Real-Time Internet Traffic Classification using Deep Learning

Vikas Verma

***Abstract*—Over the last few years, deep learning based approaches have gained significant interest in various domains such as visual recognition, speech recognition and natural language processing. However, to the best of our knowledge these deep learning approaches have not been extensively studied for Internet traffic classification task. In this paper, we apply Stacked Denoising Auto-encoders (SDAE), which is a type of deep learning algorithm, to Real-Time Internet traffic classification. We use two operational network data traces for our experiments. Our results show that significant gain in classification accuracy can be achieved by applying deep learning algorithms. We also show that the representations learned at the hidden layers of SDAE can be used with other supervised classification algorithms to improve their performance. Further, we explore the usefulness of these representations in the problem settings with very limited amount of labeled data. Finally, we investigate the robustness of the SDAE based representations by evaluating their performance across data sources. We obtain encouraging results for all of the above mentioned experiments. We hope that this paper will inspire more research for applying deep learning approaches to a wide range of Internet traffic classification and Internet Modeling and Measurement tasks.**

I. Introduction

Ever since the invention of the Internet, it has evolved into a huge and omnipresent infrastructure supporting an increasingly huge market of e-commerce, digital media, social networking, digital advertisement, knowledge discovery and information sharing etc. This success of the Internet has led to the emergence of a wide variety of Internet applications. In addition to traditional applications (e.g. web-browsing, email, file-transfer), new applications such as instant messaging, mobile-advertising and live-video streaming are gaining popularity and momentum.

Along with the development and evolution of Internet applications, there is an urgent need for an efficient Internet traffic classification mechanism. From the Quality of Service perspective, accurate classification of Internet traffic helps in facilitating the instrumentation of application specific strategies. From security perspective, fast identification of malicious traffic helps in security management and control. From network management perspective, traffic classification is a key input for various activities like network planning, capacity provisioning, traffic shaping and workload modeling. Moreover, the application level knowledge of Internet traffic is extremely useful for those who intend to model the long term changes and requirements of Internet.

Vikas Verma is with MOCS, Ericsson Research, Chennai, India (email: vikasverma.iitm@gmail.com).

Accepted in IEEE WoWMoM 2015, Boston, MA, USA.

Real-time or On-line Internet traffic classification seeks to make a classification decision by seeing only first N packets of any Internet traffic-flow. This kind of timely classification before the whole flow is finished is crucial if the network intends to react to the presence of particular classes of traffic by making provisions for dynamic resource allocation, automatic blocking of specific applications etc.

Historically, the traffic classification has been based on the static and standard port numbers in the transport header of packets. This method is based on the assumptions that most of the applications use well known port numbers. However, many application are increasingly using dynamic port numbers or employing the technique of port tunneling to hide behind known port numbers. Consequently, port number based techniques have become less reliable [1]. More sophisticated techniques of Deep Packet Inspection [2], [3], which require complete and costly exploration of payload of packets, are more effective than the packet header based approaches. However, two challenges undermine the effectiveness of this approach. First, applications may use encryption to obfuscate packet payloads or governments can impose privacy constraints limiting the possibilities of lawfully inspecting packet payloads at all. Second, the storage and computational cost to inspect every packet that traverses a link is very high.

In response to above challenges, the research community has developed classification techniques that do not require packet payload inspection. Most of these techniques can be categorized into: (1) host-level communication behaviour-based approach such as BLINC [4]. (2)Machine learning based approach which use various flow-level characteristic, such as size of packets, directions of packets, packet inter-arrival time to classify traffic. In particular, substantial efforts have been invested into machine leaning based traffic-flow classification.

II. Related Work

One of the earliest work [5] that applied Machine Learning in Internet traffic classification uses the Expectation Maximization algorithm. The work of Sander et al. [6] uses AutoClass algorithm for classifying traffic-flows.

Roughan et al. [7] proposed to use the Nearest Neighbours, Linear Discriminate Analysis and Quadratic Discriminant Analysis machine learning algorithms for classifying traffic flows into predetermined Quality of Service traffic classes. Moore and Zuev [8] proposed to apply the supervised Naive Bayes algorithm to classify Internet traffic flows by application. The work was further extended with the application of Bayesian neural network approach in [9].

Erman et al. in [10] proposed a semi-supervised traffic classification approach which combines a unsupervised clustering phase with supervised cluster labeling phase. This approach was motivated by two reasons: Firstly, it is difficult to obtain labeled traffic flows, while supervised learning methods do not generalize well when being trained with few labeled samples. Secondly, traditional supervised methods do not have the ability to detect any previously unseen application.

Park et al. [11] used Genetic algorithms for feature selection and compared the Naive Bayesian classifier with Kernel Estimation, Decision Tree and the Reduced Error Pruning Tree classifier.

Bernaille et al. [12] proposed a technique using k-means algorithm that performed real-time classification of different types of TCP-based applications using the first few packets of the traffic flow. Haffner et al. [13] also proposed real time classification mechanism based on automated construction of application signatures using machine learning techniques. Nguyen et al. [14] proposed an approach for real-time classification that does not require the classifier to capture the start of each traffic flow (as required in [12] and [13]). In this approach, classification can be initiated at any point in time during the life-time of the traffic flow. Sena et al. [15] proposed to use Support vector machine(SVM) for real-time classification of traffic-flows.

All of the above approaches have only shallow- structured architectures, i.e. they consist of at-most one layer of non-linear feature transformations. For example, knn, Naive Bayes, LDA, clustering based approaches in [5], [6], [12], semi-supervised approach in [10] use no feature transformation layer. SVMs use a one or none feature transformation layer based on whether the kernel trick has been used or not.

However, during the past few years, the machine learning community has shown that better generalization performance on complex pattern recognition tasks such as speech recognition, visual recognition and natural language processing can be achieved through *deep architectures*, i.e. by using many layers of non-linear feature transformations. These approaches are popularly known as *deep learning*. In this paper, we apply a deep learning algorithm (Stacked Denoising Autoencoder [16]) to two different operational network data sets for the task of real-time Internet traffic classification and compare its performance with previously studied machine learning based approaches. We also examine whether the higher level representations learned by Stacked Denoising Autoencoder (SDAE) can improve the performance of other supervised algorithms such as SVM. Further, we analyze the performance of higher level representations in the problem settings that have very limited number of labeled data. Finally, we examine the robustness of learned representation by measuring their performance across data sets. Our results show that applying deep learning algorithm or using the higher level representations obtained through deep learning gives better performance than the previously applied methods for real-time traffic classification task.

The rest of the paper is organized as follows: In Section III, we briefly describe the deep learning algorithms. Autoencoders, Denoising Autoencoders and Stacked Denoising Autoencoders are also described briefly in this section. The details of the network traces used for the experiments are given in Section IV. Experiments and results using SDAE for real-time Internet traffic classification task are presented in Section V.

III. Algorithms

A. Deep Learning

Deep learning based approaches aim to apply several levels of nonlinearity to the input to efficiently model the complex relationship between the variables and to achieve better performance on difficult machine learning tasks. Deep neural network is an example of these kind of approaches. Figure 1 illustrates a deep neural network and its parameters. A deep neural network contains an input layer, a output layer and l hidden layers between input and output layer. Given an input $\mathbf{x}$ clamped to input layer, the $(k+1)$-th layer of the network computes its transformation of input $h_{k+1}(\mathbf{x})$ using the transformation $h_k(\mathbf{x})$ of the k-th layer, the set of weights $\mathbf{W_k}$ between k-th layer and $(k+1)$-th layer and the biases ${b_{k+1}}^i$ of various units of $(k+1)$-th layer.

Although, the concept of deep neural networks is considerably old [17], [18] until recently it was not clear how to train such deep networks, since gradient-based optimization starting from random initialization often appears to get stuck in local minima which yields poor solutions. The recent revival of interest in deep neural networks is due to the discovery of new methods [19], [20], [21], [22] that proved successful at effectively learning its parameters. These methods follow a two-step procedure.

1) : Training one layer of deep network at a time using an unsupervised learning in a way such that it preserves the information from input and disentangles factor of variations. The $(k+1)$-th layer is trained after k-th layer has been trained and the representation learned by k-th layer is used as input for the $(k+1)$-th layer. This step is known as *Pre-training*.

2) : Training of entire deep network with respect to the ultimate criterion of interest in a supervised manner. This step is known as *Fine-tuning*

This *greedy layer-wise* procedure has been shown to yield significantly better local minima than random initialization based training of deep networks. Several deep architectures such as Stacked Autoencoders [21], Deep Belief Networks [20], Convolutional Deep Belief networks [23] have been shown to be successfully trained by following the above mentioned two-step process. In next sections, we briefly describe a Basic Autoencoder (which is a commonly used basic building block of the deep neural networks), Denoising Autoencoder and Stacked Denoising Autoencoder respectively.

B. The Basic Autoencoder

An autoencoder takes an input vector $\mathbf{x} \in \mathbf{R}^d$ and maps it to a latent representation $\mathbf{y} \in [0,1]^{d'}$ through a deterministic

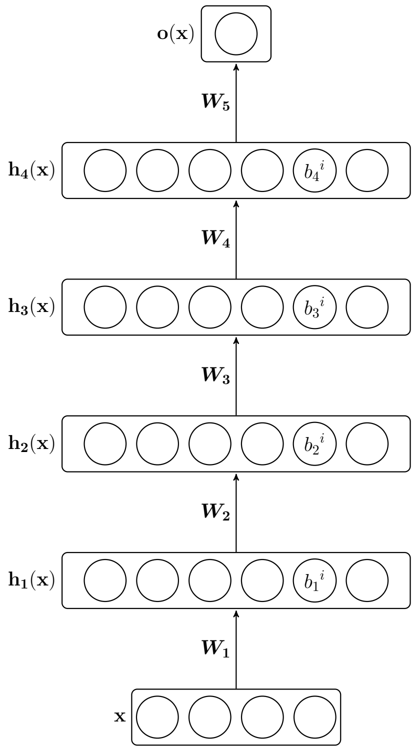
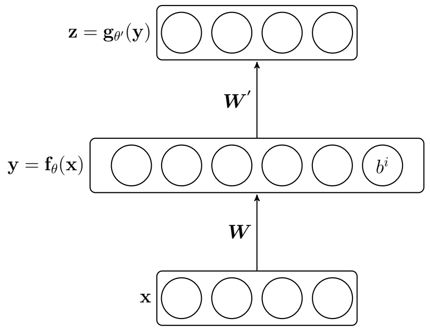


Fig. 1. Illustration of a deep network and its parameters



$\mathbf{z} = \mathbf{g}_{\theta'}(\mathbf{y})$
$\boldsymbol{W}'$
$\mathbf{y} = \mathbf{f}_{\theta}(\mathbf{x})$
b^i
$\boldsymbol{W}$
$\mathbf{x}$

Fig. 2. Illustration of Autoencoder and its parameters

mapping (called as encoder) $\mathbf{y} = f_\theta(\mathbf{x}) = \mathbf{s}(\mathbf{W}\mathbf{x} + \mathbf{b})$. This encoder is parametrized by $\theta = \{\mathbf{W}, \mathbf{b}\}$ where $\mathbf{W}$ is a $d' * d$ weight matrix and $\mathbf{b}$ is a bias vector. The resulting latent representation is mapped back to the reconstructed vector $\mathbf{z} \in [0,1]^d$ via a function $\mathbf{z} = g_{\theta'}(\mathbf{y}) = \mathbf{s}(\mathbf{W}^{'}\mathbf{y} + \mathbf{b}^{'})$. This mapping function is called the decoder. Each training sample $\mathbf{x}^{(\mathbf{i})}$ is thus mapped to a corresponding latent representation $\mathbf{y}^{(\mathbf{i})}$ and reconstructed representation $\mathbf{z}^{(\mathbf{i})}$. The parameters of this model are optimized to minimize average reconstruction error:

$$(\theta, \theta')_{optimized} = \underset{(\theta,\theta')}{\text{argmax}} \frac{1}{n} \sum_{i=1}^{n} L(\mathbf{x}^{(\mathbf{i})}, \mathbf{z}^{(\mathbf{i})}) \qquad (1)$$

where L is can be any loss function such as squared error loss $L(\mathbf{x}, \mathbf{z}) = ||\mathbf{x} - \mathbf{z}||^2$. Figure 2 shows an illustration of this model.

Many variants [24], [25] of basic autoencoder have been proposed to achieve better performances on classification problems. Denoising Autoencoder [16] is one such variant. In next section, we briefly describe Denosing Autoencoder.

C. The Denoising Autoencoder

In addition to minimum average reconstruction criterion, Denoising Autoencoder employs an additional criterion of *robustness to partial corruption of the inputs*. Instead of merely trained to reconstruct the input, the Denoising auto encoder is trained to reconstruct a clean repaired input from a artificially destroyed one. It was shown [16] that this training criteria forces the autoencoder to learn more meaningful hidden representations and capture the structure of the input distribution more effectively. In practice, first the initial input $\mathbf{x}$ is stochastically corrupted into a partially destroyed version $\widetilde{\mathbf{x}}$. Next, as with the basic autoencoder, the corrupted input $\widetilde{\mathbf{x}}$ is mapped to the hidden representation $\mathbf{y} = f_\theta(\widetilde{\mathbf{x}}) = s(\mathbf{W}\widetilde{\mathbf{x}} + \mathbf{b})$ which is used to obtain a reconstructed representation $\mathbf{z} = g_{\theta'}(\mathbf{y}) = \mathbf{s}(\mathbf{W}^{'}\mathbf{y} + \mathbf{b}^{'})$. Again, as with the basic autoencoder, the parameters are trained to minimize the average reconstruction error $L(\mathbf{x}, \mathbf{z})$ with respect to the clean training sample $\mathbf{x}$, i.e. to have $\mathbf{z}$ as close as possible to $\mathbf{x}$. However, the key difference is that since $\mathbf{z}$ is a function of $\widetilde{\mathbf{x}}$ rather than $\mathbf{x}$, minimization of average reconstruction error forces the autoencoder to learn hidden representations which are robust to noise. For corruption of input, the common choices include additive Gaussian noise or masking a random fraction ν of input components to zero known as masking noise.

D. Stacked Denoising Autoencoder

Stacked Denoising Autoencoder consist of multiple layers of Denoising Autoencoders placed over each other. As explained in Section III-A, each layer of Denoising Autoencoder is trained one at a time and representation learned by layer k-th is passed as a input to $(k+1)$th layer. Finally, the parameters of deep network are fine-tuned to meet the desired supervised criterion.

However, it is important to note that the process of corruption is used only during the pre-training phase. The fine-tuning phase receives uncorrupted inputs. Also, when the layer $(k+1)$th layer is trained, it receives uncorrupted output of the previous layer.

IV. Data Set

For our experiments, we have captured network traces from two different operational mobile network. These traces were captured at different locations and at different point of times. We have used our in-house Deep Packet Inspection engine to label the traffic flows in these traces. The details of these traces are listed in Table I.

TABLE I
DETAILS OF NETWORK TRACES

Details	Network Trace	
	Data I	Data II
Duration	2 hours	1 hour
Number of Packets	1286 million	353 million
Number Of Flows	17 million	7 million
Number of Unique End-Users	210437	67513
Number of Unique Functionalities	20	21
Number of Unique Web-Services	35	70

TABLE II
EXAMPLE CLASSES FOR FUNCTIONALITY BASED FLOW CLASSIFICATION AND WEB-SERVICE BASED FLOW CLASSIFICATION

Functionality	Web-Service
Video playback	Youtube
Instant messaging	Google
Web browsing	Twitter
Social networking	Facebook
Email	Yahoo
File sharing	Admob
Weather	AOL
Software update	Andomedia
Advertisement	Pandora
MMS	Flurry

V. EXPERIMENTS AND RESULTS

A. Features used for representing a Flow

Since packet size, packet direction and packet inter arrival time are three widely adopted features for real time traffic classification with superior classification ability [26], [27], we use these features to construct a feature vector representation of a given flow. In particular, a flow is represented by following feature vector:

$$\mathbf{x} = \{s_1, d_1, t_1, ...s_i, d_i, t_i, ...s_n, d_n, t_n\} \tag{2}$$

where s_i is the size of i-thpacket in bytes, d_i is the direction of i-th packet (0 for uplink packets, 1 for downlink packets) and t_i is the inter-arrival time between i-th and $(i+1)$-th packet.

It has been also shown that using 5 to 6 initial packets gives better classification performances for real-time traffic classification [28], [29]. Hence, in our work we have used packet size, packet direction, and packet inter arrival time of only first 6 packets. This gives us a 18 dimensional feature vector representation for each flow.

B. Performance Metric

In this work, we use overall accuracy as the performance metric to compare the performance of different classifiers. It is defined as the ratio of the sum of all True Positives (TP) to the sum of all the True and False Positives (FP) for all the classes:

$$Accuracy = \frac{\sum_{i \in classes} TP_i}{\sum_{i \in classes} (TP_i + FP_i)} \tag{3}$$

Since in our experiments test dataset consist of equal number of examples from each class (more details on training, validation and testing dataset size are given in section V-C), it is suffice to use overall accuracy as a performance metric.

C. Stacked Denoising Autoencoder based Traffic-flow classification

In this section, we present our studies on traffic-flow classification using the Stacked Denoising Autoencoder. We perform the classification at two different levels of granularity. First, at the level of *functionality* provided by the flow. Second, at the level of *web-service* delivered by the flow. Table II presents some of the example classes at each level of granularity.

We selected a random subset of 1.5 million samples from the data set for pre-training of a 3 layer Stacked Denoising Autoencoder (SDAE-3). Note that the pre-training phase is unsupervised, hence we can use both labeled and unlabeled samples in this phase.We have used masking noise to corrupt the inputs during this pre-training phase of SDAE-3. For fine-tuning of SDAE-3, from each class, we randomly sampled training, validation and testing data of size $4000, 1000, 5000$ respectively. Table III and Table IV reports the classification accuracy (averaged over 10 random selections of train, validation and test data) for the functionality classification and web-service classification respectively using SDAE-3. We also compare the performance of SDAE-3 against baseline supervised algorithms such as K-Nearest Neighbours (knn), Linear Discriminant Analysis (LDA) and Support Vector Machine(SVM). Same randomly selected sets of train, validation and test data as that of fine-tuning phase of SDAE-3 were used for all of these methods. The improvement (in %) in classification accuracy using SDAE-3 over the best performing baseline method is given in parenthesis in Table III and Table IV.

For Stacked Denoising Autoencoder, several values of various hyperparameters such as corruption fraction ν , number of units in the hidden layer, number of pre-training epochs, number of fine-tuning epochs, pre-training learning rate, fine-tuning learning rate etc were tried. Likewise, for other baseline supervised algorithms, we have tried several values of various hyperparameters. For each classification problem, the best parameters were selected based on its classification performance on validation data.

TABLE III
CLASSIFICATION ACCURACY (%) OF 3 LAYER STACKED DENOISING AUTOENCODER FOR FUNCTIONALITY CLASSIFICATION TASK

Classifier	Network Trace	
	Data I	Data II
knn	70.12	73.91
LDA	67.26	71.09
SVM	72.82	76.49
SDAE-3	**81.73**(12.23)	**84.46**(10.41)

Our results show that SDAE-3 systematically outperforms

TABLE IV
CLASSIFICATION ACCURACY (%) OF 3 LAYER STACKED DENOISING AUTOENCODER FOR WEB SERVICE CLASSIFICATION TASK

Classifier	Network Trace	
	Data I	Data II
knn	74.17	76.83
LDA	69.96	72.18
SVM	76.24	79.15
SDAE-3	**82.91**(8.04)	**86.78**(9.63)

all the baseline methods. For both of the tasks of functionality classification and web-service classification, significant classification accuracy improvement (8 to 12 %) over the best performing baseline method is achieved by using SDAE. (SVM is best performing baseline method in all the cases).

D. Utilizing the representation learned in unsupervised manner for other supervised algorithms

In this section, we examine whether the input transformations or hidden representations learned at various layers of Stacked Denoising Autoencoder during the unsupervised phase (pre-training) could improve the performance of other learning algorithms such as SVMs. To this end, we fed the hidden representations learned by the SDAE-3 at layer 1, 2 ,3 and concatenation of representation learned at all three layers to a SVM(with RBF kernel). For obtaining the hidden representation, we identified the best performing SDAE-3 (based on their validation performance in the experiments of Section V-C) and used the hidden representations learned by this SDAE prior to the fine-tuning (i.e. after the unsupervised phase of pre-training only). The parameters of SVM and RBF kernel were tuned on the validation set as usual.

Classification accuracy (averaged over 10 random selections of train, validation and test data) for the Functionality classification and Web Service classification are reported in Table V and Table VI respectively. Same sets of train, validation and test data as in Section V-C are used for these experiments.

TABLE V
CLASSIFICATION ACCURACY (%) OF SVM_{rbf} USING THE REPRESENTATIONS LEARNED BY SDAE FOR FUNCTIONALITY CLASSIFICATION TASK

Representation	Network Trace	
	Data I	Data II
RAW	72.82	76.49
SDAE-layer 1	75.34	79.72
SDAE-layer 2	78.92	81.03
SDAE-layer 3	78.89	82.48
SDAE-layer 1+2+3	**80.53**	**83.06**

Our results show that significant performance improvement can be achieved by using hidden representations learned by a SDAE instead of using raw features. We observe that in almost all the cases using the $(k+1)$-th layer representation gives better performance than using the k-th layer representation. This justifies the intuition behind the hierarchical learning of representations in the deep learning methods. We also observe that concatenating the representations learned at all the layers can further improve the classification accuracy.

TABLE VI
CLASSIFICATION ACCURACY (%) OF SVM_{rbf} USING THE REPRESENTATIONS LEARNED BY SDAE FOR WEB-SERVICE CLASSIFICATION TASK

Representation	Network Trace	
	Data I	Data II
RAW	76.24	79.15
SDAE-layer 1	78.13	82.56
SDAE-layer 2	81.82	84.68
SDAE-layer 3	81.23	85.24
SDAE- layer 1+2+3	**82.41**	**86.19**

E. Dealing with very limited number of labeled data

Correctly labeling the Internet traffic flows is expensive, difficult and time consuming process. For each class, often very limited number of correctly labeled samples are available. In the following experiments, we examine the performance of learned representations when the number of labeled data is very less. We use the same hidden representations as obtained through the experiments in Section V-D. Similar to the experiment of Section V-D, we have fed the representation learned at various layers of SDAE-3 to a SVM (with RBF kernel), but for varying number of labeled data. Table VII and Table VIII reports the results (averaged over 10 random selections of train, validation and test data) for Functionality Classification task and Web-Service Classification Task respectively on Data Set I.

TABLE VII
CLASSIFICATION ACCURACY (%) OF SVM_{rbf} FOR VARYING NUMBER OF LABELED DATA USING THE REPRESENTATIONS LEARNED BY SDAE FOR FUNCTIONALITY CLASSIFICATION TASK ON DATA SET I

Representation	Labeled data from each class			
	200	500	1000	2000
RAW	39.72	48.21	57.19	64.36
SDAE-layer 1	66.81	69.24	71.86	72.63
SDAE-layer 2	68.09	71.57	74.08	76.11
SDAE-layer 3	68.41	71.48	74.31	75.87
SDAE-layer 1+2+3	**69.89**	**72.31**	**74.92**	**76.47**

TABLE VIII
CLASSIFICATION ACCURACY (%) OF SVM_{rbf} FOR VARYING NUMBER OF LABELED DATA USING THE REPRESENTATIONS LEARNED BY SDAE FOR WEB-SERVICE CLASSIFICATION TASK ON DATA SET I

Representation	Labeled data from each class			
	200	500	1000	2000
RAW	42.47	51.13	58.10	63.79
SDAE-layer 1	67.81	68.64	73.16	75.05
SDAE-layer 2	68.24	71.01	74.82	75.76
SDAE-layer 3	69.01	71.53	75.17	76.92
SDAE-layer 1+2+3	**70.03**	**72.89**	**76.12**	**78.57**

We observe that in all the cases, using hidden representations obtained through SDAE-3 gives remarkably better performance than using raw features. We also observe a clear pattern that the difference between the performance of raw representation and hidden representation is higher if the number of labeled samples is lesser. For example, for functionality classification task on Data Set I, the difference in classification accuracy achieved using raw representation and hidden representation at 3^{rd}layer of SDAE-3 is 38.69% and 9.51% respectively for 200 and 2000 labeled samples per class. Clearly, it is very much beneficial to utilize the hidden representations learned using unsupervised training of deep architectures in the problems with very scarce labeled samples.

F. Towards the robustness of the learned representations

The characteristics of the real-world networks changes rapidly with respect to location and time. For example, the distribution of inter-arrival time of packets during a busy traffic hour will be different from distribution of inter-arrival time of packets during a non-busy traffic hour. Hence, the classifier system built for one location or at one time might not work expectedly for another location or at another time. Keeping these situations in mind, we examine the robustness of the representations learned using SDAE. In particular, we do the following:

1) : We learn the representations as described in Section V-D using data set I

2) : We train a SVM classifier (with RBF kernel) using these representation and test the performance of this classifier on the test data of data set II

We repeat the process but this time using data set II for training and data set I for testing. Table IX and Table X report the results for functionality classification and web-service classification respectively while using raw representation and SDAE layer-3 representation across data sets.

TABLE IX
CLASSIFICATION ACCURACY $(\%)$ OF SVM_{rbf} FOR FUNCTIONALITY CLASSIFICATION TASK USING RAW REPRESENTATION AND SDAE LAYER-3 REPRESENTATIONS WITH CROSS DATA SET TESTING

Raw Representation		
Training	Testing	
	Data I	Data II
Data I	72.82	69.45
Data II	66.53	76.49

SDAE layer-3 representation		
Training	Testing	
	Data I	Data II
Data I	78.89	80.19
Data II	75.21	82.48

TABLE X
CLASSIFICATION ACCURACY $(\%)$ OF SVM_{rbf} FOR WEB-SERVICE CLASSIFICATION TASK USING RAW REPRESENTATION AND SDAE LAYER-3 REPRESENTATIONS WITH CROSS DATA SET TESTING

Raw Representation		
Training	Testing	
	Data I	Data II
Data I	76.24	72.76
Data II	70.65	79.15

SDAE layer-3 representation		
Training	Testing	
	Data I	Data II
Data I	81.23	83.31
Data II	78.34	85.24

Our experiments show that in the context of across data-set utilization of classification models, classifiaction models build using hidden representation give more robust classification accuracy than classification models build using raw representation. In particular, we observe that for functionality classification task, there is a $\sim$ $(8$ to $10)\%$ relative degradation in classification accuracy while using the raw representation based SVM across data sets, whereas, there is only $\sim$ $(2$ to $4)\%$ relative degradation in classification accuracy while using the SDAE layer-3 representation based SVM. Similarly, for web-service classification task, while using raw representation based SVM across data sets, we observe a $\sim$ $(7$ to $8)\%$ relative degradation in classification accuracy, whereas using the the SDAE layer-3 representation based SVM suffers from only $\sim$ $(2$ to $3)\%$ relative degradation in classification accuracy. Clearly, the representations learned through SDAE are more robust to the location and time dependent patterns in network traces. In other words, these representations are more capable of capturing the useful relationships between the variables and disentangling the unwanted/noisy patterns.

VI. DISCUSSION AND CONCLUSION

In this work, we applied Stacked Denoising Autoencoder to real-time Internet traffic classification task. We performed the classification at dual granularity of functionality and web-service. Our results show that the deep learning algorithm like Stacked Denoising Autoencoder performs significantly better than baseline methods. We also evaluated the usefulness of representations learned at the hidden layers of SDAE by using them as an input to other supervised classification algorithms such as SVM. Further, we examined the advantage of these learned representations in the settings with very limited number of labeled sample. Finally, we evaluated the robustness of the hidden representations by testing the hidden representation based classification models across data-sets. Our results show that by leveraging a large number of unlabeled data, hidden representations achieve remarkably higher classification performance. We hope that our work will inspire more research for applying and developing deep learning methods for Internet traffic classification and measurement tasks.

REFERENCES

[1] T. Karagiannis, A. Broido, M. Faloutsos, and K. claffy, Transport Layer Identification of P2P Traffic, in *IMC04*, Taormina, Italy, October 25-27, 2004.

[2] A. Moore and K. Papagiannaki, Toward the Accurate Identification of Network Applications, in *PAM 2005*, Boston, USA, March 31-April 1,2005.

[3] S. Sen, O. Spatscheck, and D. Wang, Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures, in *WWW 2005*, New York, USA, May 17-22, 2004.

[4] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: Multilevel traffic classification in the dark. In *ACM SIGCOMM*, Philadelphia, USA August 21-26, 2005.

[5] A. McGregor, M. Hall, P. Lorier, and J. Brunskill, Flow clustering using machine learning techniques, in *Proc. Passive and Active Measurement Workshop (PAM2004)*, Antibes Juan-les-Pins, France, April 2004.

[6] S. Zander, T. Nguyen, and G. Armitage, Automated traffic classification and application identification using machine learning, in *IEEE 30th Conference on Local Computer Networks (LCN 2005)*, Sydney, Australia, November 2005.

[7] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, Class-of-service mapping for QoS: A statistical signature-based approach to IP traffic classification, in *Proc. ACM/SIGCOMM Internet Measurement Conference (IMC) 2004*, Taormina, Sicily, Italy, October 2004.

[8] A. Moore and D. Zuev, Internet traffic classification using Bayesian analysis techniques, in *ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS) 2005*, Banff, Alberta, Canada, June 2005.

[9] T. Auld, A. W. Moore, and S. F. Gull, Bayesian neural networks for Internet traffic classification, *IEEE Trans. Neural Networks, no. 1, pp.223239*, January 2007.

[10] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, Semi-supervised network traffic classification,*ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS) Performance Evaluation Review, vol. 35, no. 1, pp. 369370*, 2007.

[11] J. Park, H.-R. Tyan, and K. C.-C.J., GA-Based Internet Traffic Classification Technique for QoS Provisioning, in *Proc. 2006 International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, Pasadena, California, December 2006.

[12] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian,Traffic classification on the fly, *ACM Special Interest Group on Data Communication (SIGCOMM) Computer Communication Review*, vol. 36, no. 2, 2006.

[13] P. Haffner, S. Sen, O. Spatscheck, and D. Wang, ACAS: Automated Construction of Application Signatures, *SIGCOMM 2005 Workshops*,Philadelphia, USA, August 22-26, 2005.

[14] T. Nguyen and G. Armitage, Training on multiple sub-flows to optimise the use of Machine Learning classifiers in real-world IP networks, in *Proc. IEEE 31st Conference on Local Computer Networks*, Tampa,Florida, USA, November 2006.

[15] G. Sena and P. Belzarena, ”Early traffic classification using Support Vector Machines”, in *LANC09* , Pelotas, Brazil, September 24-25, 2009.

[16] P.Vincent, H. Larochelle,I. Lajoie,Y. Bengio, P. Manzagol,Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion, in *J. Mach. Learn. Res.*, vol. 11, 2010

[17] J.L. McClelland, D.E. Rumelhart, and the PDP Research Group. Parallel Distributed Processing:Explorations in the Microstructure of Cognition , in *MIT Press*,volume 2, Cambridge, 1986.

[18] G.E. Hinton. Connectionist learning procedures. in *Artificial Intelligence* , 40:185234, 1989

[19] G.E. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. in *Science* , 313(5786):504507, July 2006.

[20] G.E. Hinton, S. Osindero, and Y.W. Teh. A fast learning algorithm for deep belief nets. in *Neural Computation*, 18:15271554, 2006.

[21] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layerwis e training of deep networks. in *Advances in Neural Information Processing Systems 19 (NIPS06)* , pages 153160. MIT Press, 2007.

[22] M. Ranzato, C.S. Poultney, S. Chopra, and Y. LeCun. Efficient learning of sparse representations with an energy-based model. in *Advances in Neural Information Processing Systems 19 (NIPS06)*, pages 11371144. MIT Press, 2007

[23] H. Lee, R. Grosse, R. Ranganath, A. Y. Ng. Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. in *Proceedings of the 26 th International Conference on Machine Learning* , Montreal, Canada, 2009

[24] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, Yoshua Bengio. Contractive Auto-Encoders: Explicit Invariance During Feature Extraction.in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011

[25] Alireza Makhzani, Brendan Frey. k-Sparse Autoencoders in *International Conference on Learning Representations* Banff, Canada, 2014.

[26] Crotti, Manuel and Dusi, Maurizio and Gringoli, Francesco and Salgarelli, Luca, Traffic classification through simple statistical fingerprinting, *SIGCOMM Computer Communication Review, vol. 37, no. 1, pp. 516*, 2007.

[27] Dainotti, A. and Gargiulo, F. and Kuncheva, L. I. and Pescape, A. and Sansone, C., Identification of Traffic Flows Hiding behind TCP Port 80, in *Proceedings of the 2010 IEEE international conference on Communications, 2010*, Cape Town, South Africa, 2010

[28] Buyu Qu, Zhibin Zhang, Li Guo, Dan Meng, ”On Accuracy of Early Traffic Classification,” in *IEEE Seventh International Conference on Networking, Architecture, and Storage*, Xiamen, China, 2012

[29] Wei Li ,Andrew W. Moore,A Machine Learning Approach for Efficient Traffic Classification, in *Proceedings of the IEEE MASCOTS*,Istanbul, Turkey,2007

[30] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, Andrew Y. Ng, Self-taught learning: transfer learning from unlabeled data in *Proceedings of the 24th international conference on Machine learning*,Pages 759-766 ACM New York, NY, USA, 2007