RMDL: Random Multimodel Deep Learning for Classification

Kamran Kowsari
Department of System and Information Engineering, University of Virginia
Charlottesville, VA, USA
kk7nc@virginia.edu

Mojtaba Heidarysafa
Department of System and Information Engineering, University of Virginia
Charlottesville, VA, USA
mh4pk@virginia.edu

Donald E. Brown*†
Department of System and Information Engineering, University of Virginia
Charlottesville, VA, USA
deb@virginia.edu

Kiana J. Meimandi
Department of System and Information Engineering, University of Virginia
Charlottesville, VA, USA
kj6vd@virginia.edu

Laura E. Barnes*‡
Department of System and Information Engineering, University of Virginia
Charlottesville, VA, USA
lb3dp@virginia.edu

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]

CCS CONCEPTS

• **Information systems → Decision support systems, Data mining**;

KEYWORDS

Data Mining, Text Classification, Image Classification, Deep Neural Networks, Deep Learning, Supervised Learning

*Data Science Institute, University of Virginia Charlottesville, VA USA
†Predictive Technology Laboratory, University of Virginia,Charlottesville, VA USA
‡Sensing Systems for Health Lab, University of Virginia,Charlottesville, VA USA
[1]Code is shared as an open source tool at https://github.com/XXXXXXXXX/RMDL

ICISDM'18, April 2018, Florida USA

ACM ISBN 978-1-4503-6354-9…$15.00
https://doi.org/10.1145/000

ACM Reference Format:
Kamran Kowsari, Mojtaba Heidarysafa, Donald E. Brown, Kiana J. Meimandi, and Laura E. Barnes. 2018. RMDL: Random Multimodel Deep Learning for Classification. In *Proceedings of ACM conference (ICISDM'18).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/000

1 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 [9, 23, 27, 29, 50]), 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.

This paper describes an approach to this challenge using ensembles of deep learning architectures. 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) [42]. 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 2 gives related work for feature extraction, other classification techniques, and deep learning for classification task; Section 3 describes current techniques for classification tasks which are used as our baseline; Section 4 describes Random Multimodel Deep Learning methods and the architecture for RMDL including Section 4.1 shows feature extraction in RMDL, Section 4.2 talks about overall view of RMDL; Section 4.3 addresses the deep learning structure used in this model, Section 4.4 discusses optimization problem; Section 5.1 talks about evaluation of these techniques; Section 5 shows the experimental results which includes the accuracy and performance of RMDL; and finally, Section 6 presents discussion and conclusions of our work.

2 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 [25] 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 [33] introduced weighted values for each word and then G. Salton *et. al.* in 1988 [43] 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.* [35] 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 [40] 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 [37] and I. Rish in 2001 [41] 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 [53] introduced latent variables into the discriminative model as a new structure for SVM, and S. Tong *et. al.* in 2001 [49] 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) [18] 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 [10] 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 [24] used CNN, but they have used $2D$ convolutional layers combined with the $2D$ feature space of the image. Another example of CNN in [27] showed excellent accuracy for image classification. This architecture can also be used for text classification as shown in the work of [21]. 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) [9] or Long Short-Term Memory (LSTM) units [15]. These new units help control for instability problems in the original network architecure. RNN have been successfully used for natural language processing [36]. Recently, Z. Yang *et. al.* in 2016 [52] 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 [50] and M. Liang *et. al.*in 2015 [32] implemented innovative combinations of CNN and RNN called *A Recurrent Convolutional Neural Network (RCNN)*. K. Kowsari *et. al.* in 2017 [23] 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 [23] to provide a more general approach to supervised learning for classification.

3 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:

3.1 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) [23].

3.1.1 Deep Learning. The baseline, we used in this paper is Deep Learning without Hierarchical level. One of our baselines for text classification is [52]. In our methods' Section 4, we will explain the basic models of deep learning such as DNN, CNN, and RNN which are used as part of RMDL model.

3.1.2 Support Vector Machine (SVM). The original version of SVM was introduced by Vapnik, VN and Chervonenkis, A Ya [6] in 1963. The early 1990s, nonlinear version was addressed in [3].

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.

The natural way to solve k-class problem is to construct a decision function of all k classes at once [5, 51]. 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 [54], 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 [13, 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 $x \rightarrow \mathbb{R}^{l^k}$

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 [30]. The main problem of SVM for string sequences is time complexity of these models. S. Ritambhara *et. al.* in 2017 [46] addressed the problem of time for gap k-mers kernel called *GaKCo* which is used only for protein and DNA sequences.

3.1.3 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 [45, 47].

3.1.4 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 [34]. 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:

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)} \tag{1}$$

where d is document, c indicates classes.

$$\begin{aligned} C_{MAP} &= \arg\max_{c \in C} P(d \mid c)P(c) \\ &= \arg\max_{c \in C} P(x_1, x_2, ..., x_n \mid c)p(c) \end{aligned} \tag{2}$$

The baseline of this paper is word level of NBC [20] as follows:

$$P(c_j \mid d_i; \hat{\theta}) = \frac{P(c_j \mid \hat{\theta})P(d_i \mid c_j; \hat{\theta}_j)}{P(d_i \mid \hat{\theta})} \tag{3}$$

3.1.5 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 [23] 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.

3.2 Image Classification Baselines

For image classification, we have five baselines as follows: Deep L2-SVM [48], Maxout Network [14], BinaryConnect [11], PCANet-1 [4], and gcForest [55].

Deep L2-SVM: This technique is known as deep learning using linear support vector machines which simply softmax is replaced with linear SVMs [48].

Maxout Network: I. Goodfellow*et. al.* in 2013 [14] 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 [11] 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 [4] 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 [55] 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.

4 METHOD

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.

4.1 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.

4.1.1 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.

4.1.2 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 [16] for extracting features; besides, we also can use N-gram algorithm for extracting features for neural deep learning [12, 19]. For example, feature extraction in this 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 [40]. A vector-space model is a mathematical mapping of the word space, defined as

$$d_j = (w_{1,j}, w_{2,j}, ..., w_{i,j}..., w_{l_j,j}) \tag{4}$$

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.

4.2 Random Multimodel Deep Learning

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).

$$M(y_{i1}, y_{i2}, ..., y_{in}) = \left\lfloor \frac{1}{2} + \frac{(\sum_{j=1}^{n} y_{ij}) - \frac{1}{2}}{n} \right\rfloor \tag{5}$$

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 5 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:

$$\hat{y_i} = \left[\hat{y}_{i1} \ \ldots \ \hat{y}_{ij} \ \ldots \ \hat{y}_{in}\right]^T \tag{6}$$

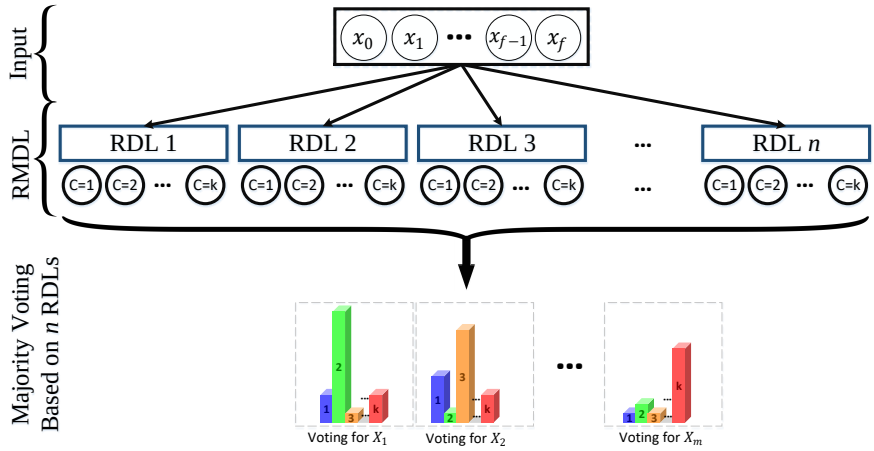


Figure 1: Overview of RDML: Random Multimodel Deep Learning for classification that includes n Random models which are d random model of DNN classifiers, c models of CNN classifiers, and r RNN classifiers where $r + c + d = n$.

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:

$$\hat{y}_{i,j} = arg\max_{k}[softmax(y^*_{i,j})] \tag{7}$$

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

4.3 Deep Learning in RMDL

The RMDL model structure (section 4.2) includes three basic techniques of deep learning in parallel. We describe each individual model separately. The final model contains d random DNNs (Section 4.3.1), r RNNs (Section 4.3.2), and c CNNs models (Section 4.3.3).

4.3.1 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 8), ReLU [38] (equation 9) as activation function. The output layer for multi-class classification, should use $Softmax$ equation 10.

$$f(x) = \frac{1}{1 + e^{-x}} \in (0, 1) \tag{8}$$

$$f(x) = \max(0, x) \tag{9}$$

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \forall\, j \in \{1, \ldots, K\} \tag{10}$$

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.

4.3.2 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 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 11 where x_t is the state at time t and $\boldsymbol{u_t}$ refers to the input at step t.

$$x_t = F(x_{t-1}, \boldsymbol{u_t}, \theta) \tag{11}$$

More specifically, we can use weights to formulate the Equation 11 with specified parameters in Equation 12

$$x_t = \mathbf{W_{rec}}\sigma(x_{t-1}) + \mathbf{W_{in}}\mathbf{u_t} + \mathbf{b} \tag{12}$$

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 [2].

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

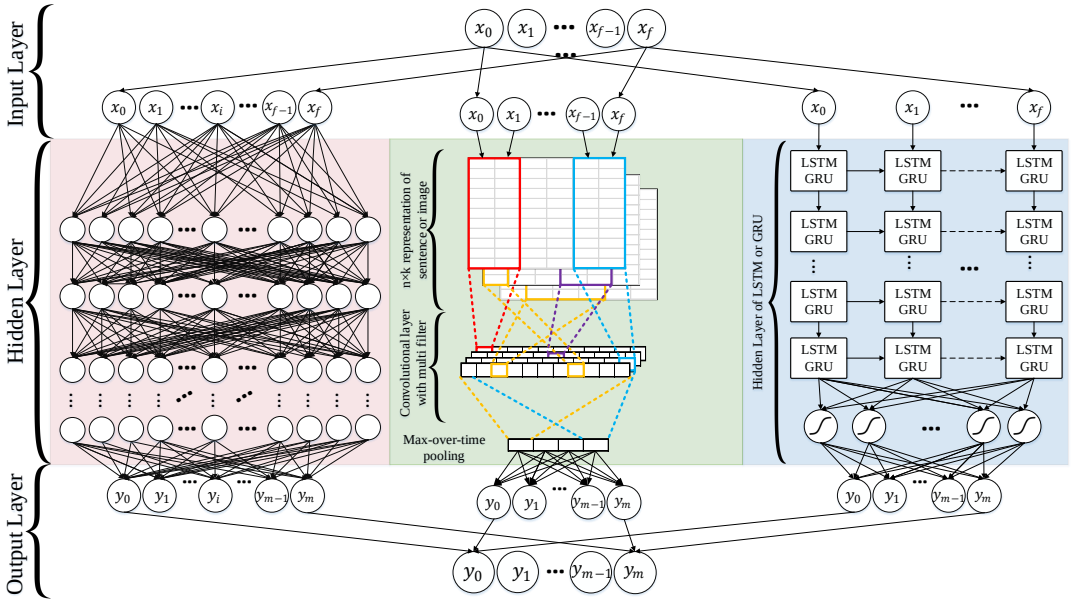


Figure 2: Random Multimodel Deep Learning (RDML) architecture for classification which includes 3 Random models, a DNN classifier at left, a Deep CNN classifier at middle, and a Deep RNN classifier at right (each unit could be LSTM or GRU).

vanishing gradient problem [39]. 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:

$$i_t = \sigma(W_i[x_t, h_{t-1}] + b_i), \quad (13)$$
$$\tilde{C}_t = \tanh(W_c[x_t, h_{t-1}] + b_c), \quad (14)$$
$$f_t = \sigma(W_f[x_t, h_{t-1}] + b_f), \quad (15)$$
$$C_t = i_t * \tilde{C}_t + f_t C_{t-1}, \quad (16)$$
$$o_t = \sigma(W_o[x_t, h_{t-1}] + b_o), \quad (17)$$
$$h_t = o_t \tanh(C_t), \quad (18)$$

Where equation 13 is input gate, Equation 14 shows candid memory cell value, Equation 15 is forget gate activation, Equation 16 is new memory cell value, and Equation 17 and 18 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 4.3.3 were introduced which deploys a max-pooling layer to determine descriminative phrases in a text [26].

Gated Recurrent Unit (GRU): Gated Recurrent Unit (GRU) is a gating mechanism for RNN which was introduced by [9] and [7]. 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 [7]. 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:

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1}] + b_z), \quad (19)$$

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.

$$\tilde{r_t} = \sigma_g(W_r x_t + U_r h_{t-1}] + b_r), \quad (20)$$

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.

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h) \quad (21)$$

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.

4.3.3 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 [28]; 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 [44]. 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) [17]. This means the dimensionality of the CNN for text is very high.

4.4 Optimization

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

4.4.1 Stochastic Gradient Descent (SGD) Optimizer. We used Stochastic Gradient Descent (SGD) that is shown in equation 22 that uses a momentum on re-scaled gradient which is shown in equation 23 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.

$$\theta \leftarrow \theta - \alpha \nabla_\theta J(\theta, x_i, y_i) \tag{22}$$

$$\theta \leftarrow \theta - \left(\gamma\theta + \alpha \nabla_\theta J(\theta, x_i, y_i)\right) \tag{23}$$

4.4.2 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 24, 25, 26, and 27) and average

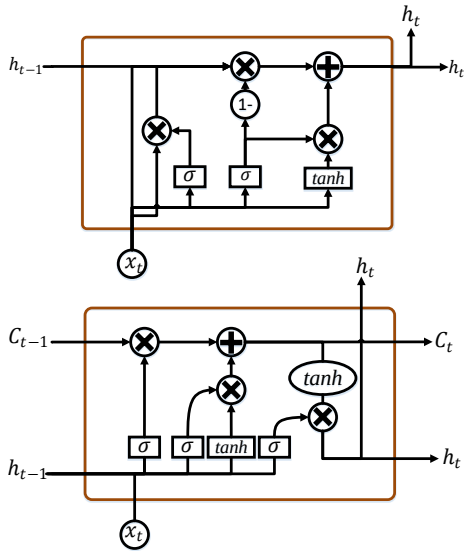


Figure 3: Left Figure is a cell of GRU, and right Figure is a cell of LSTM

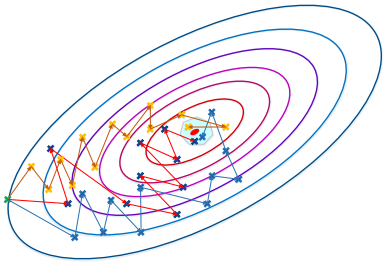


Figure 4: This figure Shows multi SGD optimizer

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 [22].

$$\theta \leftarrow \theta - \frac{\alpha}{\sqrt{\hat{v}} + \epsilon}\hat{m} \tag{24}$$

$$g_{i,t} = \nabla_\theta J(\theta_i, x_i, y_i) \tag{25}$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_{i,t} \tag{26}$$

$$m_t = \beta_2 v_{t-1} + (1 - \beta_2) g_{i,t}^2 \tag{27}$$

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 $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$

4.4.3 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 4 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.

5 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 5.

5.1 Evaluation

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

$$Precision_{micro} = \frac{\sum_{l=1}^{L} TP_l}{\sum_{l=1}^{L} TP_l + FP_l} \tag{28}$$

$$Recall_{micro} = \frac{\sum_{l=1}^{L} TP_l}{\sum_{l=1}^{L} TP_l + FN_l} \tag{29}$$

$$F1 - Score_{micro} = \frac{\sum_{l=1}^{L} 2TP_l}{\sum_{l=1}^{L} 2TP_l + FP_l + FN_l} \tag{30}$$

However, the performance of our model is evaluated only in terms of F1-score for evaluation that is shown in Tables 1, 2, and 3. 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.

5.2 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.

5.2.1 Text Datasets. For text classification, we used 4 different datasets, namely, *WOS* , *Reuters*, *IMDB*, and 20*newsgroups*.
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.

5.2.2 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 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.

Table 1: 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			
		W.1	W.2	W.3	R
Baseline	DNN	86.15	80.02	66.95	85.3
	CNN [52]	88.68	83.29	70.46	86.3
	RNN [52]	89.46	83.96	72.12	88.4
	NBC	78.14	68.8	46.2	83.6
	SVM [54]	85.54	80.65	67.56	86.9
	SVM (TF-IDF) [5]	88.24	83.16	70.22	88.93
	Stacking SVM [47]	85.68	79.45	71.81	NA
	HDLTex [23]	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**

Table 2: Error rate comparison for Image classification (MNIST and CIFAR-10 datasets)

Methods		MNIST	CIFAR-10
Baseline	Deep L2-SVM [48]	0.87	11.9
	Maxout Network [14]	0.94	11.68
	BinaryConnect [11]	1.29	9.90
	PCANet-1 [4]	0.62	21.33
	gcForest [55]	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**

5.3 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*.

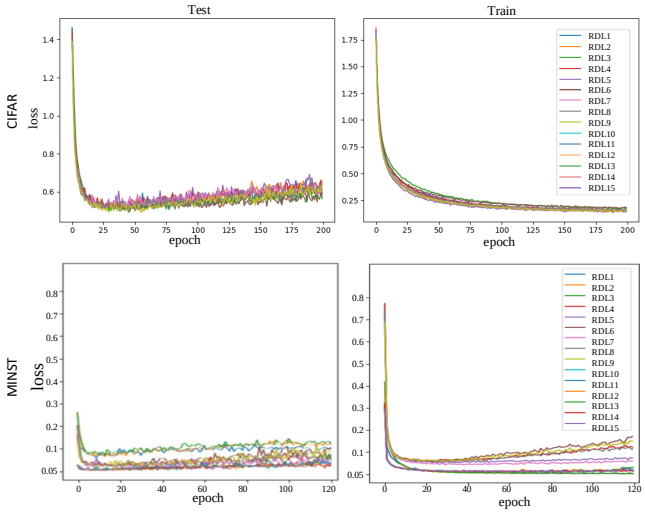
5.4 Framework

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 [1, 8].

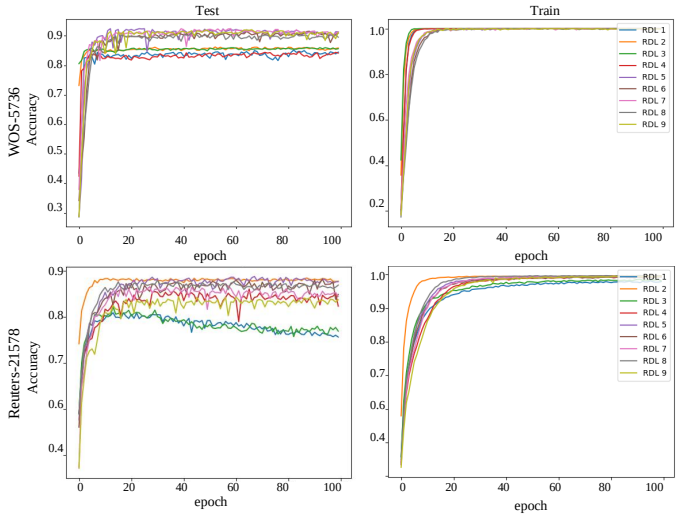
5.5 Empirical Results

5.5.1 Image classification. Table 2 shows the error rate of RMDL for image classification. The comparison between the RMDL with baselines (as described in Section 3.2), 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.

5.5.2 Document categorization. Table 1 shows that for four ground truth datasets, RMDL improved the accuracy in comparison to the baselines. In Table 1, 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 3, for two ground truth datasets, RMDL improves the accuracy. In Table 3, 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 [52] is 87.44%, RNN [52] is 88.59%, Naïve Bayes Classifier is 83.19%, SVM [54] is 87.97%, and SVM [5] using



(a) 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



(b) 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

Figure 5: This figure shows results of individual RDLs (accuracy and loss) for each epoch as part of RMDL.

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 [52] is 82.91%, RNN [52] is 83.75%, Naïve Bayes Classifier is 81.67%, SVM [54] is 84.57%, and SVM [5] using TF-IDF is equal to 86.00%.

Figure 5 indicates accuracies and losses of RMDL which are shown with 9 (RDLs) for text classification and 15 RDLs for image classification. As shown in Figure 5a, 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 2 is competing with our baselines.
In Figure 5a, 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 5a shows the loss value over each epoch of two ground truth datasets, *CIFAR* and *IMDB* for 15 random deep learning models (RDL).The Figure 5b indicates the accuracy of 15 random model for *Reuters-21578* respectively. In Figure 5b, 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 1 is competing with our baselines.

Table 3: Accuracy comparison for text classification on IMDB and 20NewsGroup datasets

	Model	Dataset	
		IMDB	20NewsGroup
Baseline	DNN	88.55	86.50
	CNN [52]	87.44	82.91
	RNN [52]	88.59	83.75
	Naïve Bayes Classifier	83.19	81.67
	SVM [54]	87.97	84.57
	SVM(TF-IDF) [5]	88.45	86.00
RMDL	3 RDLs	**89.91**	**86.73**
	9 RDLs	**90.13**	**87.62**
	15 RDLs	**90.79**	**87.91**

6 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] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).

[2] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* 5, 2 (1994), 157–166.

[3] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. 1992. A training algorithm for optimal margin classifiers. In *COLT92*. ACM, 144–152.

[4] Tsung-Han Chan, Kui Jia, Shenghua Gao, Jiwen Lu, Zinan Zeng, and Yi Ma. 2015. PCANet: A simple deep learning baseline for image classification? *IEEE Transactions on Image Processing* 24, 12 (2015), 5017–5032.

[5] Kewen Chen, Zuping Zhang, Jun Long, and Hao Zhang. 2016. Turning from TF-IDF to TF-IGM for term weighting in text classification. *Expert Systems with Applications* 66 (2016), 245–260.

[6] Alexey Ya Chervonenkis. 2013. Early history of support vector machines. In *Empirical Inference*. Springer, 13–20.

[7] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[8] François Chollet et al. 2015. Keras: Deep learning library for theano and tensorflow. *URL: https://keras. io/k* (2015).

[9] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[10] Dan CireşAn, Ueli Meier, Jonathan Masci, and Jürgen Schmidhuber. 2012. Multi-column deep neural network for traffic sign classification. *Neural Networks* 32 (2012), 333–338.

[11] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. 2015. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*. 3123–3131.

[12] Kushal Dave, Steve Lawrence, and David M Pennock. 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *WWW*. ACM, 519–528.

[13] Eleazar Eskin, Jason Weston, William S Noble, and Christina S Leslie. 2002. Mismatch string kernels for SVM protein classification. In *Advances in neural information processing systems*. 1417–1424.

[14] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. *arXiv preprint arXiv:1302.4389* (2013).

[15] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[16] Hajime Hotta, Masanobu Kittaka, and Masafumi Hagiwara. 2010. Word vectorization using relations among words for neural network. *IEEE Transactions on Electronics, Information and Systems* 130, 1 (2010), 75–82.

[17] Rie Johnson and Tong Zhang. 2014. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058* (2014).

[18] Fasihul Kabir, Sabbir Siddique, Mohammed Rokibul Alam Kotwal, and Mohammad Nurul Huda. 2015. Bangla text document categorization using stochastic gradient descent (sgd) classifier. In *CCIP*. IEEE, 1–4.

[19] Vlado Kešelj, Fuchun Peng, Nick Cercone, and Calvin Thomas. 2003. N-gram-based author profiles for authorship attribution. In *Proceedings of the conference pacific association for computational linguistics, PACLING*, Vol. 3. 255–264.

[20] Sang-Bum Kim, Kyoung-Soo Han, Hae-Chang Rim, and Sung Hyon Myaeng. 2006. Some effective techniques for naive bayes text classification. *IEEE transactions on knowledge and data engineering* 18, 11 (2006), 1457–1466.

[21] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).

[22] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[23] Kamran Kowsari, Donald E Brown, Mojtaba Heidarysafa, Kiana Jafari Meimandi, Matthew S Gerber, and Laura E Barnes. [n. d.].

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[25] Lester E Krueger and Ronald G Shapiro. 1979. Letter detection with rapid serial visual presentation: Evidence against word superiority at feature extraction. *Journal of Experimental Psychology: Human Perception and Performance* 5, 4 (1979), 657.

[26] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent Convolutional Neural Networks for Text Classification.. In *AAAI*, Vol. 333. 2267–2273.

[27] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[28] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[29] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. 2009. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*. ACM, 609–616.

[30] Christina S Leslie, Eleazar Eskin, Adiel Cohen, Jason Weston, and William Stafford Noble. 2004. Mismatch string kernels for discriminative protein classification. *Bioinformatics* 20, 4 (2004), 467–476.

[31] Christina S Leslie, Eleazar Eskin, and William Stafford Noble. 2002. The spectrum kernel: A string kernel for SVM protein classification.. In *Pacific symposium on biocomputing*, Vol. 7. 566–575.

[32] Ming Liang and Xiaolin Hu. 2015. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3367–3375.

[33] Hans Peter Luhn. 1957. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development* 1, 4 (1957), 309–317.

[34] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. 2008. *Introduction to information retrieval*. Vol. 1. Cambridge university press Cambridge.

[35] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[36] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model.. In *Interspeech*, Vol. 2. 3.

[37] Kevin P Murphy. 2006. Naive bayes classifiers. *University of British Columbia* (2006).

[38] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. 807–814.

[39] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML (3)* 28 (2013), 1310–1318.

[40] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[41] Irina Rish. 2001. An empirical study of the naive Bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, Vol. 3. IBM, 41–46.

[42] Stephen Robertson. 2004. Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of documentation* 60, 5 (2004), 503–520.

[43] Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management* 24, 5 (1988), 513–523.

[44] Dominik Scherer, Andreas Müller, and Sven Behnke. 2010. Evaluation of pooling operations in convolutional architectures for object recognition. *Artificial Neural Networks–ICANN 2010* (2010), 92–101.

[45] Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR)* 34, 1 (2002), 1–47.

[46] Ritambhara Singh, Arshdeep Sekhon, Kamran Kowsari, Jack Lanchantin, Beilun Wang, and Yanjun Qi. 2017. GaKCo: a Fast GApped k-mer string Kernel using COunting. *arXiv preprint arXiv:1704.07468* (2017).

[47] Aixin Sun and Ee-Peng Lim. 2001. Hierarchical text classification and evaluation. In *ICDM*. IEEE, 521–528.

[48] Yichuan Tang. 2013. Deep learning using linear support vector machines. *arXiv preprint arXiv:1306.0239* (2013).

[49] Simon Tong and Daphne Koller. 2001. Support vector machine active learning with applications to text classification. *Journal of machine learning research* 2, Nov (2001), 45–66.

[50] Mehmet Turan, Yasin Almalioglu, Helder Araujo, Ender Konukoglu, and Metin Sitti. 2017. Deep EndoVO: A Recurrent Convolutional Neural Network (RCNN) based Visual Odometry Approach for Endoscopic Capsule Robots. *arXiv preprint arXiv:1708.06822* (2017).

[51] Jason Weston and Chris Watkins. 1998. *Multi-class support vector machines*. Technical Report. Department of Computer Science, Royal Holloway, University of London.

[52] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J Smola, and Eduard H Hovy. 2016. Hierarchical Attention Networks for Document Classification.. In *HLT-NAACL*. 1480–1489.

[53] Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural svms with latent variables. In *ICML*. ACM, 1169–1176.

[54] Wen Zhang, Taketoshi Yoshida, and Xijin Tang. 2008. Text classification based on multi-word with support vector machine. *Knowledge-Based Systems* 21, 8 (2008), 879–886.

[55] Zhi-Hua Zhou and Ji Feng. 2017. Deep forest: Towards an alternative to deep neural networks. *arXiv preprint arXiv:1702.08835* (2017).