Author identification

Praveen Baburao Kulkarni
KOGS
University of Hamburg
Hamburg, Germany
4kulkarn@informatik.uni-hamburg.de

***Abstract*—In this paper, a framework for writer identification and different components needed to build it will be presented. The system uses five different kinds of features (Curvature, Directional, Edge Based Chain code and Tortuosity) extracted from the images. These features can then be used for classification using different classifier algorithms. We observed that changing the feature representation and then applying training classifiers improves the results considerably as it helps untangle and reveal the different explanatory factors of variation behind the data. In our work, we used two strategies for changing input feature representation namely dimensionality reduction and representation learning.**

Two major contributions of this paper are:

- **Using Restricted Boltzmann Machines (RBMs) for representation learning in context of writer identification task and comparing it with Latent Dirichlet Allocation**
- **Design of parallel framework for writer identification task**

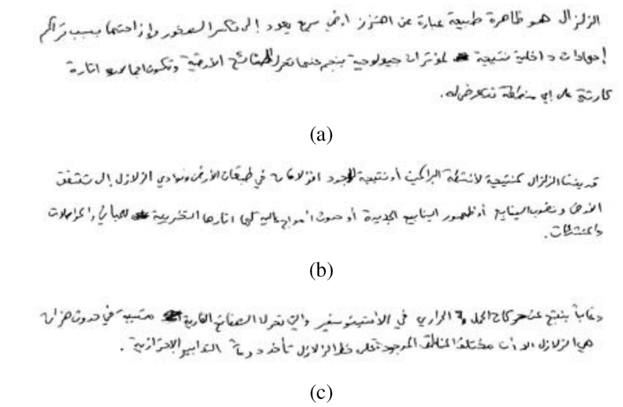
We report classification accuracy of around 90 percent by doing dimensionality reduction with Linear Support Vector Classifier followed by classification using Linear Discriminant Analysis (LDA). On the other hand, 89.7 percent accuracy was observed by doing representation learning using RBMs followed by LDA for classification. The classification accuracy is equal to the state of art implementations which do not use RBMs. At the end of the paper, the parallel framework that was designed for writer identification and how all the components fit inside it will be presented.

***Keywords*-Writer identification; dimension reduction; representation learning; Restricted Boltzmann machines; parallel framework; Linear Discriminant Analysis**

I. Introduction

Writer identification is of high importance for forensic document examination where it can serve as evidence (e.g. a will or ransom note) [1]. It is used in medicine where the prescription has to come from authorized person and in banks it can be used for signature verification. It can also be used for the analysis of ancient documents so that indexing and retrieval are possible. Despite the development of electronic documents and use of printed text, the importance of handwritten documents has retained its place because it carries additional information of the person who has written it.

For our experiments we have used ICFHR 2012 Writer Identification Competition dataset [2]. There are 206 writers with three paragraphs each. Two are used for training and the remaining one is used for testing. The first two paragraphs have been removed from the training set for some writers in order to check the systems capability to reject such writers which do not belong to training dataset.



(a)

(b)

(c)

Figure 1: Handwritten text by a single writer

Writer identification systems can be designed using codebook-based [3] or feature-based approaches [4] [5]. Codebook-based approaches use the occurrence histograms of the shapes in the codebook. These shapes are extracted from small fragments of connected components obtained from the handwriting. On the other hand, feature-based approaches use some structural, geometrical or textural features in the handwriting. Feature-based approaches are generally preferred when there is a limited amount of handwriting available. In Section II we will discuss in detail the kind of features extracted for the writer identification task.

In Section III we will discuss different classifiers that we experimented with and report the accuracy. Further in Section IV we will discuss how improving the feature representation improved the accuracy of classifiers.

Experimenting with different classifiers and features, along with analyzing the different feature representation techniques requires a lot of computational power. But, on the other hand, much of these tasks are independent and can be run in parallel. In Section VI we will discuss how various components that are needed for building writer identification system, can be put together in a parallel framework. Finally in Section VII we will report and analyze the accuracy of different approaches.

II. Feature extraction

The writer identification system we propose consists of three major steps: a) feature extraction, b) learning new feature representation, and c) training models. In this section, we will discuss the type of features extracted as proposed by Abdelaali et al [5]. In writer identification task the features extracted are represented as a probability distribution function (PDF) instead of single value [6]. Below is the list of features that are extracted from the handwritten documents:

1) Turtuosity features
2) Curvature features
3) Directional features
4) Edge based features
5) Chain code features

Chain code features as proposed by Siddiqi and Vincent [7] achieve high identification rate on IAM database [8]. These features along with edge-based features can be efficiently combined with new features based on directions, curvatures and tortuosities as shown by Abdelaali et al [5]. Chain code features are generated by tracing the contour and assigning a number to each pixel based on its location with respect to the previous pixel. Figure 2 shows an example for chain code features.

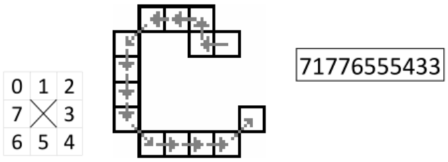


Figure 2: Contour and its corresponding chaincode [5]

Directional features are extracted by computing Zhang skeleton of the binarized images. This skeleton is then segmented at its junction. Then we move along these segments in a predefined order and perform a linear regression of these pixels to get a good estimate of the tangent. Figure 2(fig:directional) shows the segments in green color. Linear regression is performed for the green pixels in every segment to estimate the local direction.

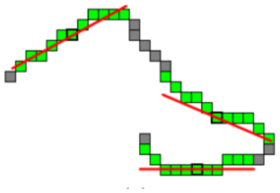


Figure 3: Directional features (tangent at a pixel) [5]

Curvature features are computed by estimating local curvature at junctions and then PDF of curvatures is computed. Tortuosity features are useful to distinguish between fast and slow writers. Tortuosity is estimated by finding longest line segment that traverses every pixel [5]. Edge based features are computed using the histogram of gradients with three different scale space.

III. Training models

The dataset we are using has around 200 classes that represent 200 out of 206 writers. The remaining 6 writers are not present in the training dataset. There are around three documents per writer. We use two-third of the dataset for training and remaining for testing. Note that in test dataset there are writers which do not belong to the group of 200 writers, hence, care needs to be taken to reject the writers that do not belong to the group of writers that were not present during training.

In the current setup, we have selected five classifiers for training. More classifiers can be added to the system if needed. The goal here is to experiment with classifiers with different characteristics. We will discuss them below and later in Section VII we will analyze the performance of these classifiers in different scenarios:

A. Decision trees

Decision trees are non-parametric supervised learning methods. It tries to create a model that predicts the output label by learning simple decision rules inferred from the data features. If we select a deeper tree it learns more complex decision rules which may lead to over-fitting.

Some advantages of decision trees are that they are easy to interpret and trees can be visualized. That is if a given situation is easily observable in a model, the explanation for the condition is easily explained by boolean logic. This is in contrast to neural networks where it is black box model and results are more difficult to interpret. It requires less data preparation that is less preprocessing.

B. Extra trees

Extra trees (also called Extremely Randomized Trees) is ensemble technique that fits multiple randomized decision trees on data. Later to improve accuracy and control over-fitting these different models are averaged. It can be thought of using multiple decision trees with model averaging. In our setup, we used ten decision trees and then averaged it.

Extra trees technique is slightly different than random forest technique [9]. A random forest which is made up of an ensemble of decision trees decides the split of leaf node not based on the best split among all features, but instead, it is the best split among a random subset of the features. Extra trees go a step further in randomness. Instead of looking at most discriminative thresholds, the thresholds are drawn at random for each feature in a random subset, and then the best of these randomly generated thresholds is picked as the splitting rule.

C. SVM

A standard implementation based on libsvm was used. Radial basis function was used for SVM kernel. For multi-class classification, one-vs-one decision function was used.

D. Nearest Neighbors

KNN algorithm with K=1 was used for classification. Uniform weights were assigned to each neighbor. One of the reasons to use this classifier is that the authors of the dataset [2] provided a benchmark score on their dataset using KNN.

E. Linear Discriminant Analysis (LDA)

LDA is a classifier with linear decision boundary. The model fits Gaussian density to each class using Bayes rule. It assumes that all classes share the same covariance matrix. In our current setup, we are using SVD solver to fit the model.

IV. Improving feature representation

If the input is mathematically and computationally convenient to process then machine learning tasks such as classification perform well. But real world data is generally complex, redundant and highly variable. This necessitates discovering a good representation for the input. In this section, we look at methods that can achieve the same and see how finding good representation improves the accuracy.

In figure 6 you can see that the input to our machine learning system is concatenated features, which is a union of different types of features we extract as discussed in section II. We do not directly give raw images to classifier instead the input is the different kind of features which we have extracted heuristically assuming that they can help us in classification. But there is a problem with such representation. We do not know which features are important and discriminative. Also, there is redundancy in features. Hence, our goal here is to transform features such that they are more discriminative and try to learn dependency and redundancy in features. We have experimented with two approaches (displayed in green in figure 6) to improve feature representation and will be discussed in below subsections.

A. Feature selection / Dimensionality reduction

Dimensionality reduction is the process of reducing the number of random variables under consideration by obtaining a set of uncorrelated variables. PCA is a well-known example for dimensionality reduction where the input is projected in uncorrelated eigenvector space, but this transform is linear. PCA performs linear transform to a lower dimensional space in such a way that variance of data in low dimension representation is maximized.

In our experiment instead of using PCA, we trained Linear Support Vector Classifier on concatenated features using one vs rest scheme. Then we were able to weight features based on their importance i.e. more important features responsible for classification gets more weight. Then with this new ranking scheme we applied all five classifiers discussed in section III. In the context of this paper, we call this approach Linear SVC representation (i.e. dimensionality reduction achieved by linear SVC).

Some advantages of dimensionality reduction techniques are that they reduce time and storage requirements. They are good at removing collinearity and hence improve the performance of machine learning models. Also, reduction in dimensionality enables visualization and interpretation of data. Apart from the technique we used in our paper, there are many other techniques and a survey of the same can be found at [10].

B. Representation learning

Unsupervised learning tries to infer a function to describe hidden structure from unlabeled data. We can think of this as a problem of density estimation in statistics [11]. However, it also tries to summarize and explain key features of data. We already discussed advantages of improving feature representation, and in this section, we will look at how to learn better representation which is also termed as *Representation Learning*. Representation learning has become a field in itself in the machine learning community. Although domain knowledge can be used to help design representations, learning with generic priors can also be used [12]. The design of more powerful representation-learning algorithms implementing such priors is a challenging task. In our implementation, we will be looking at basic Restricted Boltzmann Machine configured which uses contrastive divergence algorithm [13].

1) RBMs for writer identification: RBMs have found applications in multiple areas of machine learning like dimensionality reduction [14], classification [15], collaborative filtering [16], feature learning [17] and topic modeling [18]. We are interested in feature learning aspect of RBM. In context of writer identification task resent attempt has been made to use Latent Dirichlet Allocation [19] which tries to learn the representation by topic modeling. But on the other hand Rustan et al. [18] have shown that RBMs outperform Latent Dirichlet Allocation in topic modeling task (but not in context of writer identification). So in our experiment setup we will try to investigate how RBM perform with writer identification task.

2) Training RBMs: RBMs are usually trained using contrastive divergence procedure [13]. The initial weights are initialized close to zero between 0.1 to zero. The number of hidden units is fixed to 4000. We selected stochastic binary hidden units for training with a momentum of 0.9 and batch size of 64 for the training. Most of these parameters were selected heuristically and finalized based on the visualization of hidden activities. The criteria for stopping the training was based on measuring mean squared error.

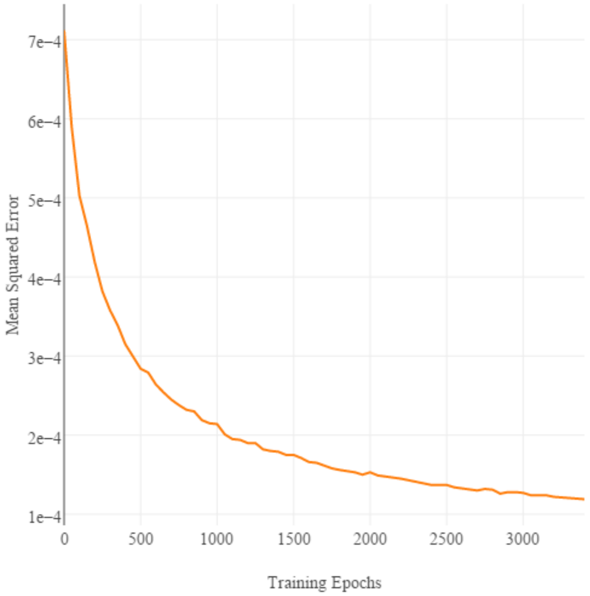


Figure 4: RBM training error

3) Monitoring the progress of learning: We print out the squared error between the data and the reconstructions for every training epoch as it is easy to calculate. Generally, the reconstruction error on the entire training set should fall rapidly and consistently at the start of learning and then more slowly. You can see the drop of mean squared error for the parameters we selected in figure 4. Although this approach is convenient to monitor learning it is not the right measure of the progress of learning. The error function we are monitoring is not what contrastive divergence algorithm is trying to optimize. So as of now we have used this error plot for making the decision when to stop the learning. We stop the training at the 3550th epoch as the error curve rapidly decays and flattens out. Advanced monitoring techniques are not explored in our work and leave an opportunity for further investigation.

4) Sparsity plots: One good way to understand what RBMs are learning is to visualize sparsity. Multiple ways are discussed in [20] to encourage sparsity while training. In our experimental setup with the parameters we selected as in section IV-B2 the RBMs were able to learn sparse activities as shown in figure 5a and 5b. As the training progress, we can see that some of the neurons activities grow stronger as training progress. While the activities of remaining neurons dampen out. At zeroth epoch, almost all neurons show average activity and as training progress, we can see how activities get sparser in the contour plot.

Although our hidden units are made up of binary stochastic neurons and the activities are either one or zero, in our plot the activities are real-valued probabilities of the firing of each neuron. This was achieved by showing the trained RBM the test images and building the histogram of hidden unit activities. Also note that in 5a and 5b we are only plotting small slice (i.e. 100 neurons) out of 4000 hidden units.

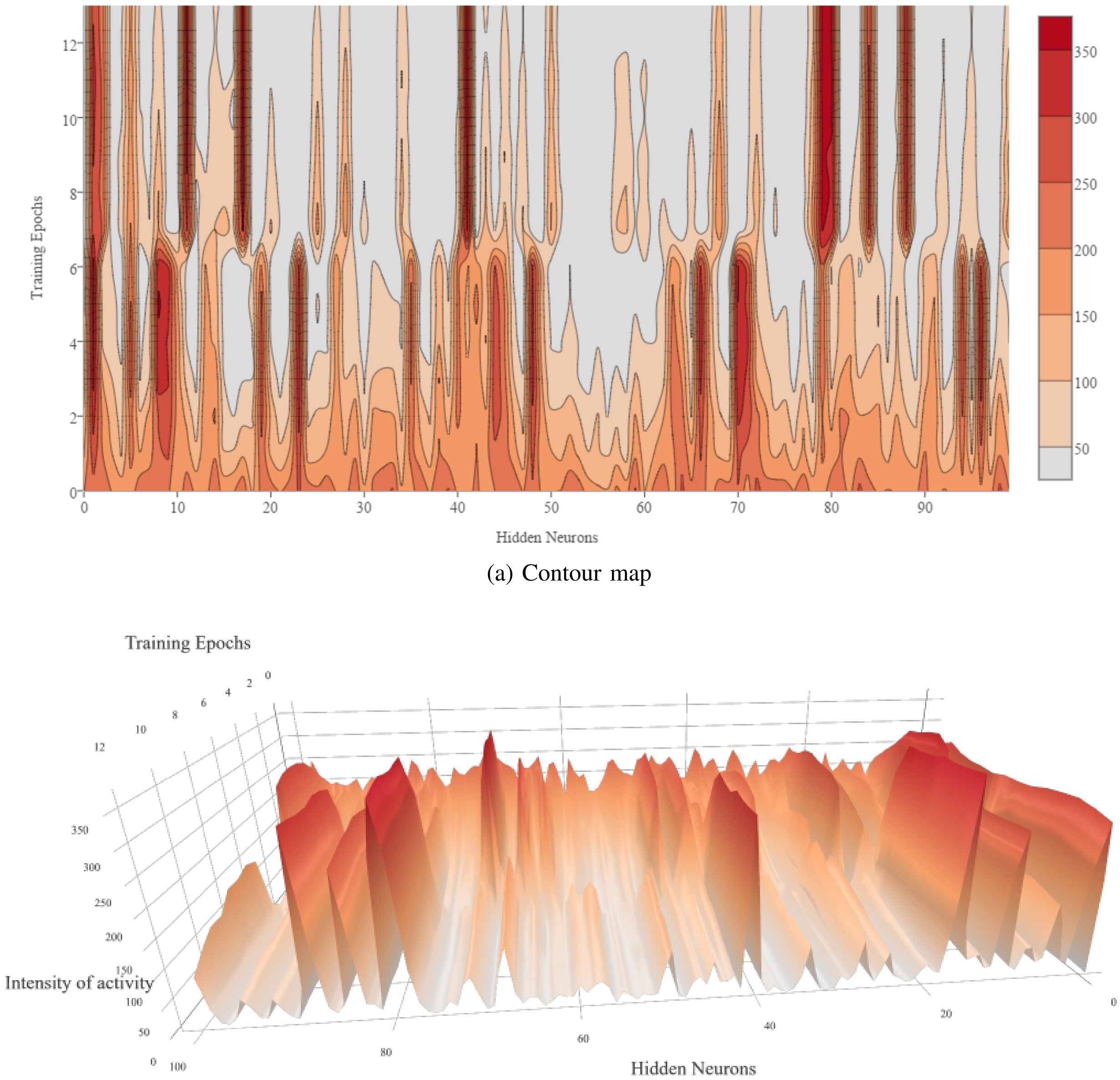
V. Rejection strategies

Writer identification task also involves rejecting those authors which do not belong to training dataset. A strategy needs to be implemented to reject such writers which are part of test dataset as has been studied in [21].

For writer rejection, we build one class classifier to represent all those 200 writers we trained on. During the test phase, we also find the probability that the writer does not belong to the model learned by this one class classifier. We do not claim this to be a better approach and more experiments need to be made. We achieved a small improvement with this approach. Our final results as shown in 7 do not reflect these improvements as more experiments need to be performed before justifying the improvement. The ’Rejection strategy’ block at the end of 6 in our parallel framework will serve as a container for implementing rejection strategies.

VI. Parallel framework

In this section, we will describe the parallel framework to perform author identification. In the case of handwritten manuscript analysis, there are different tasks like involving feature extraction, machine learning, image processing. Most of these tasks are independent. We have built a system



(a) Contour map

(b) 3D intensity plot

Figure 5: Change in hidden layer activities of RBM as training progresses

specific to writer identification. Please refer to figure 6 for a detailed architecture.

A. Tasks that can be paralleled

Below are some of the tasks that can be paralleled:

1) Feature extraction: For better analysis one needs to experiment with different type of features. As discussed in section II for our system we extract five different kinds of features. All these features are extracted in parallel from the same image source. The framework has a generic interface to plug more feature extraction techniques and run them in parallel with an option to persist extracted features.

2) Generating new representation: We highlighted the importance of changing representation of input features in section IV. In order to try

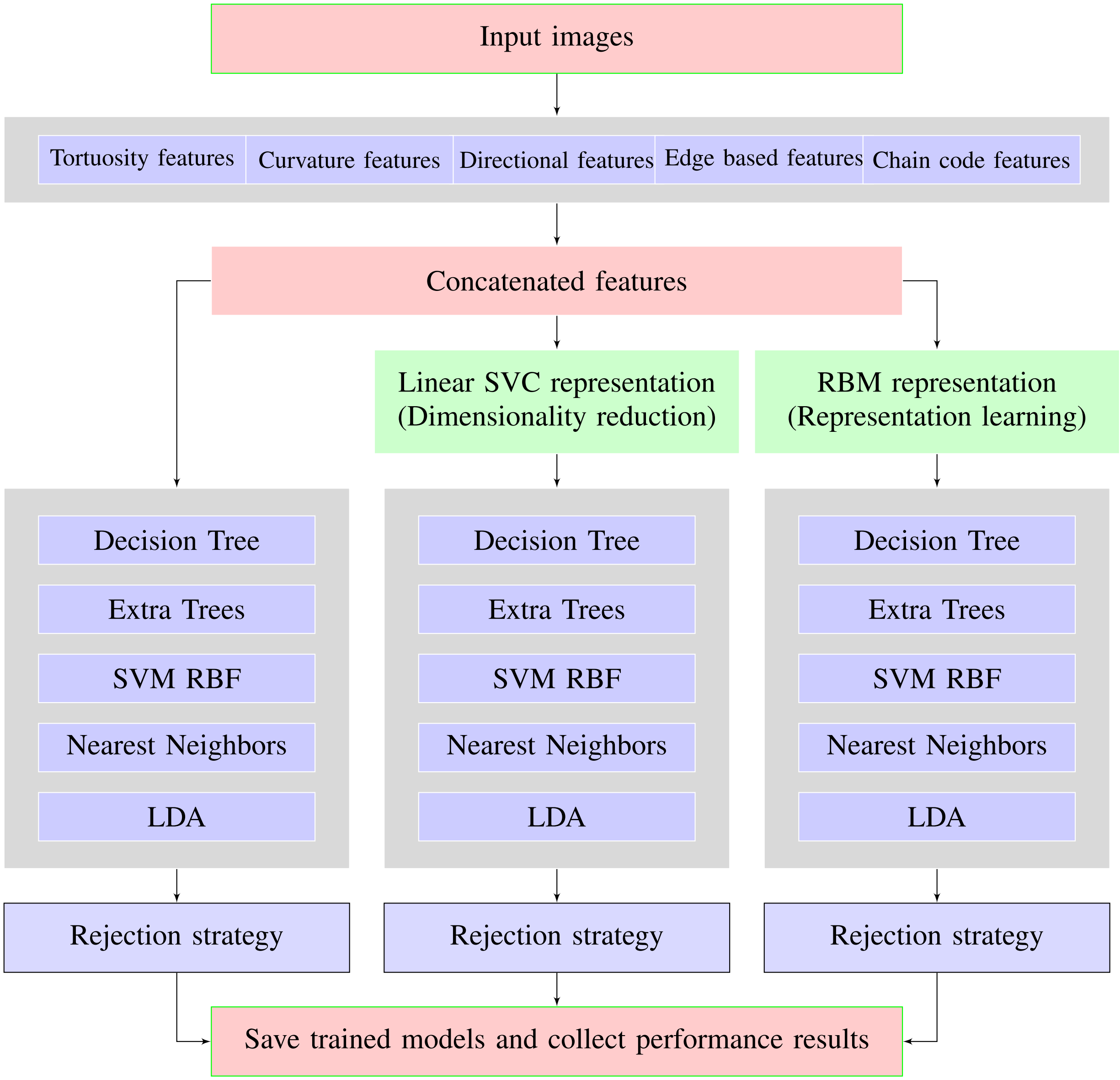


Figure 6: Parallel frame work for writer identification

many such techniques, you can just plug the representation changing algorithm and run them in parallel. Since this stage depends on features extracted from the earlier stage you need to wait for all the features extraction algorithms to finish in the earlier stage.

3) Learning models: In figure 6 we can see three gray boxes with five different machine learning models inside it. The first gray box gets the input directly without a change in the feature representation. While the other two gray boxes get features with changed representation. The system we designed can run all these three gray boxes in parallel. Also, it can run internal machine learning models in parallel threads. In the current setup, 3*5 threads will be spawned and some threads will get queued. No efforts are taken for load balancing, it is just a simple framework which can launch tasks in parallel.

Along with this, the machine learning model in rejection strategy can also be trained in parallel. But in the current setup, it is not implemented.

Finally, all the models trained in this stage can be persisted before we can run experiments on test data.

4) Performance benchmarking and reporting: Once we have trained models we can simply run these models on test dataset and save the result to a file. The result of this execution is shown in figure 7.

B. Usage scenarios for the parallel framework

Parallelization saves times for analysis. We can use this framework to figure out which feature extraction techniques worked better on data. Also, we can perform side by side comparison of the new representation generating strategies. Finally, we can also get a quick idea about which classifier models are better suited for our data. As no free lunch theorem states, no single classifier algorithm is better and comparative performance differs base on the input distribution of our data.

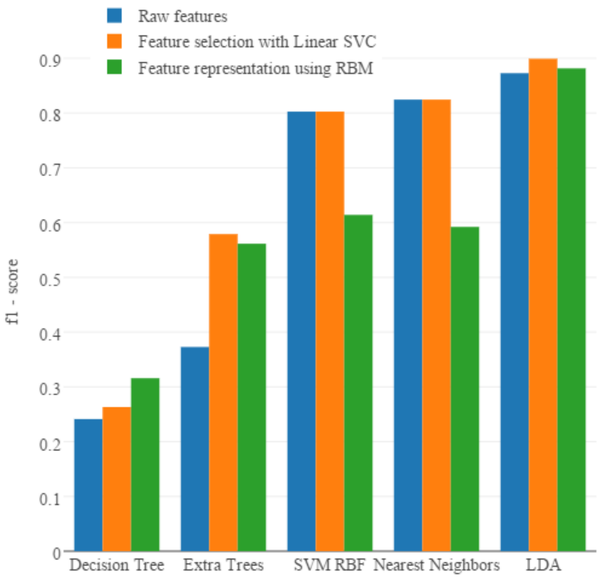


Figure 7: Performance results

VII. Results and analysis

Figure 7 show the combined results of the experiments we ran using the parallel framework. Best results were obtained by performing feature selection followed by Linear Discriminant Analysis. This suggests that the features extracted were not discriminative. Also changing feature representation improved accuracy as the new feature space has better feature representation. In the below section detailed analysis of the steps, we took to improve accuracy is discussed.

A. Feature Union

Finding right features for classification is always a problem. We need to experiment with different types of features. Also combining different features has more expressive power. For example, let us say that there are two writers who write with 10-degree tilt. So the directional features will easily figure out these writers provided that other writers do not have 10-degree tilt in their writing. But it will be still difficult to distinguish between these two writers with same 10degree tilt. So some additional features like curvature or tortuosity can help us identify theses writers uniquely. So one can say that combination of features is a powerful strategy. But too many features can ruin classification problem so learning the dependencies between features help overcome the problem. This is also evident from the results where accuracy was improved because of changing representation.

B. Analysis of classifiers used

In our framework as in figure 6 we tried to use five classifiers on the features we extracted. We also applied these classifiers on changed feature representation by performing dimensionality reduction and representation learning. In below subsections we will analyze the classifiers we used.

1) Decision trees: For the features, we extracted decision trees performed poorly. Changing representation improved results but not to an extent that they outperform LDA. Some of the reasons can be that decision trees create over complex trees and do not generalize on data well. The only way to improve it is selecting the depth of tree appropriately and deciding the minimum number of samples needed at the leaf node. In our current setup, we allow tree nodes to expand until all leaves are pure.

Another problem can be that the decision trees are sensitive to data and slightly different data can lead to a completely different tree. Given that we have very fewer data per writer the decision tree is perhaps not a good choice as it is unstable.

Apart from this decision trees can create biased trees and can get easily biased towards unbalanced dataset, but this cannot be a cause of deterioration in accuracy as we are using balanced dataset.

2) Extra trees: The reason to try this classifier is to overcome data sensitivity of single decision tree as explained in section VII-B1. We have only two samples per class for training and hence training multiple decision trees and then averaging may remove instability in the model learned by an individual decision tree. Also, averaging model is a popular strategy in machine learning competitions and we wanted to experiment how it performs with our dataset. As expected using multiple (ten in our setup) decision trees with model averaging improved the accuracy.

3) SVM: With SVM, we found a very good improvement in prediction results. One way to see this improvement in results is that SVM generalizes well compared to Random forests on small data and decision boundaries are more stable.

4) Nearest Neighbors: We experimented with different values of K and received the best performance with K=1. The KNN algorithm was better than SVM by a small fraction. One of the reasons to use this classifier was that the benchmark score was provided on the dataset using KNN algorithm [2].

5) Linear Discriminant Analysis: For performing model fitting using LDA solver we used SVD (Singular Value Decomposition) solver. In our case, the number of features is large and SVD solver is useful in this case as we do not need to calculate covariance matrix. That is it is scalable with as number of features increase.

With LDA, we got the best results in classification and this suggests that not all the features provided were discriminative. LDA with SVD can perform both transform and classification. By transform, we mean that the model has the capability to transform features and project them on most discriminative direction.

C. Analysis of impact of changing representation

Finding better representation for the data, finding the transform function where the data can be classified easily, or learning the relation between features has its own advantages. It reduces the complexity of learned models and improves the accuracy of classifiers. In our experiments when we trained classifier without changing the feature representation, our results were poor. But then we tried dimension reduction using Linear SVC as stated in section IV-A and we got best results. We also tried representation learning as described in section IV-B and obtained good results. These improvements suggest that changing representation found a better transform function that transforms our data in new feature space classifiers we used were able to model changed representation efficiently.

D. Feature Ranking

We trained linear SVC on the extracted features and checked the weights assigned to each category of features i.e. we ranked the features based on their contribution to classifier accuracy. We observed that chain code feature followed by curvature and directional features were the most discriminative features.

VIII. Conclusion

There is no one good solution when it comes to machine learning algorithms and analyzing the performance incrementally guides the process of finding a better solution and this also applies for writer identification techniques. We were able to experiment with different kind of features and concatenating these features improved accuracy. Changing the feature representation improved the results considerably as it helps untangle and reveal the different explanatory factors of variation behind the data. Linear Discriminant analysis classifier performed the best out of all classifiers as the input feature space was highly redundant. In the process of experimenting different algorithms, we were also able to build a plug and play parallel framework which can be reused for more experimentation. Further work needs to be done to improve writer rejection strategy.

For future work, we will investigate our RBM approach in comparison to Latent Dirichlet Allocation for writer identification task [19].

REFERENCES

[1] S. N. Srihari, S. H. Cha, H. Arora, and S. Lee, "Individuality of handwriting: A validation study," in *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, vol. 2001-Janua, 2001, pp. 106–109.

[2] A. Hassaine and S. A. Maadeed, "Icfhr 2012 competition on writer identification challenge 2: Arabic scripts," in *2012 International Conference on Frontiers in Handwriting Recognition*. IEEE, 9 2012, pp. 835–840.

[3] G. Ghiasi and R. Safabakhsh, "Offline text-independent writer identification using codebook and efficient code extraction methods," *Image and Vision Computing*, vol. 31, no. 5, pp. 379–391, 2013.

[4] C. Hertel and H. Bunke, "A set of novel features for writer identification," in *Audio- and Video-Based Biometric Person Authentication*, 2003, vol. 2688, pp. 679–687.

[5] A. B. Abdelâali Hassaïne, Somaya Al-Maadeed, "A set of geometrical features for writer identification," in *Volume 7667 of the series Lecture Notes in Computer Science*, ser. Lecture Notes in Computer Science, T. Huang, Z. Zeng, C. Li, and C. S. Leung, Eds. Springer Berlin Heidelberg, 2012, vol. 7667, ch. Neural Inf, pp. 584–591.

[6] G. Louloudis, N. Stamatopoulos, and B. Gatos, "Icdar 2011 writer identification contest," in *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, 2011, pp. 1475–1479.

[7] I. Siddiqi and N. Vincent, "Text independent writer recognition using redundant writing patterns with contour-based orientation and curvature features," *Pattern Recognition*, vol. 43, no. 11, pp. 3853–3865, 2010.

[8] U. V. Marti and H. Bunke, "The iam-database: An english sentence database for offline handwriting recognition," *International Journal on Document Analysis and Recognition*, vol. 5, no. 1, pp. 39–46, 2003.

[9] T. K. Ho, "Random decision forests," *Proceedings of 3rd International Conference on Document Analysis and Recognition*, vol. 1, pp. 278–282, 1995.

[10] I. K. Fodor, "A survey of dimension reduction techniques," *Library*, vol. 18, no. 1, pp. 1–18, 2002.

[11] C. M. Bishop, *Neural networks for pattern recognition*, 1995, vol. 92.

[12] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[13] M. a. Carreira-Perpinan and G. E. Hinton, "On contrastive divergence learning," *Artificial Intelligence and Statistics*, vol. 0, p. 17, 2005. [Online]. Available: http://learning.cs.toronto.edu/~hinton/absps/cdmiguel.pdf

[14] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[15] H. Larochelle and Y. Bengio, "Classification using discriminative restricted boltzmann machines," *Icml*, pp. 536–543, 2008. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1390156.1390224

[16] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," *Proceedings of the 24th international conference on Machine learning - ICML '07*, pp. 791–798, 2007. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1273496.1273596

[17] A. Coates, A. Arbor, and A. Y. Ng, "An analysis of single-layer networks in unsupervised feature learning," *Aistats 2011*, pp. 215–223, 2011.

[18] R. Salakhutdinov and G. Hinton, "Replicated softmax : an undirected topic model," *Advances in Neural Information Processing Systems 22*, vol. 22, pp. 1607–1614, 2009.

[19] A. Bhardwaj, M. Reddy, S. Setlur, V. Govindaraju, and S. Ramachandrula, *Latent Dirichlet allocation based writer identification in offline handwriting*, 2010. [Online]. Available: http://doi.acm.org/10.1145/1815330.1815376

[20] G. Hinton, "A practical guide to training restricted boltzmann machines a practical guide to training restricted boltzmann machines," *Computer*, vol. 9, no. 3, p. 1, 2010.

[21] D. Fecker, A. Asi, W. Pantke, V. Märgner, J. El-Sana, and T. Fingscheidt, “Document writer analysis with rejection for historical arabic manuscripts,” in *Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR*, vol. 2014-Decem, 2014, pp. 743–748.