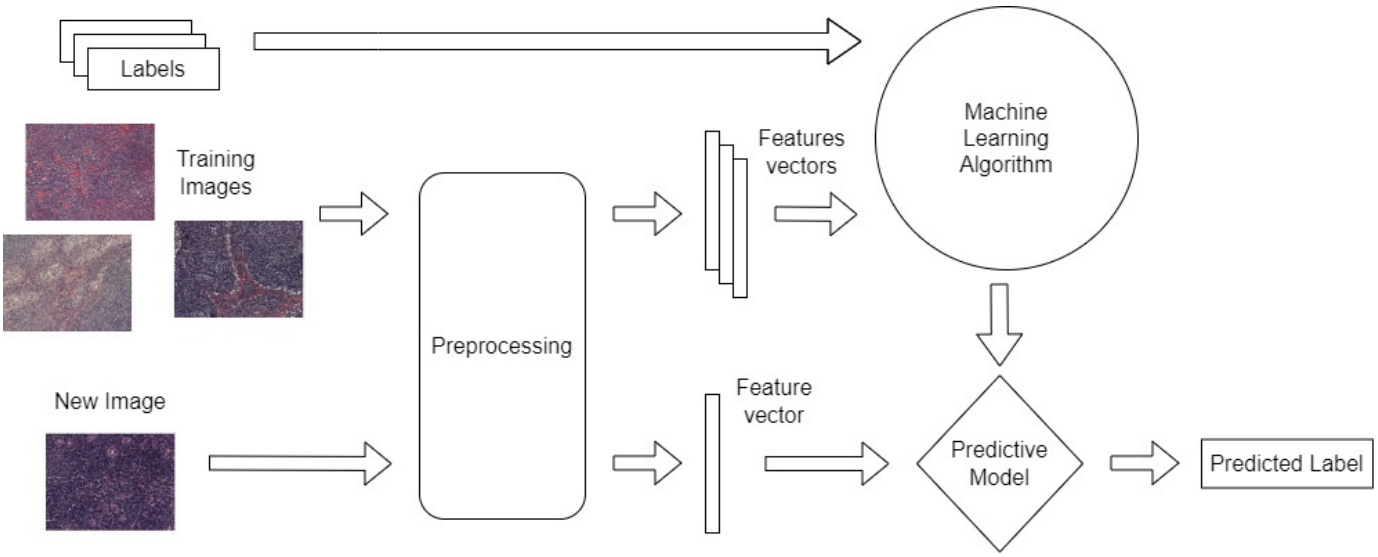


Fig. 1: Supervised learning paradigm.

maximum possible number of patches to maintain a correct compromise between overfit risks and computational cost of the training procedure. Here are the values which we have empirically found, that were used in the four networks (recall that in section V we will explain in detail the four networks involved):

CNN1 and CNN2:

- $r = 0.0015 \implies N = 767822$
- $XTrain$ shape: $(614257, 32, 32, 3)$
- $YTrain$ shape: $(614257, 3)$
- $XTest$ shape: $(153564, 32, 32, 3)$
- $YTest$ shape: $(153564, 3)$

CNN3 and CNN4:

- $r = 0.00015 \implies N = 76670$
- $XTrain$ shape: $(61336, 32, 32, 3)$
- $YTrain$ shape: $(61336, 3)$
- $XTest$ shape: $(15334, 32, 32, 3)$
- $YTest$ shape: $(15334, 3)$

V. Learning Framework

Our learning framework is composed by four different convolutional neural networks, as these type of networks are widely used for image classification tasks. We implemented each network differently, with layers and hyperparameters. The layers used in the different networks are described below:

- **Input layer**: a N x 32 x 32 x 3 vector
- **Convolution layer**: a 2D spatial convolution using f filters of size k x k. This layer performs a dot product of the convolution filter with the layer's input and generate some feature maps. This allows the model to extract visual, local and global, features of the image. In our framework the size of the filters is 3 x 3, while the number of filters differs in each network.
- **Batch normalization**: allows a faster and more stable learning by applying a transformation that maintains the mean output close to 0 and the output standard deviation close to 1.
- **Max pooling**: reduces the dimensions of the feature maps generated by a convolutional layer. This operation selects the maximum element from the region of the feature map covered by the filter. Here, a 2 x 2 filter with a 2 x 2 stride has been used.
- **Dense layer**: a layer where neurons are fully connected to every neuron of its preceding layer.
- **ReLU**: a piecewise linear activation function that will output the input directly if it is positive, otherwise, it will output zero.
- **Softmax**: an activation function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector.

$$\sigma(y_i) = \left(\frac{e^{y_i}}{\sum_j e^{y_j}} \right) j = 1, ..., n \tag{2}$$

In our case, since we are classifying among three classes, n=3. This activation function is used at the last layer of each network.

A. The four network at comparison

The first network, called CNN1, is composed by three convolutional layers with respectively 48, 48 and 32 filters. Each convolutional layer is followed by batch normalization and max pooling layers. Then, there are two dense layer, each followed by a batch normalization layer. The last layer is a dense layer composed by 3 neurons, followed by a softmax activation function which outputs a vector of dimension 3, in which each component indicates the predicted probability of the input patch to be part of a CLL, FL or MCL lymphoma subtype.

The CNN2 is composed by two convolutional layers with respectively 32 and 64 filters. Each convolutional layer is followed by a max pooling layer. Then a dense layer composed of 128 neurons is applied, followed by a 3 neurons dense layer which output, with softmax, the predicted labels probabilities.

In CNN3 we use just one convolutional layer of 48 filters, followed by batch normalization and max pooling layers. Just like the other networks, the output is given by a 3 dense layer followed by softmax.

CNN4 starts similar to the CNN3, but adds a 64 dense layer followed by batch normalization before the 3 dense layer of CNN3. The structures of the four CNNs is well depicted in Fig. 2.

B. Selecting appropriate number of epochs and batch sizes for each model

During the initial comparison between the different models, the CNN1 was the one that seemed to obtain better results, this implied that the major effort in hyperparameters optimization was done especially for this particular model. We found that the best values of CNN1 for training epochs and batch size was respectively 8 and 64. For the other CNNs, the epochs and batch size were chosen to achieve a suitable training time and good performance on the validation set, resulting in the values in the following table:

CNN	epochs	batch size
1	8	64
2	8	64
3	100	128
4	6	128

C. Training and testing procedure

In the learning phase, the image dataset is used to train the networks. The images are preprocessed as described in section IV by first creating 32 x 32 patches of each image, then by normalizing the pixel values. Each model has been trained with the loss crossentropy function, which is what is usually used in multiclass classification and "Adam" seemed to be the gradient that worked best for our purposes. We used accuracy as performance metric.

Each network is then tested against the test set composed by images from the dataset that the trained networks have never seen. These images are first subdivided in patches, then each patch is fed to the model, which outputs the predicted label for each patch. The final prediction for the whole image is done with a "winner-takes-all" approach: the predicted label for the whole image is obtained by selecting the label which is more probable within all the patches predictions.

D. Other models

We also tried to work with autoencoders, where the idea was to extract features common all the elements of a class in the central part of the autoencoder. Unfortunately, the performance of our implementations has not given the expected results, so our work has focused more on convolutional neural networks.

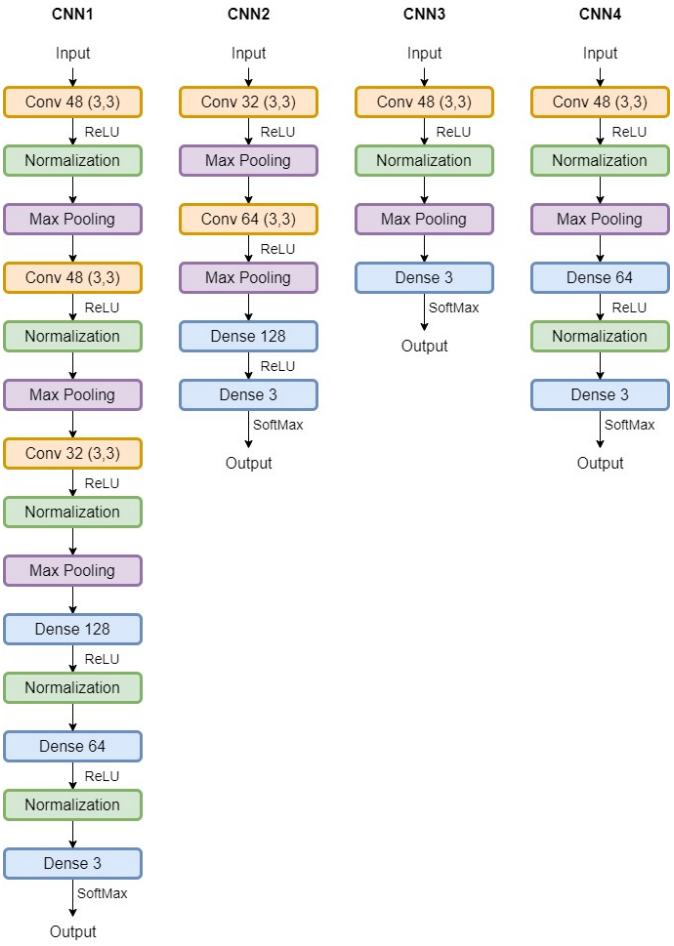


Fig. 2: Structures of the four networks used.

E. Technology Architecture

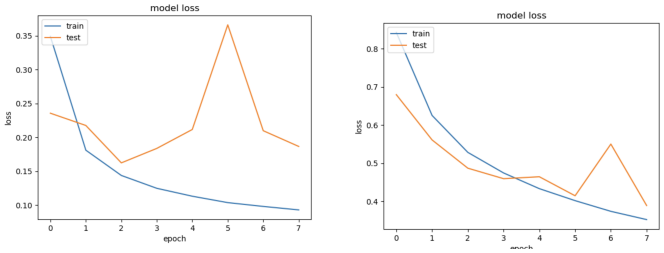
In our practical implementation we wrote all the Code in Python, using Keras and scikit-learn libraries, widely used in Machine Learning. More in details:

- *Keras* was used to prototype all the modules related to Convolutional Neural Networks.
- *Scikit-learn* focused more in the first part of our pipeline, where we had to deal with the dataset.

VI. Results

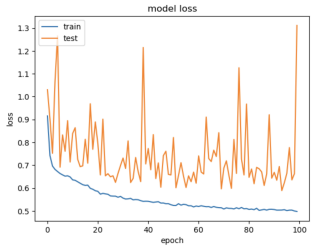
We have trained the four networks and plotted the training and test loss for each epoch (Fig. 3). The models have been trained until they started to show sign of overfitting (low training loss but high test loss). CNN1 and CNN3 needed to be trained for a considerably higher amount of time before starting to overfit, with respect to CNN2 and CNN4.

We can see in the table "Accuracy of the model", that the network which gets better results in the accuracy of the test set is CNN1 $(0, 93)$. The CNN2 got a lower score (0.84), but the training time required was much lower. The CNN3 got an accuracy of $0, 73$ on the test set, with a training time of almost one hour, which reveals that is not a good model for our purposes. The CNN4 got an high accuracy on the training

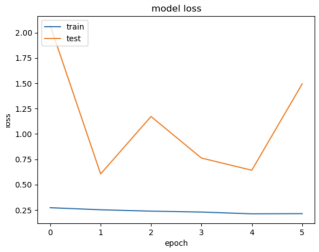


(a) CNN1

(b) CNN2

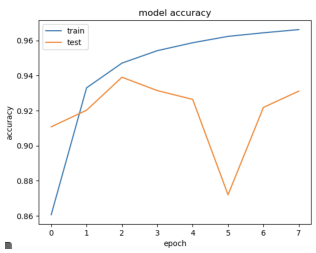


(c) CNN3

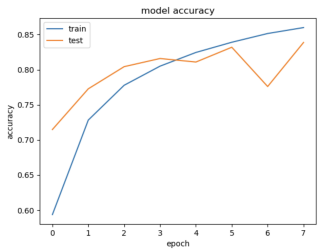


(d) CNN4

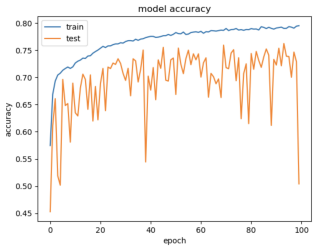
Fig. 3: Train vs test loss



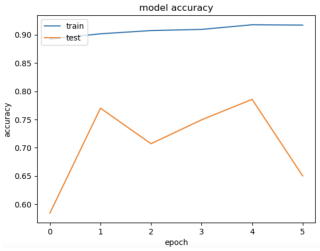
(a) CNN1



(b) CNN2



(c) CNN3



(d) CNN4

Fig. 4: Train vs test accuracy

set $(0, 92)$, with a low training time, but unfortunately fails to generalize (test accuracy is only $0, 65$, which probably means it is overfitting).

Accuracy of the model			
model	training set	test set	n. of samples
CNN1	0,96	0,93	767822
CNN2	0,86	0,84	767822
CNN3	0,79	0,73	76670
CNN4	0,92	0.65	76670

The computational cost of the models highly depends on the total number of trainable parameters and the number of training epochs. Here is a table indicating the number of trainable parameters and the training time for each model:

Computational cost		
model	trainable parameters	training time
CNN1	61587	2:26:21
CNN2	314819	0:05:02
CNN3	33843	1:05:00
CNN4	693027	0:03:51

Given that the training must be done only once, it makes sense to spend a few more hours on the training, as long as good accuracy is maintained and overfit avoided. In fact, the two CNNs trained with about 700000 image patches (CNN1 and CNN2) have a much lower train-test accuracy deviation than the two CNNs towed with about 70000 image patches (CNN3 and CNN4), as we can see in Fig. 4. Showing that CNN1 and CNN2 tend to overfit less.

VII. Concluding Remarks

In conclusion, what we wanted to implement in this work was a model able to accurately recognize and classify the three main types of lymphomas. We believe that this field of AI applied on the healthcare scenarios can have a great applicability in the future, with the goal of helping healthcare personell to make correct decisions. Any tool that can help improve people's health and save lives should be taken into consideration.

The four CNNs we have created do not want to be precise and "definitive" models, but want instead to be a comparison between four differently structured models and what could be the best starting point from which to build a truly "definitive" model. Here we only experimented with a few models and we shown what the potential problems could be when you implement a CNN for these purposes.

We tried to implement autoencoders in attempt to reach the same goals in a different manner. The main idea was to identify "fundamental" features in the encoding part, but we were not able to achieve satisfactory results and therefore we abandoned this idea.

We could also have used data augmentation techniques (rotating images and/or adding salt and pepper noise) in order to increase the number of available images and make the models be able to generalize better. By the way, in this work we used a simpler handling of our dataset. In further improvements of our work, image normalization in the preprocessing step could also be applied, since the images of the dataset have been probably acquired in different environment conditions and with different instruments.

What we personally learned from this experience was to tackle a problem of this kind from scratch and, in the case of one of the two authors, work with the Keras library for the first time. We faced the most typical problems of script writing, applied in particular to this machine learning context, and in drafting this document we tried to comply with the standards of a paper.

REFERENCES

[1] N. V. Orlov, W. W. Chen, D. M. Eckley, T. J. Macura, L. Shamir, E. S. Jaffe, and I. G. Goldberg, "Automatic classification of lymphoma images with transform-based global features," *IEEE transactions on information technology in biomedicine*, vol. 14, pp. 1003–1013, July 2010.

[2] A. Janowczyk and A. Madabhushi, "Deep learning for digital pathology image analysis: A comprehensive tutorial with selected usecases," *Journal of Pathology Informatics*, vol. 7, p. 29, July 2016.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[4] R. Tambe, S. Mahajan, U. Shah, M. Agrawal, and B. Garware, "Towards designing an automated classification of lymphoma subtypes using deep neural networks," in *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data (CoDS-COMAD 2019)*, (Association for Computing Machinery, New York, NY, USA), Jan. 2019.