A Novel Approach of Data Classification using <u>R</u>andom <u>M</u>ultimodel <u>D</u>eep <u>L</u>earning (RMDL)

Mojtaba Heidarysafa, Kamran Kowsari, Donald E. Brown, Kiana Jafari Meimandi,
Matthew S. Gerber, and Laura E. Barnes

***Abstract*—The continually increasing number of complex datasets each year necessitates ever improving machine learning methods for robust and accurate categorization of these data. This paper introduces Random Multimodel Deep Learning (RMDL): a new ensemble, deep learning approach for classification. Deep learning models have achieved state-of-the-art results across many domains. RMDL solves the problem of finding the best deep learning structure and architecture while simultaneously improving robustness and accuracy through ensembles of deep learning architectures. RDML can accept as input a variety data to include text, video, images, and symbolic. This paper describes RMDL and shows test results for image and text data including MNIST, CIFAR-10, WOS, Reuters, IMDB, and 20newsgroup. These test results show that RDML produces consistently better performance than standard methods over a broad range of data types and classification problems[1] .**

***Index Terms*—Text Mining; Document Classification; Deep Neural Networks; Hierarchical Learning; Deep Learning, Multimodel Deep Learning, RMDL**

I. Introduction

Categorization and classification with complex data such as images, documents, and video are central challenges in the data science community. Recently, there has been an increasing body of work using deep learning structures and architectures for such problems. However, the majority of these deep architectures are designed for a specific type of data or domain. There is a need to develop more general information processing methods for classification and categorization across a broad range of data types.

While many researchers have successfully used deep learning for classification problems (*e.g.,* see [1]–[5]), the central problem remains as to which deep learning architecture (DNN, CNN, or RNN) and structure (how many nodes (units) and hidden layers) is more efficient for different types of data and applications. The favored approach to this problem is trial and error for the specific application and dataset.

[1]Code is shared as an open source tool at https://github.com/kk7nc/RMDL

Manuscript received January 20, 2018; revised xx 00 2018,
This work was supported in part by the U.S.

Authors are with the Department of System and Information Engineering, University of Virginia Charlottesville, VA 22911 USA
Kamran Kowsari and Laura E. Barnes are with Sensing Systems for Health Lab, University of Virginia,Charlottesville, VA 22911 USA
Donald E. Brown, Matthew S. Gerber, and Laura E. Barnes are with Predictive Technology Laboratory, University of Virginia, Charlottesville, VA USA
Donald E. Brown and Matthew S. Gerber are with Data Science Institute, University of Virginia, Charlottesville, VA 22911 USA
e-mail: {kk7nc, mh4pk, deb, kj6vd, msg8u, lbarnes}@virginia.edu

This paper describes an approach to this challenge using ensembles of deep learning architectures as extended version of the previous authors' work [6]. This approach, called Random Multimodel Deep Learning (RMDL), uses three different deep learning architectures: Deep Neural Networks (DNN), Convolutional Neural Netwroks (CNN), and Recurrent Neural Networks (RNN). Test results with a variety of data types demonstrate that this new approach is highly accurate, robust and efficient.

The three basic deep learning architectures use different feature space methods as input layers. For instance, for feature extraction from text, DNN uses term frequency-inverse document frequency (TF-IDF) [7]. RDML searches across randomly generated hyperparameters for the number of hidden layers and nodes (desity) in each hidden layer in the DNN. CNN has been well designed for image classification. RMDL finds choices for hyperparameters in CNN using random feature maps and random numbers of hidden layers. CNN can be used for more than image data. The structures for CNN used by RMDL are 1D convolutional layer for text, 2D for images and 3D for video processings. RNN architectures are used primarily for text classification. RMDL uses two specific RNN structures: Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTM). The number of GRU or LSTM units and hidden layers used by the RDML are also the results of search over randomly generated hyperparameters.

The main contributions of this work are as follows:
I) Description of an ensemble approach to deep learning which makes the final model more robust and accurate.
II) Use of different optimization techniques in training the models to stabilize the classification task.
III) Different feature extraction approaches for each Random Deep Leaning (RDL) model in order to better understand the feature space (specially for text and video data).
IV) Use of dropout in each individual RDL to address over-fitting.
V) Use of majority voting among the n RDL models. This majority vote from the ensemble of RDL models improves the accuracy and robustness of results. Specifically, if k number of RDL models produce inaccuracies or overfit classifications and $n > k$, the overall system is robust and accurate VI) Finally, the RMDL has ability to process a variety of data types such as text, images and videos.

The rest of this paper is organized as follows: Section II gives related work for feature extraction, other classification techniques, and deep learning for classification task; Section III describes current techniques for classification tasks which are used as our baseline; Section IV shows feature extraction and pre-processing step in RMDL; Section V describes Random Multimodel Deep Learning methods and the architecture for RMDL including basic review of RMDL; Section V-A addresses the deep learning structure used in this model, Section V-B discusses optimization problem; Section VI-A talks about evaluation of these techniques; Section VI shows the experimental results which includes the accuracy and performance of RMDL; and finally, Section VII presents discussion and conclusions of our work.

II. Related Work

Researchers from a variety of disciplines have produced work relevant to the approach described in this paper. We have organized this work into three areas: I) Feature extraction; II) Classification methods and techniques (baseline and other related methods); and III) Deep learning for classification.

Feature Extraction: Feature extraction is a significant part of machine learning especially for text, image, and video data. Text and many biomedical datasets are mostly unstructured data from which we need to generate a meaningful and structures for use by machine learning algorithms. As an early example, L. Krueger *et. al.* in 1979 [8] introduced an effective method for feature extraction for text categorization. This feature extraction method is based on word counting to create a structure for statistical learning. Even earlier work by H. Luhn [9] introduced weighted values for each word and then G. Salton *et. al.* in 1988 [10] modified the weights of words by frequency counts called term frequency-inverse document frequency (TF-IDF). The TF-IDF vectors measure the number of times a word appears in the document weighted by the inverse frequency of the commonality of the word across documents. Although, the TF-IDF and word counting are simple and intuitive feature extraction methods, they do not capture relationships between words as sequences. Recently, T. Mikolov *et. al.* [11] introduced an improved technique for feature extraction from text using the concept of embedding, or placing the word into a vector space based on context. This approach to word embedding, called *Word2Vec*, solves the problem of representing contextual word relationships in a computable feature space. Building on these ideas, J. Pennington *et. al.* in 2014 [12] developed a learning vector space representation of the words called *Glove* and deployed it in Stanford NLP lab. The RMDL approach described in this paper uses Glove for feature extraction from textual data.

Classification Methods and Techniques: Over the last 50 years, many supervised learning classification techniques have been developed and implemented in software to accurately label data. For example, the researchers, K. Murphy in 2006 [13] and I. Rish in 2001 [14] introduced the Naïve Bayes Classifier (NBC) as a simple approach to the more general respresentation of the supervised learning classification problem. This approach has provided a useful technique for text classification and information retrieval applications. As with most supervised learning classification techniques, NBC takes an input vector of numeric or categorical data values and produce the probability for each possible output labels. This approach is fast and efficient for text classification, but NBC has important limitations. Namely, the order of the sequences in text is not reflected on the output probability because for text analysis, naïve bayes uses a bag of words approach for feature extraction. Because of its popularity, this paper uses NBC as one of the baseline methods for comparison with RMDL. Another popular classification technique is Support Vector Machines (SVM), which has proven quite accurate over a wide variety of data. This technique constructs a set of hyper-planes in a transformed feature space. This transformation is not performed explicitly but rather through the kernal trick which allows the SVM classifier to perform well with highly nonlinear relationships between the predictor and response variables in the data. A variety of approaches have been developed to further extend the basic methodology and obtain greater accuracy. C. Yu *et. al.* in 2009 [15] introduced latent variables into the discriminative model as a new structure for SVM, and S. Tong *et. al.* in 2001 [16] added active learning using SVM for text classification. For a large volume of data and datasets with a huge number of features (such as text), SVM implementations are computationally complex. Another technique that helps mediate the computational complexity of the SVM for classification tasks is stochastic gradient descent classifier (SGDClassifier) [17] which has been widely used in both text and image classification. SGDClassifier is an iterative model for large datasets. The model is trained based on the SGD optimizer iteratively.

Deep Learning: Neural networks derive their architecture as a relatively simply representation of the neurons in the human's brain. They are essentially weighte combinations of inputs the pass through multiple non-linear functions. Neural networks use an iterative learning method known as back-propagation and an optimizer (such as stochastic gradient descent (SGD)).

Deep Neural Networks (DNN) are based on simple neural networks architectures but they contain multiple hidden layers. These networks have been widely used for classification. For example, D. CireşAn *et. al.* in 2012 [18] used multi-column deep neural networks for classification tasks, where multi-column deep neural networks use DNN architectures. Convolutional Neural Networks (CNN) provide a different architectural approach to learning with neural networks. The main idea of CNN is to use feed-forward networks with convolutional layers that include local and global pooling layers. A. Krizhevsky in 2012 [19] used CNN, but they have used $2D$ convolutional layers combined with the $2D$ feature space of the image. Another example of CNN in [2] showed excellent accuracy for image classification. This architecture can also be used for text classification as shown in the work of [20]. For text and sequences, $1D$ convolutional layers are used with word embeddings as the input feature space. The final type of deep learning architecture is Recurrent Neural Networks (RNN) where outputs from the neurons are fed back into the network as inputs for the next step. Some recent extensions to this architecture uses Gated Recurrent Units (GRUs) [4] or Long Short-Term Memory (LSTM) units [21]. These new units help control for instability problems in the original network architecure. RNN have been successfully used for natural language processing [22]. Recently, Z. Yang *et. al.* in 2016 [23] developed hierarchical attention networks for document classification. These networks have two important characteristics: hierarchical structure and an attention mechanism at word and sentence level.

New work has combined these three basic models of the deep learning structure and developed a novel technique for enhancing accuracy and robustness. The work of M. Turan *et. al.* in 2017 [5] and M. Liang *et. al.*in 2015 [24] implemented innovative combinations of CNN and RNN called *A Recurrent Convolutional Neural Network (RCNN)*. K. Kowsari *et. al.* in 2017 [1] introduced hierarchical deep learning for text classification (HDLTex) which is a combination of all deep learning techniques in a hierarchical structure for document classification has improved accuracy over traditional methods. The work in this paper builds on these ideas, spcifically the work of [1] to provide a more general approach to supervised learning for classification.

III. Baseline

In this paper, we use both contemporary and traditional techniques of document and image classification as our baselines. The baselines of image and text classification are different due to feature extraction and structure of model; thus, text and image classification's baselines are described separately as follows:

A. Text Classification Baselines

Text classification techniques which are used as our baselines to evaluate our model are as follows: regular deep models such as Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), and Deep Neural Networks (DNN). Also, we have used two different techniques of Support Vector Machine (SVM), naïve bayes classification (NBC), and finally Hierarchical Deep Learning for Text Classification (HDLTex) [1].

1) Deep Learning: The baseline, we used in this paper is Deep Learning without Hierarchical level. One of our baselines for text classification is [23]. In our methods' Section V, we will explain the basic models of deep learning such as DNN, CNN, and RNN which are used as part of RMDL model.

2) Support Vector Machine (SVM): The original version of SVM was introduced by Vapnik, VN and Chervonenkis, A Ya [25] in 1963. The early 1990s, nonlinear version was addressed in [26].

Multi-class SVM: The original version of SVM is used for binary classification, so for multi class we need to generate Multimodel or MSVM. One-Vs-One is a technique for multi-class SVM and needs to build N(N-1) classifiers.

$$f(x) = arg\max_i \Big(\sum_j f_{ij}(x)\Big) \tag{1}$$

The natural way to solve k-class problem is to construct a decision function of all k classes at once [27], [28]. In general, multi-class svm is an optimization problem of:

$$\min_{w_1,w_2,..,w_k,\zeta} \frac{1}{2}\sum_k w_k^T w_k + C\sum_{(x_i,y_i)\in D}\zeta_i \tag{2}$$

such that:

$$\begin{aligned} &w_{y_i}^T x - w_k^T x \leq i - \zeta_i, \\ &\forall (x_i,y_i) \in D, k \in \{1,2,...,K\}, k \neq y_i \end{aligned} \tag{3}$$

Where (x_i, y_i) is training data point such that $(x_i, y_i) \in D$. C is the penalty parameter, ζ is slack parameter, k stands for classes. Another technique of multi-class classification using SVM is All-against-One. In SVM we have two different methods for feature extraction which are word sequences feature extracting [29], and Term frequency-inverse document frequency (TF-IDF).

String Kernel: The basic idea of String Kernel (SK) is using $\Phi(.)$ for mapping string in the feature space; Therefore, the only different between the three techniques we use in our paper is the way to map the string into feature space. For many applications such as text, DNA, and protein classification, Spectrum Kernel (SP) is addressed [30], [31]. The basic idea of SP is counting number of time a word appears in string x_i as feature map where defining feature maps from $x \rightarrow \mathbb{R}^{l^k}$

$$\Phi_k(x) = \Phi_j(x)_{j\in\Sigma^k} \tag{4}$$

where:

$$\Phi_j(x) = number\ of\ j\quad feature\ appears\ in\ x \tag{5}$$

The feature map $\Phi_i(x)$ is generated by the sequence x_i and kernel defines as follows:

$$F = \Sigma^k \tag{6}$$

$$K_i(x,x') =< \Phi_i(x), \Phi_i(x') > \tag{7}$$

Mismatch Kernel is the other stable way to map the string into feature space. The key idea is using k which stands for $k-mer$ or size of the word and allow to have m mismatch in feature space [32]. The main problem of SVM for string sequences is time complexity of these models. S. Ritambhara *et. al.* in 2017 [33] addressed the problem of time for gap k-mers kernel called *GaKCo* which is used only for protein and DNA sequences. The features are generated using dictionary size Σ and F is number of feature and bounded by equation 6. The kernel calculation is similar with Spectrum

Kernel and use equation 7, and finally, we normalize Kernel using equation 8.

$$K^{Norm}(x,y) \leftarrow \frac{K(x,y)}{\sqrt{K(x,x)}\sqrt{K(y,y)}} \tag{8}$$

$$< f^x, f^y >= \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} h_{gm}(u_i^{s_1}, u_j^{s_2}) \tag{9}$$

Where two sequences ,$u_i^{s_1}$ and $u_j^{s_2}$, are lengths of s_1 and $s2$ respectively, and h_{gm} k-mers as features space.

Stacking Support Vector Machine (SVM): We use Stacking SVMs as another baseline method for comparison with RMDL for datasets which has capability to use hierarchical labels. The stacking SVM provides an ensemble of individual SVM classifiers and generally produces more accurate results than single-SVM models [34], [35].

3) Naïve Bayes Classification (NBC): his technique has been used in industry and academia for a long time, and it is the most traditional method of text categorization which is widely used in Information Retrieval [36]. If the number of n documents, fit into k categories where $k \in \{c_1, c_2, ..., c_k\}$ the predicted class as output is $c \in C$. Naïve bayes is a simple algorithm as follows:

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)} \tag{10}$$

where d is document, c indicates classes.

$$\begin{aligned} C_{MAP} =& \ arg\max_{c\in C} P(d \mid c)P(c) \\ =& \ arg\max_{c\in C} P(x_1, x_2, ..., x_n \mid c)p(c) \end{aligned} \tag{11}$$

The baseline of this paper is word level of NBC [37] as follows:

$$P(c_j \mid d_i; \hat{\theta}) = \frac{P(c_j \mid \hat{\theta})P(d_i \mid c_j; \hat{\theta}_j)}{P(d_i \mid \hat{\theta})} \tag{12}$$

4) Hierarchical Deep Learning for Text Classification (HDLTex): Hierarchical Deep Learning for Text Classification (HDLTex) is used as one of our baselines for hierarchical datasets. When documents are organized hierarchically, multi-class approaches are difficult to apply using traditional supervised learning methods. The HDLTex [1] introduced a new approach to hierarchical document classification that combines multiple deep learning approaches to produce hierarchical classification. The primary contribution of HDLTex research is hierarchical classification of documents. A traditional multi-class classification technique can work well for a limited number of classes, but performance drops with increasing number of classes, as is present in hierarchically organized documents. HDLTex solved this problem by creating architectures that specialize deep learning approaches for their level of the document hierarchy.

B. Image Classification Baselines

For image classification, we have five baselines as follows: Deep L2-SVM [38], Maxout Network [39], BinaryConnect [40], PCANet-1 [41], and gcForest [42].

Deep L2-SVM: This technique is known as deep learning using linear support vector machines which simply softmax is replaced with linear SVMs [38].

Maxout Network: I. Goodfellow*et. al.* in 2013 [39] defined a simple novel model called *maxout* (named because its outputs' layer is a set of max of inputs' layer, and it is a natural companion to dropout). Their design both facilitates optimization by using dropout, and also improves the accuracy of dropout's model.

BinaryConnect: M. Courbariaux*et. al.* in 2015 [40] worked on training Deep Neural Networks (DNN) with binary weights during propagations. They have introduced a binarization scheme for binary weights during forward and backward propagations (BinaryConnect) which mainly used for image classification. BinaryConnect is used as our baseline for RMDL on image classification.

PCANet: I. Chan *et. al.* in 2015 [41] is simple way of deep learning for image classification which uses CNN structure. Their technique is one of the basic and efficient methods of deep learning. The CNN structure they used, is part of RMDL with significant differences that number of hidden layers and nodes in RMDL is selected automatically.

gcForest (Deep Forest): Z. Zhou *et. al.* in 2017 [42] introduced a decision tree ensemble approach with high performance as an alternative to deep neural networks. Deep forest creates multi level of forests as decision tree.

IV. Feature Extraction and Data Pre-processing

The feature extraction is divided into two main parts for RMDL (Text and image). Text and sequential datasets are unstructured data, while the feature space is structured for image datasets.

1) Image and 3D Object Feature Extraction: Image features are the followings: $h \times\ w \times\ c$ where h denotes the height of the image, w represents the width of image, and c is the color that has 3 dimensions (RGB). For gray scale datasets such as $MNIST$ dataset, the feature space is $h \times\ w$. A 3D object in space contains n cloud points in space and each cloud point has 6 features which are (*x, y, z, R, G, and B*). The 3D object is unstructured due to number of cloud points since one object could be different with others. However, we could use simple instance down/up sampling to generate the structured datasets.

2) Text and Sequences Feature Extraction: In this paper we use several techniques of text feature extraction which are word embedding (Glove and Word2vec) and also TF-IDF. In this paper, we use word vectorization techniques [43] for extracting features; besides, we also use N-gram algorithm for extracting features for neural deep learning [44], [45] the string "In this Paper" would be composed of the following N-grams [46]

- bi-grams: _I, In, n_, _ t,..

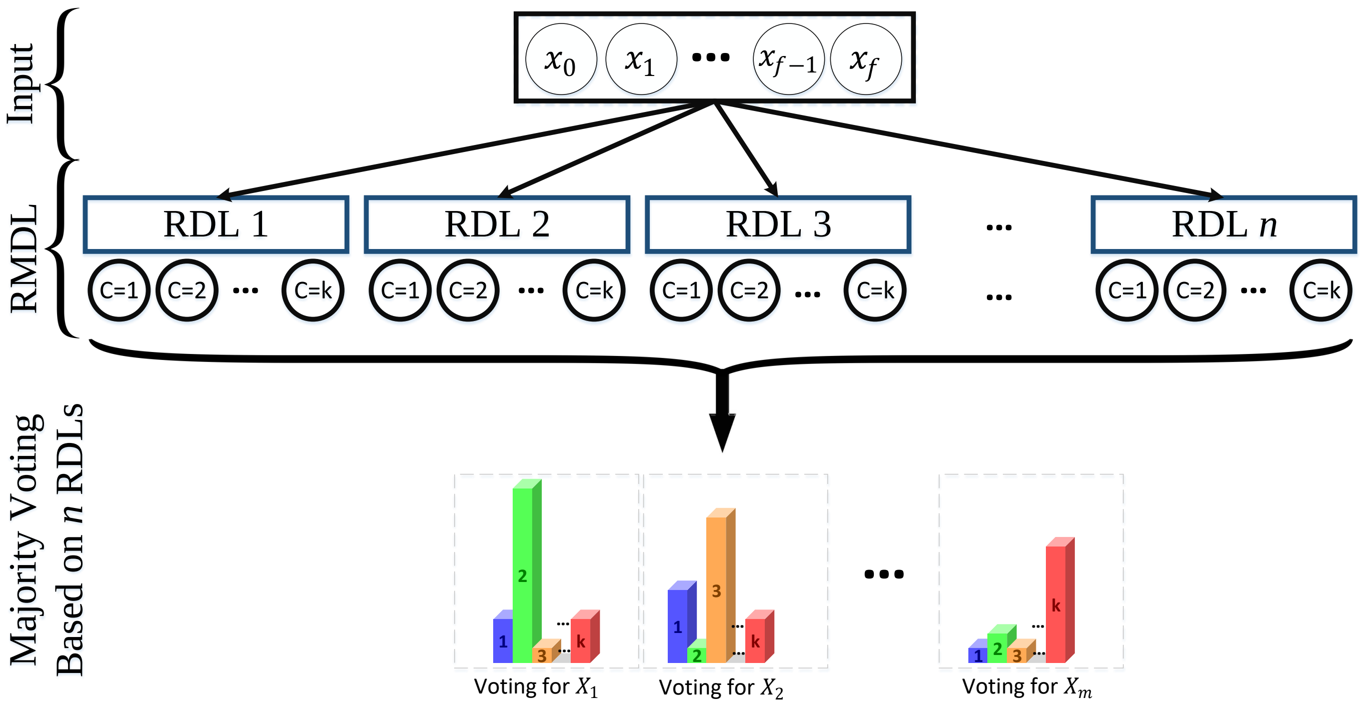


Fig. 1: Overview of RDML: Random Multimodel Deep Learning for classification. The RMDL includes n Random models which are d random model of DNN classifiers, c models of CNN classifiers, and r RNN classifiers where $r + c + d = n$.

- tri-grams: _In, In_, n_ t, _ th,...
- quad-grams: _ In_ , In_ t,...

in word level of our techniques, fixed length is not used in our technique. Feature extraction in our model for the string ”In this Paper we introduced this technique” would be composed of the following:

- Feature count(1) { (In 1) , (this 2), (Paper 1), (we 1), (introduced 1), (technique 1) }
- Feature count(2) { (In 1) , (this 2), (Paper 1), (we 1), (introduced 1), (technique 1), (In this 1), (This Paper 1), (Paper we 1), (we introduced 1), (introduced this), (this technique 1) } A vector-space model is a mathematical mapping of the word space, defined as

Documents enter our models via features extracted from the text. We employed different feature extraction approaches for the deep learning architectures we built. For CNN and RNN, we used the text vector-space models using 200 dimensions as described in Glove [12]. A vector-space model is a mathematical mapping of the word space, defined as

$$d_j = (w_{1,j}, w_{2,j}, ..., w_{i,j} ..., w_{l_j,j}) \tag{13}$$

where l_j is the length of the document j, and $w_{i,j}$ is the Glove word embedding vectorization of word i in document j.

Term Frequency-Inverse Document Frequency:
K. Sparck Jones [47] proposed inverse document frequency (IDF) that can be used in conjunction with term frequency to lessen the effect of such common words in the corpus. therefore, a higher weight will e assigned to the words with both high frequency of a term and low frequency of the term in the document. The mathematical representation of weight of a term in a document by Tf-idf is given in 14 .

$$W(d,t) = TF(d,t) * log(\frac{N}{df(t)}) \tag{14}$$

Where N is number of documents and df(t) is the number of documents containing the term t in the corpus. The first part in 14 would improve recall and the later would improve the precision of the word embedding [**?**]. Although tf-idf tries to overcome the problem of common terms in document, it still suffers from some other descriptive limitations. Namely, tf-idf can not account for the similarity between words in the document since each word is presented as an index.In the recent years, with development of more complex models such as neural nets, new methods has been presented that can incorporate concepts such as similarity of words and part of speech tagging. This work uses ,word2vec and Glove, two of the most common methods that have been successfully used for deep learning techniques.

Word2Vec:
T. Mikolov et al. [11] presented *”word to vector”* representation as a better word embedding architecture. Word2vec approach uses two neural network namely continuous bag of words (CBOW) and continuous skip-gram to create a high dimension vector for each words. The CBOW tries to find the word based on previous words

while skip-gram tries to find words that might come in the vicinity of each word. The method provides very powerful relationship discovery as well as similarity between the words.For instance, this embedding would consider the two words such as ”big” and ”bigger” close to each other in the vector space it assign them. In this work, we used word2vec based on its powerful representation it provides. The model has been trained on all reports and then fed into the deep neural networks described in XX. Figure () shows the result of our embedding using t-sne for visualization.

Global Vectors for Word Representation (GloVe): Another powerful word embedding technique that has been used in this work is Global Vectors (Glove) presented in [12]. The approach is very similar to the word to vector method where each words is presented by a high dimension vector and trained based on the surrounding words over huge corpus. The pre-trained embedding for words used in this work are based on 400,000 vocabulary trained over Wikipedia 2014 and Gigaword 5 as the corpus and 50 dimensions for word presentation. Glove also provides other pre-trained word vectorizations with 100,200, 300 dimensions which are trained over even bigger corpus as well as over twitter. figure ()shows a visualization of the word distances over accident reports using the same t-sne technique.

V. Random Multimodel Deep Learning

The novelty of this work is in using multi random deep learning models including DNN, RNN, and CNN techniques for text and image classification. The method section of this paper is organized as follows: first we describe RMDL and we discuss three techniques of deep learning architectures (DNN, RNN, and CNN) which are trained in parallel. Next, we talk about multi optimizer techniques that are used in different random models.

Random Multimodel Deep Learning is a novel technique that we can use in any kind of dataset for classification. An overview of this technique is shown in Figure 2 which contains multi Deep Neural Networks (DNN), Deep Convolutional Neural Networks (CNN), and Deep Recurrent Neural Networks (RNN). The number of layers and nodes for all of these Deep learning multi models are generated randomly (*e.g.* 9 Random Models in RMDL constructed of 3 CNNs, 3 RNNs, and 3 DNNs, all of them are unique due to randomly creation).

$$M(y_{i1}, y_{i2}, ..., y_{in}) = \left\lfloor \frac{1}{2} + \frac{(\sum_{j=1}^{n} y_{ij}) - \frac{1}{2}}{n} \right\rfloor \tag{15}$$

Where n is the number of random models, and y_{ij} is the output prediction of model for data point i in model j (Equation 15 is used for binary classification, $k \in \{0 \text{ or } 1\}$). Output space uses majority vote for final $\hat{y}_i$. Therefore, $\hat{y}_i$ is given as follows:

$$\hat{y}_i = \begin{bmatrix} \hat{y}_{i1} & \dots & \hat{y}_{ij} & \dots & \hat{y}_{in} \end{bmatrix}^T \tag{16}$$

Where n is number of random model, and $\hat{y}_{ij}$ shows the prediction of label of document or data point of $D_i \in \{x_i, y_i\}$ for model j and $\hat{y}_{i,j}$ is defined as follows:

$$\hat{y}_{i,j} = arg \max_k [softmax(y^*_{i,j})] \tag{17}$$

After all RDL models (RMDL) are trained, the final prediction is calculated using majority vote of these models.

A. Deep Learning in RMDL

The RMDL model structure (section V) includes three basic techniques of deep learning in parallel. We describe each individual model separately. The final model contains d random DNNs (Section V-A1), r RNNs (Section V-A2), and c CNNs models (Section V-A3).

1) Deep Neural Networks:

Deep Neural Networks’ structure is designed to learn by multi connection of layers that each layer only receives connection from previous and provides connections only to the next layer in hidden part. The input is a connection of feature space with first hidden layer for all random models. The output layer is number of classes for multi-class classification and only one output for binary classification. But our main contribution of this paper is that we have many training DNN for different purposes. In our techniques, we have multi-classes DNNs which each learning models is generated randomly (number of nodes in each layer and also number of layers are completely random assigned). Our implementation of Deep Neural Networks (DNN) is discriminative trained model that uses standard back-propagation algorithm using sigmoid (equation 18), ReLU [48] (equation 19) as activation function. The output layer for multi-class classification, should use $Softmax$ equation 20.

$$f(x) = \frac{1}{1 + e^{-x}} \in (0, 1) \tag{18}$$

$$f(x) = \max(0, x) \tag{19}$$

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{20}$$
$$\forall\ j \in \{1, \dots, K\}$$

Given a set of example pairs $(x, y), x \in X, y \in Y(x, y), x \in X, y \in Y$ The goal is to learn from these input and target space using hidden layers. In text classification, the input is string which is generated by vectorization of text. In Figure 2 the left model shows how DNN contribute in RMDL.

2) Recurrent Neural Networks (RNN):

Another neural network architecture that contributes in RMDL is Recurrent Neural Networks (RNN). RNN assigns more weights to the previous data points of sequence. Therefore, this technique is a powerful method for text, string and

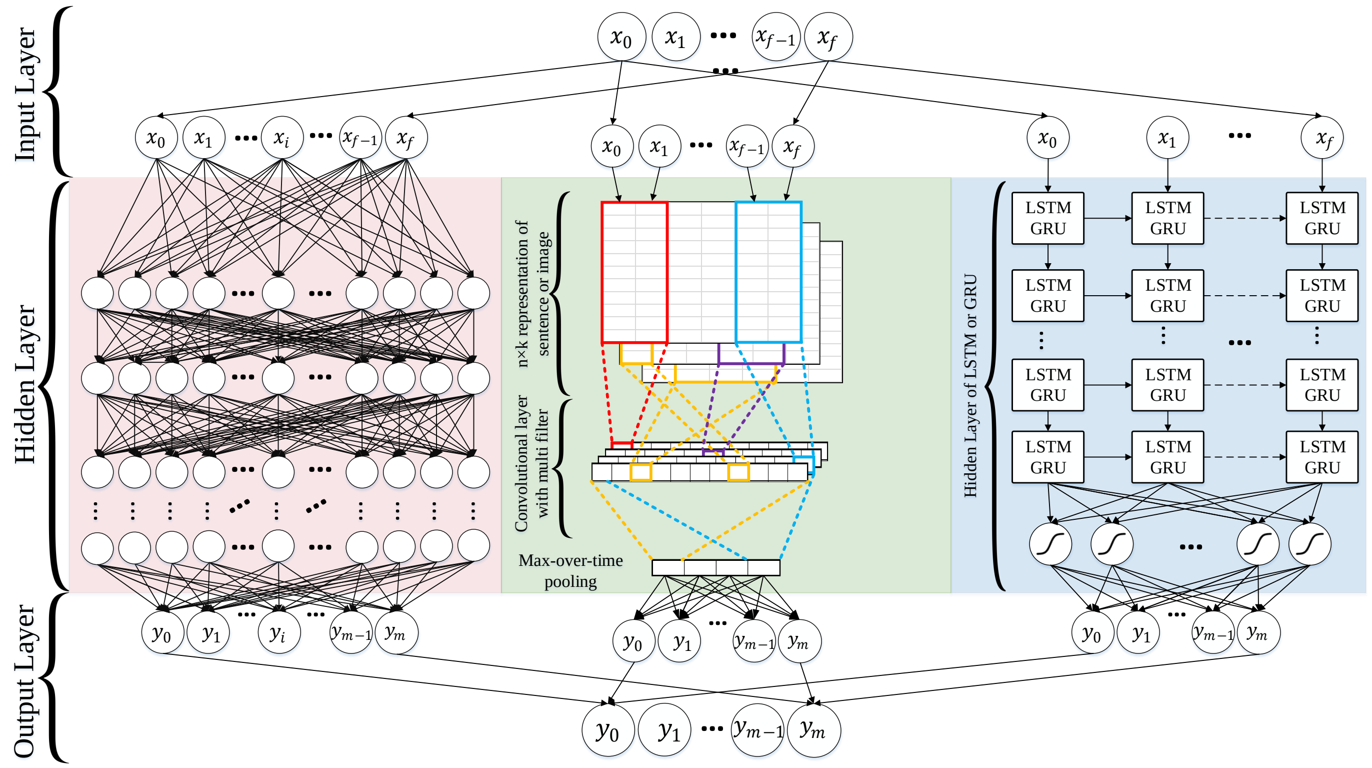

Input Layer
Hidden Layer
Output Layer
n×k representation of sentence or image
Convolutional layer with multi filter
Max-over-time pooling
Hidden Layer of LSTM or GRU
LSTM GRU

Fig. 2: Random Multimodel Deep Learning (RDML) architecture for classification. RMDL includes 3 Random models, one DNN classifier at left, one Deep CNN classifier at middle, and one Deep RNN classifier at right (each unit could be LSTM or GRU).

sequential data classification, and also we used this technique for image classification. In RNN the neural net considers the information of previous nodes in a very sophisticated method which allows for better semantic analysis of structures of dataset. General formulation of this concept is given in Equation 21 where x_t is the state at time t and $\boldsymbol{u_t}$ refers to the input at step t.

$$x_t = F(x_{t-1}, \boldsymbol{u_t}, \theta) \tag{21}$$

More specifically, we can use weights to formulate the Equation 21 with specified parameters in Equation 22

$$x_t = \mathbf{W_{rec}}\sigma(x_{t-1}) + \mathbf{W_{in}u_t} + \mathbf{b} \tag{22}$$

Where $\mathbf{W_{rec}}$ refers to recurrent matrix weight, $\mathbf{W_{in}}$ refers to input weights, $\mathbf{b}$ is the bias and σ denotes an element-wise function.

Again we have modified the basic architecture for use RMDL. Figure 2 left side shows this extended RNN architecture. Several problems arise from RNN when the error of the gradient descent algorithm is back propagated through the network: vanishing gradient and exploding gradient [49].

Long Short-Term Memory (LSTM):
To deal with these problems Long Short-Term Memory (LSTM) is a special type of RNN that preserve long term dependency in a more effective way in comparison to the basic RNN. This is particularly useful to overcome vanishing gradient problem [50]. Although LSTM has a chain-like structure similar to RNN, LSTM uses multiple gates to carefully regulate the amount of information that will be allowed into each node state. Figure 3 shows the basic cell of a LSTM model. A step by step explanation of a LSTM cell is as following:

$$i_t = \sigma(W_i[x_t, h_{t-1}] + b_i), \tag{23}$$

$$\tilde{C}_t = \tanh(W_c[x_t, h_{t-1}] + b_c), \tag{24}$$

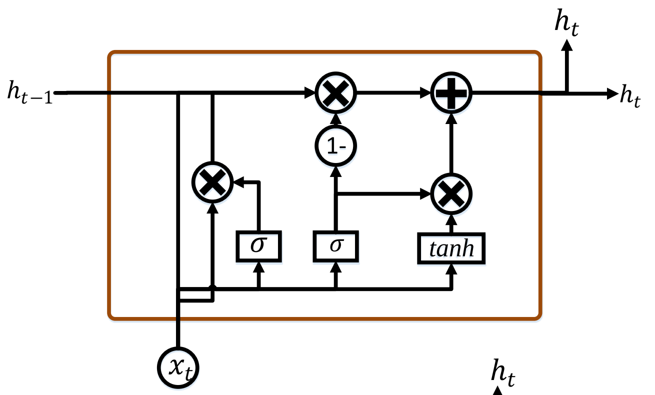
$$f_t = \sigma(W_f[x_t, h_{t-1}] + b_f), \tag{25}$$

$$C_t = i_t * \tilde{C}_t + f_t C_{t-1}, \tag{26}$$

$$o_t = \sigma(W_o[x_t, h_{t-1}] + b_o), \tag{27}$$

$$h_t = o_t \tanh(C_t), \tag{28}$$

Where equation 23 is input gate, Equation 24 shows candid memory cell value, Equation 25 is forget gate activation, Equation 26 is new memory cell value, and Equation 27



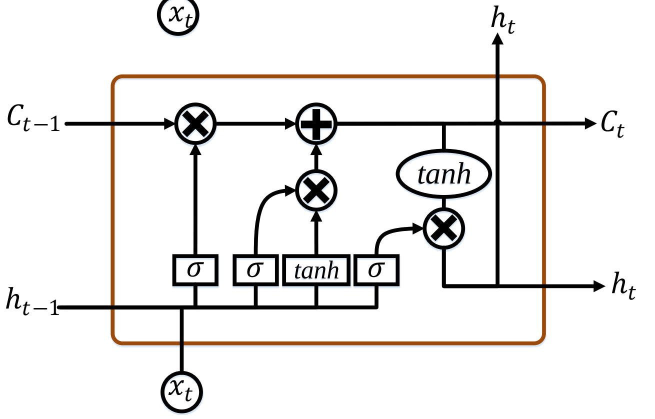


Fig. 3: Left Figure is a cell of GRU, and right Figure is a cell of LSTM

and 28 show output gate value. In the above description all b represents bias vectors and all W represent weight matrices and x_t is used as input to the memory cell at time t. Also, i, c, f, o indices refer to input, cell memory, forget and output gates respectively. Figure 3 shows the structure of these gates with a graphical representation.
An RNN can be biased when later words are more influential than the earlier ones. To overcome this bias Convolutional Neural Network (CNN) models (discussed in Subsection V-A3 were introduced which deploys a max-pooling layer to determine descriminative phrases in a text [51].

Gated Recurrent Unit (GRU):
Gated Recurrent Unit (GRU) is a gating mechanism for RNN which was introduced by [4] and [52]. GRU is a simplified variant of the LSTM architecture, but there are differences as follows: GRU contains two gates, a GRU does not possess internal memory (the C_{t-1} in Figure 3; and finally, a second non-linearity is not applied (tanh in Figure 3). Gated Recurrent Unit (GRU) is a simple gating mechanism for RNN that was introduced by [52]. GRU is a simplified variant of the LSTM architecture, but there are differences as follows: GRU contains two gates, a GRU does not possess internal memory (the C_{t-1} in Figure 3; and finally, a second non-linearity is not applied (tanh in Figure 3). A step by step explanation of a GRU cell is as following:

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1}] + b_z), \tag{29}$$

Where z_t refers to update gate vector of t, x_t stands for input vector, W, U and b are parameter matrices and vector, σ_g is activation function that could be sigmoid or ReLU.

$$\tilde{r_t} = \sigma_g(W_r x_t + U_r h_{t-1}] + b_r), \tag{30}$$

Where r_t stands for reset gate vector of t, x_t is input vector, W, U and b are parameter matrices and vector, σ_g is activation function that could be sigmoid or ReLU.

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h) \tag{31}$$

Where h_t is output vector of t, r_t stands for reset gate vector of t, z_t is update gate vector of t, σ_h indicates the hyperbolic tangent function.

3) *Convolutional Neural Networks (CNN):*

The final deep learning approach which contribute in RMDL is Convolutional Neural Networks (CNN) that is employed for hierarchical document or image classification. Although originally built for image processing with architecture similar to the visual cortex, CNN have also been effectively used for text classification [53]; thus, in RMDL, this technique is used in all datasets.
In the basic CNN for image processing an image tensor is convolved with a set of kernels of size $d \times d$. These convolution layers are called feature maps and these can be stacked to provide multiple filters on the input. To reduce the computational complexity CNN use pooling which reduces the size of the output from one layer to the next in the network. Different pooling techniques are used to reduce outputs while preserving important features [54]. The most common pooling method is max pooling where the maximum element is selected in the pooling window.
In order to feed the pooled output from stacked featured maps to the next layer, the maps are flattened into one column. The final layers in a CNN are typically fully connected.
In general during the back propagation step of a convolutional neural network not only the weights are adjusted but also the feature detector filters . A potential problem of CNN used for text is the number of 'channels', Σ (size of the feature space). This might be very large (*e.g.* 50K), for text but for images this is less of a problem (*e.g.* only 3 channels of RGB) [55]. This means the dimensionality of the CNN for text is very high.

B. Optimization

In this paper we use two type of stochastic gradient optimizer in our neural networks implementation which are RMSProp and Adam optimizer:

TABLE I: Accuracy comparison for text classification. W.1 (WOS-5736) refers to Web of Science dataset, W.2 represents W-11967, W.3 is WOS-46985, and R stands for Reuters-21578

	Model	Dataset			
		WOS-5736	W-11967	WOS-46985	Reuters-21578
Baseline	DNN	86.15	80.02	66.95	85.3
	CNN (Z. Yang*et al.* (2016) [23])	88.68	83.29	70.46	86.3
	RNN (Z. Yang*et al.* (2016) [23])	89.46	83.96	72.12	88.4
	NBC (C. D. Manning *et al.* (2008) [36])	78.14	68.8	46.2	83.6
	SVM (W. Zhang*et al.* (2008) [29])	85.54	80.65	67.56	86.9
	SVM (TF-IDF)(K. Chen*et al.* (2016) [27]	88.24	83.16	70.22	88.93
	Stacking SVM (A. Sun and E.-P. Lim (2001) [34])	85.68	79.45	71.81	NA
	HDLTex (K. Kowsari*et al.* (2017) [1])	90.42	86.07	76.58	NA
RMDL	3 RDLs	**90.86**	**87.39**	**78.39**	**89.10**
	9 RDLs	**92.60**	**90.65**	**81.92**	**90.36**
	15 RDLs	**92.66**	**91.01**	**81.86**	**89.91**
	30 RDLs	**93.57**	**91.59**	**82.42**	**90.69**

1) Stochastic Gradient Descent (SGD) Optimizer:

We used Stochastic Gradient Descent (SGD) that is shown in equation 32 that uses a momentum on re-scaled gradient which is shown in equation 33 for updating parameters. The other technique of optimizer that is used is RMSProp which does not do bias correction. This will be a significant problem while dealing with sparse gradient.

$$\theta \leftarrow \theta - \alpha \nabla_\theta J(\theta, x_i, y_i) \tag{32}$$

$$\theta \leftarrow \theta - \big(\gamma\theta + \alpha \nabla_\theta J(\theta, x_i, y_i)\big) \tag{33}$$

2) Adam Optimizer:

Adam is another stochastic gradient optimizer which uses only the first two moments of gradient (v and m that are shown in equation 34, 35, 36, and 37) and average over them. It can handle non-stationary of objective function as in RMSProp while overcoming the sparse gradient issue that was a drawback in RMSProp [56].

$$\theta \leftarrow \theta - \frac{\alpha}{\sqrt{\hat{v}} + \epsilon}\hat{m} \tag{34}$$

$$g_{i,t} = \nabla_\theta J(\theta_i, x_i, y_i) \tag{35}$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_{i,t} \tag{36}$$

$$m_t = \beta_2 v_{t-1} + (1-\beta_2) g_{i,t}^2 \tag{37}$$

where m_t is the first moment and v_t indicates second moment that both are estimated. $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$ and $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$

3) Multi Optimization rule:

The main idea of using multi model with different optimizers is that if one optimizer does not provide a good fit for an specific datasets, the RMDL model with n random models (some of them might use different optimizers) could ignore k models which are not efficient if and only if $n > k$. The Figure 5 provides a visual insight on how three optimizers work better in the concept of majority voting. Using multi techniques of optimizers such as SGD, adam, RMSProp, Adagrad, Adamax, and so on helps the RMDL model to be more stable for any type of datasets. In this research, we only used 3 optimizers (SGD, adam, and RMSProp) for evaluating our model, but the RMDL model has the capability to use any kind of optimizer.

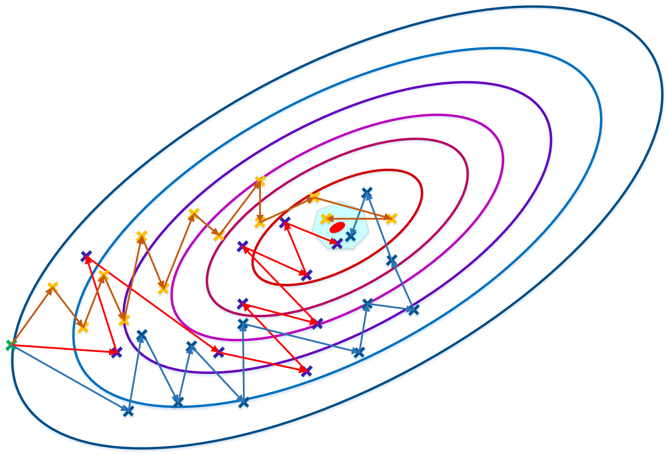


Fig. 5: This figure Shows multi SGD optimizer

VI. Experimental Results

In this section, we discuss experimental results which includes evaluation of method, experimental setup, datasets. Also, we discuss the hardware and frameworks which are used in RMDL; finally, we compare our empirical results with baselines. Moreover, losses and accuracies of this model for each individual RDL (in each epoch) is shown in Figure **??**.

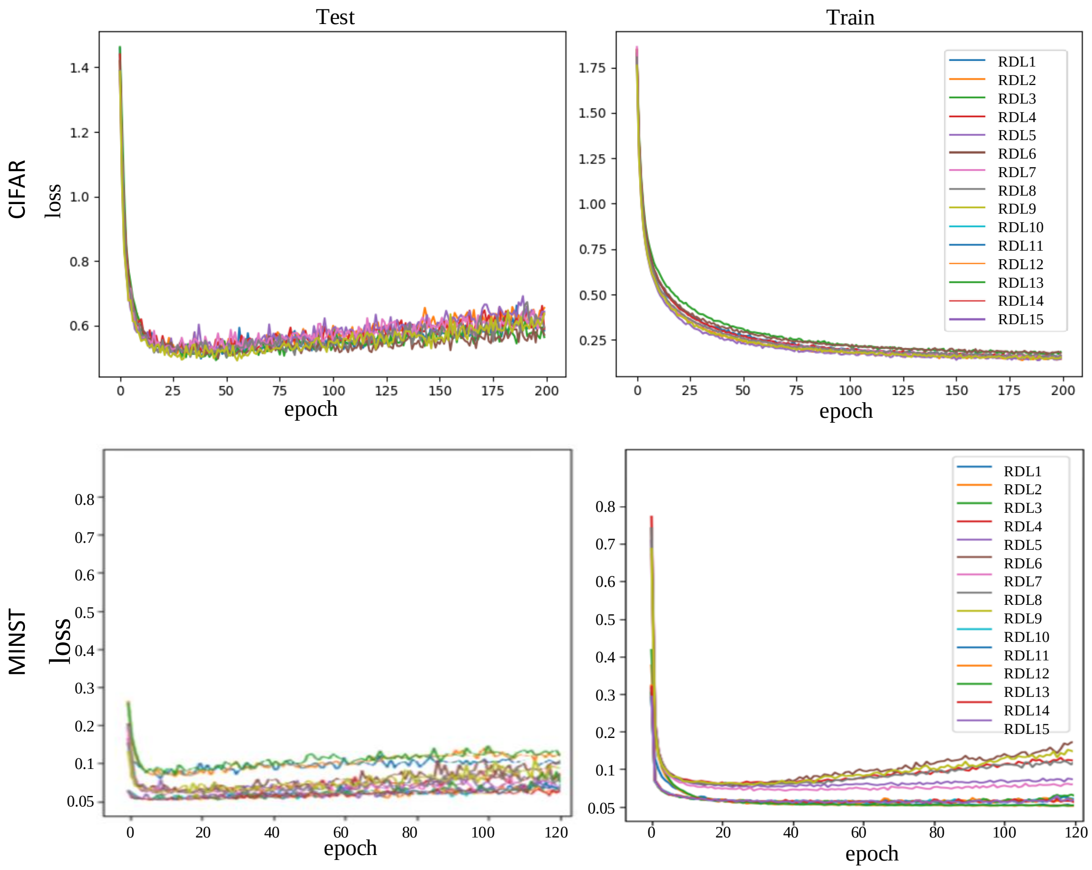


Fig. 4: This sub-figure indicates MINST and CIFAR-10 loss function for 15 Random Deep Learning (RDL) model. The MNNST shown as 120 epoch and CIFAR has 200 epoch

A. Evaluation

In this work, we report accuracy and *Micro F1-Score* which are given as follows:

$$Precision_{micro} = \frac{\sum_{l=1}^{L} TP_l}{\sum_{l=1}^{L} TP_l + FP_l} \tag{38}$$

$$Recall_{micro} = \frac{\sum_{l=1}^{L} TP_l}{\sum_{l=1}^{L} TP_l + FN_l} \tag{39}$$

$$F1 - Score_{micro} = \frac{\sum_{l=1}^{L} 2TP_l}{\sum_{l=1}^{L} 2TP_l + FP_l + FN_l} \tag{40}$$

However, the performance of our model is evaluated only in terms of F1-score for evaluation that is shown in Tables I, II, and III. Formally, given $I = \{1, 2, \cdots, k\}$ a set of indices, we define the i^{th} class as C_i. If we denote $l = |I|$ and for TP_i-true positive of C_i, FP_i-false positive, FN_i-false negative, and TN_i-true negative counts respectively then the following definitions apply for our multi-class classification problem.

B. Experimental Setup

We used two types of datasets (text and image) to test and evaluate our algorithm performance; although, in theory the model has capability to solve classification problems with a variety of data including video, text, and images.

1) Text Datasets: For text classification, we used 4 different datasets, namely, WOS , $Reuters$, $IMDB$, and $20newsgroups$.
Web Of Science (WOS) dataset is a collection of academic articles' abstracts which contains three corpora (*5736, 11967, and 46985* documents) for (*11, 34, and 134* topics).
The *Reuters-21578* news dataset contains $10,788$ documents which are divided into $7,769$ documents for training and $3,019$ for testing with total of 90 classes.
IMDB dataset contains $50,000$ reviews that is splitted into a set of $25,000$ highly polar movie reviews for training, and $25,000$ for testing.
20NewsGroup dataset includes $19,997$ documents with maximum length of $1,000$ words. In this dataset, we have $15,997$ for training and $4,000$ samples are used for validation.

2) Image datasets: For image classification, two traditional and ground truth datasets are used, namely, *MINST* hand writing dataset and *CIFAR*.
MINST: this dataset is handwritten number $k \in \{0, 1, ..., 9\}$ and input feature space is $28 \times 28 \times 1$. The training set contains $60,000$ data points, and the test set $10,000$ examples.
CIFAR: This dataset consists of $60,000$ with $32 \times 32 \times 3$ images

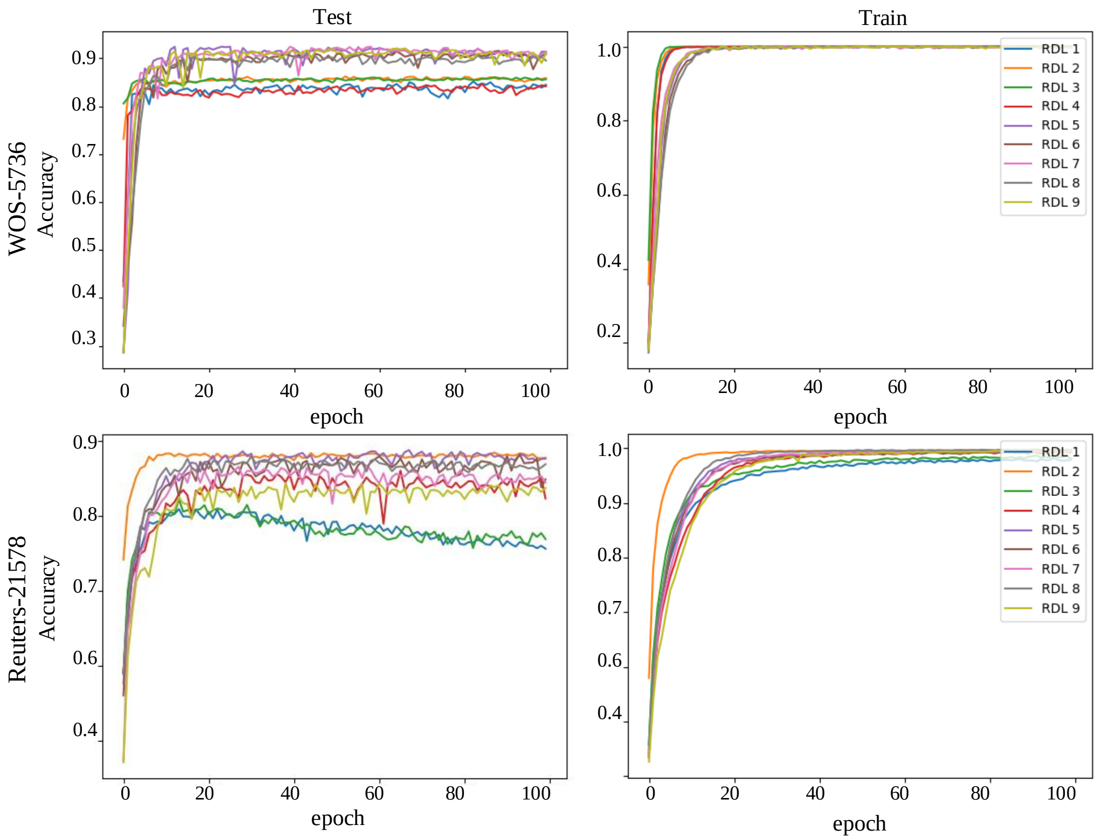


Fig. 6: This sub-figure indicates WOS-5736 (*Web Of Science dataset* with 11 categories and 5736 documents) accuracy function for 9 Random Deep Learning (RDL) model, and bottom figure indicates Reuters-21578 accuracy function for 9 Random Deep Learning (RDL) model

in 10 classes, with $6,000$ images per class that is splitted into $50,000$ training images and $10,000$ test images. Classes are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

C. Hardware

All of the results shown in this paper are performed on Central Process Units (CPU) and Graphical Process Units (GPU). Also, RMDL is capable to be performed on only GPU, CPU, or both.The processing units that has been used through this experiment was intel on *Xeon E5-2640 (2.6 GHz)* with *12 cores* and *64 GB* memory (DDR3). Also, we use three graphical cards on our machine which are two *Nvidia GeForce GTX 1080 Ti* and *Nvidia Tesla K20c*.

D. Framework

This work is implemented in Python using Compute Unified Device Architecture (CUDA) which is a parallel computing platform and Application Programming Interface (API) model created by $Nvidia$. We used $TensorFelow$ and $Keras$ library for creating the neural networks [57], [58].

TABLE II: Error rate comparison for Image classification (MNIST and CIFAR-10 datasets)

	Methods	MNIST	CIFAR-10
Baseline	Deep L2-SVM [38]	0.87	11.9
	Maxout Network [39]	0.94	11.68
	BinaryConnect [40]	1.29	9.90
	PCANet-1 [41]	0.62	21.33
	gcForest [42]	0.74	31.00
RMDL	3 RDLs	**0.51**	**9.89**
	9 RDLs	**0.41**	**9.1**
	15 RDLs	**0.21**	**8.74**
	30 RDLs	**0.18**	**8.79**

E. Empirical Results

1) Image classification: Table II shows the error rate of RMDL for image classification. The comparison between the RMDL with baselines (as described in Section III-B), shows that the error rate of the RMDL for MNIST dataset has been improved to 0.51, 0.41, and 0.21 for 3, 9 and 15 random models respectively. For the CIFAR-10 datasets, the error rate has been decreased for RMDL to 9.89, 9.1, 8.74, and 8.79,using 3, 9, 15, and 30 RDL respectively.

2) Document categorization: Table I shows that for four ground truth datasets, RMDL improved the accuracy in comparison to the baselines. In Table I, we evaluated our empirical results by three different

RMDL models (using 3, 9, 15, and 30 RDLs). For *Web of Science (WOS-5,736)* the accuracy is improved to 90.86, 92.60, 92.66, and 93.57 respectively. For *Web of Science (WOS-11,967)*, the accuracy is increased to 87.39, 90.65, 91.01, and 91.59 respectively, and for *Web of Science (WOS-46,985)* the accuracy has increased to 78.39, 81.92, 81.86, and 82.42 respectively. The accuracy of *Reuters-21578* is 88.95, 90.29, 89.91, and 90.69 respectively. We report results for other ground truth datasets such as *Large Movie Review Dataset (IMDB)* and *20NewsGroups*. As it is mentioned in Table III, for two ground truth datasets, RMDL improves the accuracy. In Table III, we evaluated our empirical results of two datasets (*IMDB reviewer and 20NewsGroups*).The accuracy of *IMDB* dataset is 89.91, 90.13, and 90.79 for 3, 9, and 15 RDLs respectively, whereas the accuracy of DNN is 88.55%, CNN [23] is 87.44%, RNN [23] is 88.59%, Naïve Bayes Classifier is 83.19%, SVM [29] is 87.97%, and SVM [27] using TF-IDF is equal to 88.45%. The accuracy of *20NewsGroup* dataset is 86.73%, 87.62%, and 87.91% for 3, 9, and 15 random models respectively, whereas the accuracy of DNN is 86.50%, CNN [23] is 82.91%, RNN [23] is 83.75%, Naïve Bayes Classifier is 81.67%, SVM [29] is 84.57%, and SVM [27] using TF-IDF is equal to 86.00%.

TABLE III: Accuracy comparison for text classification on IMDB and 20NewsGroup datasets

	Model	Dataset	
		IMDB	20NewsGroup
Baseline	DNN	88.55	86.50
	CNN [23]	87.44	82.91
	RNN [23]	88.59	83.75
	Naïve Bayes Classifier	83.19	81.67
	SVM [29]	87.97	84.57
	SVM(TF-IDF) [27]	88.45	86.00
RMDL	3 RDLs	**89.91**	**86.73**
	9 RDLs	**90.13**	**87.62**
	15 RDLs	**90.79**	**87.91**

Figure 4 and 6 indicate accuracies and losses of RMDL which are shown with 9 (RDLs) for text classification and 15 RDLs for image classification. As shown in Figure 4, 4 RDLs' loss of MINST dataset are increasing over each epochs (RDL 6, RDL 9, RDL 14 and RDL 15) after 40 epochs, but RMDL model contains 15 RDL models; thus, the accuracy of the majority votes for these models as presented in Table II is competing with our baselines.

In Figure 4, for CIFAR dataset, the models don't have overfitting problem, but for MINST datasets at least 4 models' losses are increasing over each epoch after 40 iteration (RDL 4, RDL 5, RDL 6, and RDL 9); although the accuracy and F1-measure of these 4 models will drop after 40 epochs, the majority votes' accuracy is rubost and efficient which means RMDL performance will ignore them due to majority votes between 15 models. The Figure 4 shows the loss value over each epoch of two ground truth datasets, *CIFAR* and *IMDB* for 15 random deep learning models (RDL). The Figure 6 indicates the accuracy of 15 random model for *Reuters-21578* respectively. In Figure 6, the accuracy of Random Deep Learning (RDLs) model is addressed over each epochs for WOS-5736 (*Web Of Science dataset* with 17 categories and 5,736 documents), the majority votes of theses models as shown in Table I is competing with our baselines.

VII. Discussion and Conclusion

The classification task is an important problem to address in machine learning, given the growing number and size of datasets that need sophisticated classification. We propose a novel technique to solve the problem of choosing best technique and method out of many possible structures and architectures in deep learning. This paper introduces a new approach called RMDL (Random Multimodel Deep Learning) for the classification that combines multi deep learning approaches to produce random classification models. Our evaluation on datasets obtained from the Web of Science (WOS), Reuters, MINST, CIFAR, IMDB, and 20NewsGroups shows that combinations of DNNs, RNNs and CNNs with the parallel learning architecture, has consistently higher accuracy than those obtained by conventional approaches using naïve Bayes, SVM, or single deep learning model. These results show that deep learning methods can provide improvements for classification and that they provide flexibility to classify datasets by using majority vote. The proposed approach has the ability to improve accuracy and efficiency of models and can be use across a wide range of data types and applications.

References

[1] K. Kowsari, D. E. Brown, M. Heidarysafa, K. Jafari Meimandi, M. S. Gerber, and L. E. Barnes, "Hdltex: Hierarchical deep learning for text classification," in *ICMLA*, Dec 2017, pp. 364–371.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 609–616.

[4] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[5] M. Turan, Y. Almalioglu, H. Araujo, E. Konukoglu, and M. Sitti, "Deep endovo: A recurrent convolutional neural network (rcnn) based visual odometry approach for endoscopic capsule robots," *arXiv preprint arXiv:1708.06822*, 2017.

[6] K. Kowsari, M. Heidarysafa, D. E. Brown, K. Jafari Meimandi, and L. E. Barnes, "Rmdl: Random multimodel deep learning for classification," in *Proceedings of the 2018 International Conference on Information System and Data Mining*. ACM, 2018.

[7] S. Robertson, "Understanding inverse document frequency: on theoretical arguments for idf," *Journal of documentation*, vol. 60, no. 5, pp. 503–520, 2004.

[8] L. E. Krueger and R. G. Shapiro, "Letter detection with rapid serial visual presentation: Evidence against word superiority at feature extraction." *Journal of Experimental Psychology: Human Perception and Performance*, vol. 5, no. 4, p. 657, 1979.

[9] H. P. Luhn, "A statistical approach to mechanized encoding and searching of literary information," *IBM Journal of research and development*, vol. 1, no. 4, pp. 309–317, 1957.

[10] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information processing & management*, vol. 24, no. 5, pp. 513–523, 1988.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[12] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[13] K. P. Murphy, "Naive bayes classifiers," *University of British Columbia*, 2006.

[14] I. Rish, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22. IBM, 2001, pp. 41–46.

[15] C.-N. J. Yu and T. Joachims, "Learning structural svms with latent variables," in *ICML*. ACM, 2009, pp. 1169–1176.

[16] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *Journal of machine learning research*, vol. 2, no. Nov, pp. 45–66, 2001.

[17] F. Kabir, S. Siddique, M. R. A. Kotwal, and M. N. Huda, "Bangla text document categorization using stochastic gradient descent (sgd) classifier," in *CCIP*. IEEE, 2015, pp. 1–4.

[18] D. CireşAn, U. Meier, J. Masci, and J. Schmidhuber, "Multi-column deep neural network for traffic sign classification," *Neural Networks*, vol. 32, pp. 333–338, 2012.

[19] 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.

[20] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model." in *Interspeech*, vol. 2, 2010, p. 3.

[23] Z. Yang, D. Yang, C. Dyer, X. He, A. J. Smola, and E. H. Hovy, "Hierarchical attention networks for document classification." in *HLT-NAACL*, 2016, pp. 1480–1489.

[24] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3367–3375.

[25] A. Y. Chervonenkis, "Early history of support vector machines," in *Empirical Inference*. Springer, 2013, pp. 13–20.

[26] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *COLT92*. ACM, 1992, pp. 144–152.

[27] K. Chen, Z. Zhang, J. Long, and H. Zhang, "Turning from tf-idf to tf-igm for term weighting in text classification," *Expert Systems with Applications*, vol. 66, pp. 245–260, 2016.

[28] J. Weston and C. Watkins, "Multi-class support vector machines," Department of Computer Science, Royal Holloway, University of London, Tech. Rep., 1998.

[29] W. Zhang, T. Yoshida, and X. Tang, "Text classification based on multi-word with support vector machine," *Knowledge-Based Systems*, vol. 21, no. 8, pp. 879–886, 2008.

[30] C. S. Leslie, E. Eskin, and W. S. Noble, "The spectrum kernel: A string kernel for svm protein classification." in *Pacific symposium on biocomputing*, vol. 7, no. 7, 2002, pp. 566–575.

[31] E. Eskin, J. Weston, W. S. Noble, and C. S. Leslie, "Mismatch string kernels for svm protein classification," in *Advances in neural information processing systems*, 2002, pp. 1417–1424.

[32] C. S. Leslie, E. Eskin, A. Cohen, J. Weston, and W. S. Noble, "Mismatch string kernels for discriminative protein classification," *Bioinformatics*, vol. 20, no. 4, pp. 467–476, 2004.

[33] R. Singh, A. Sekhon, K. Kowsari, J. Lanchantin, B. Wang, and Y. Qi, "Gakco: a fast gapped k-mer string kernel using counting," *arXiv preprint arXiv:1704.07468*, 2017.

[34] A. Sun and E.-P. Lim, "Hierarchical text classification and evaluation," in *ICDM*. IEEE, 2001, pp. 521–528.

[35] F. Sebastiani, "Machine learning in automated text categorization," *ACM computing surveys (CSUR)*, vol. 34, no. 1, pp. 1–47, 2002.

[36] C. D. Manning, P. Raghavan, H. Schütze *et al.*, *Introduction to information retrieval*. Cambridge university press Cambridge, 2008, vol. 1, no. 1.

[37] S.-B. Kim, K.-S. Han, H.-C. Rim, and S. H. Myaeng, "Some effective techniques for naive bayes text classification," *IEEE transactions on knowledge and data engineering*, vol. 18, no. 11, pp. 1457–1466, 2006.

[38] Y. Tang, "Deep learning using linear support vector machines," *arXiv preprint arXiv:1306.0239*, 2013.

[39] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," *arXiv preprint arXiv:1302.4389*, 2013.

[40] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems*, 2015, pp. 3123–3131.

[41] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, "Pcanet: A simple deep learning baseline for image classification?" *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5017–5032, 2015.

[42] Z.-H. Zhou and J. Feng, "Deep forest: Towards an alternative to deep neural networks," *arXiv preprint arXiv:1702.08835*, 2017.

[43] H. Hotta, M. Kittaka, and M. Hagiwara, "Word vectorization using relations among words for neural network," *IEEE Transactions on Electronics, Information and Systems*, vol. 130, no. 1, pp. 75–82, 2010.

[44] V. Kešelj, F. Peng, N. Cercone, and C. Thomas, "N-gram-based author profiles for authorship attribution," in *Proceedings of the conference pacific association for computational linguistics, PACLING*, vol. 3, 2003, pp. 255–264.

[45] K. Dave, S. Lawrence, and D. M. Pennock, "Mining the peanut gallery: Opinion extraction and semantic classification of product reviews," in *WWW*. ACM, 2003, pp. 519–528.

[46] W. B. Cavnar, J. M. Trenkle *et al.*, "N-gram-based text categorization," *Ann Arbor MI*, vol. 48113, no. 2, pp. 161–175, 1994.

[47] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[48] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[49] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[50] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks." *ICML (3)*, vol. 28, pp. 1310–1318, 2013.

[51] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification." in *AAAI*, vol. 333, 2015, pp. 2267–2273.

[52] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[53] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[54] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," *Artificial Neural Networks–ICANN 2010*, pp. 92–101, 2010.

[55] R. Johnson and T. Zhang, "Effective use of word order for text categorization with convolutional neural networks," *arXiv preprint arXiv:1412.1058*, 2014.

[56] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[57] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[58] F. Chollet *et al.*, "Keras: Deep learning library for theano and tensorflow," *URL: https://keras. io/k*, 2015.

Kamran Kowsari is PhD student at School of Engineering and Applied Science of University of Virginia, Departement of Systems and Information Engineering. He earned his Master of Science from Department of Computer Science at The George Washington University in 2014.

He has more than 10 year's experiences in software development, system and database engineering experience, and research. His experience includes numerous projects and academic projects. He is interested and has experience in Machine learning, mathematical modeling, algorithms and data structure, Real-time rendering use machine learning, and data manning.

Mojtaba Heidarysafa is PhD student at School of Engineering and Applied Science of University of Virginia, Departement of Systems and Information Engineering. He earned his Master of Science from .... in 2015. He has more than 10 year's experiences in software development, system and database engineering experience, and research. His experience includes numerous projects and academic projects. He is interested and has experience in Machine learning, mathematical modeling, algorithms and data structure, Real-time rendering use machine learning, and data manning.

Donald E. Brown (F'01) received the B.S. degree from the U.S. Military Academy, West Point, NY, USA, the M.S. and M.E. degrees from the University of California, Berkeley, CA, USA, and the Ph.D. degree from the University of Michigan, Ann Arbor, MI, USA. He is currently the Director of the Data Science Institute, University of Virginia, Charlottesville, VA, USA, and the William Stansfield Calcott Professor of Systems and Information Engineering. His research focuses on techniques that enable the combination of different types of data for prediction and engineering design. Dr. Brown was a recipient of the IEEE Joseph Wohl Career Achievement Award and the IEEE Norbert Wiener Award for Outstanding Research.

Kiana Jafari Meimandi is PhD student at School of Engineering and Applied Science of University of Virginia, Departement of Systems and Information Engineering. She earned her M.S. from Khaje Nasir University of Technology in 2012.

Matthew S. Gerber received his Ph.D. in computer science from Michigan State University in 2011 and is currently a Research Assistant Professor in the Department of Systems and Information Engineering at the University of Virginia.

His research interests include natural language processing, data mining, and predictive modeling with applications in crime analysis and medical informatics.

Laura E. Barnes (M'03) received the B.S. degree in computer science from Texas Tech University, Lubbock, in 2003 and the M.S. and Ph.D. degrees in computer science and engineering from the University of South Florida, Tampa, in 2007 and 2008, respectively.

She was a Research Assistant with the Center for Robot-Assisted Search and Rescue from 2003 to 2005 and with the Unmanned Systems Laboratory, University of South Florida, from 2005 to 2008. In addition, she held a fellowship from the Army Research Laboratory, where she spent two summers with the Vehicles Technology Directorate. In June 2008, she joined the Automation and Robotics Research Institute, University of Texas, Arlington, as a Faculty Associate Researcher. Her research interests include swarms, unmanned aerial vehicles, and distributed/intelligent systems.