Regularization and Learning an Ensemble of RNNs by Decorrelating Representations

Mohit Yadav
TCS Research New-Delhi, India
y.mohit@tcs.com

Sakshi Agarwal
IIT Kharagpur
sakshi0594@gmail.com

Abstract

Recurrent Neural Networks (RNNs) and their variants (such as LSTMs and GRUs) have been remarkably successful at machine-learning tasks on diverse kinds of sequential data (e.g. text, time-series, etc.). However, training of RNNs continue to be a challenge due to difficulties stemming from regularization and the highly non-convex optimizations involved. In this paper, we propose to regularize training of RNNs by encouraging higher decorrelation in the hidden representations. The cost function is devised to minimize non-diagonal elements of the correlation matrix computed over the hidden representations of RNNs, along with the usual training accuracy term; thereby penalizing redundancy in the learned model. Furthermore, we propose to utilize the idea of decorrelating representations in learning an ensemble of RNNs, in order to maximize diversity in the resulting models; thus enforcing every individual network of the ensemble to gain abilities that are complementary to the ensemble. Extensive experiments are presented on various datasets with different architectures of RNNs. Results are offered for multiple tasks and show that the proposed methods yield a significant improvement; when compared with the state-of-the-art methods.

Introduction

Recurrent Neural Networks have shown a remarkable progress in learning patterns from sequential data and have been successfully applied to a variety of tasks, including but not limited to, language modelling (Mikolov et al. 2010), machine translation (Sutskever, Vinyals, and Le 2014; Bahdanau, Cho, and Bengio 2014) and dialogue systems (Vinyals and Le 2015). The two most prevalent variants of RNNs are Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) networks and Gated Recurrent Unit (GRU) (Chung et al. 2014) networks.

The key factors behind the resurgence of neural networks are: a) the availability of large scale datasets, resulting in the exploration of (high capacity) deep models that can optimally exploit these datasets, and b) improved optimization techniques to train deep models (LeCun, Bengio, and Hinton 2015). Despite its success, overfiting remains a major challenge while training deep models. Even the availability of large scale datasets does not completely eliminate overfiting from deep models (Zhou et al. 2014). As a result, various regularization methods such as Dropout and Batch Normalization have been proposed and regularization of deep architectures continues to be an actively pursued direction of research (Srivastava et al. 2014). Recently proposed regularization techniques for networks have demonstrated a significant gain in the generalization performance for feed-forward networks, but the performance gains for RNNs are not as impressive (Semeniuta, Severyn, and Barth 2016; Cooijmans et al. 2016). Hence, regularization techniques performing well on feed-forward networks cannot be automatically assumed to work for recurrent networks.

Successfully training a machine learning model often hinges upon maintaining a balance between model expressiveness and availability of training data. Therefore, regularization plays a vital role in training machine learning models, as it provides adaptive control over model expressiveness. Also, availability of large data reduces overfiting, or results in narrowing the gap between training and validation accuracy. It has previously been demonstrated by Cogswell et al. (2015) that the availability of large volumes of training data diminishes the Frobenius norm of the correlation matrix computed over the learned hidden representations. Additionally, it was shown that the decrease in the correlation of hidden units correspond to a decrease in the gap between the training and validation accuracies. Essentially, such a trend is indicative of a strong relationship between overfiting and high correlation across hidden state representations. This relationship begets the question of whether or not such a relationship will be causal, and if so, for which networks (Cogswell et al. 2015). The presence of such a causal relationship can potentially be very useful, as it would allow higher generalization performance; merely by biasing the training procedure to minimize the correlation of the hidden representations. With this motivation, this paper investigates the possibility of biasing the training procedure for RNNs to encourage decorrelation between hidden state representations, to improve generalization performance.

Learning an ensemble of models is yet another classical method to achieve higher generalization performance (Hansen and Salamon 1990; Breiman 1996). A large body of literature suggests that ensembles tend to achieve high performance when there is significant diversity between the models in the ensemble (Perrone and Cooper 1993). In particular, negative correlation learning based ensemble learning methods attempt to lessen the correlation between the individual models, by explicitly penalizing the correlation between the errors made by them (Wang, Chen, and Yao 2010; Liu and Yao 1999). Such a penalty term emphasizes diversity and encourages each individual model to assimilate the competence that is complementary to the remaining models of the ensemble. With similar motivations, this paper makes an attempt to train an ensemble of RNNs by decorrelating the hidden representations in order to encourage diversity in the resultant models. While this may offer some resemblance to the past work, the decorrelation of hidden representations has not been utilized as a regularizer for RNNs; and further to diversify the method to learn an ensemble of RNNs, to the best of our knowledge.

The focus of this paper is to achieve higher generalization performance via an enhanced regularization method for RNNs and to learn an ensemble of RNNs efficiently. In the process, this paper makes the following major contributions:

1. We devise a novel cost function *DeCov RNN Loss* by utilizing the decorrelation among the hidden representations to regularize the training of RNNs.
2. We propose a novel cost function *DeCov Ensemble Loss* based on decorrelation of the hidden representations to enforce diversity while learning an ensemble of RNNs.

The remaining of the paper is organized as follows: we start by detailing out the preliminaries on RNNs prior to describing the proposed cost function *DeCov RNN Loss*. Thereafter, the proposed cost function *DeCov Ensemble Loss* for the learning of an ensemble of RNNs is presented. Experimental details and results on various datasets are presented for multiple recurrent architectures. In the next section, an overview of prior related work is provided. Lastly, conclusions and suggestions for future research are placed.

Recurrent Neural Networks

RNNs are inherently capable of processing information arising from sequences of arbitrary length - due to the presence of recurrent connections shared across time. Given an input sequence of vectors $(x_1, x_2, x_3, ..., x_T)$, RNNs try to learn a running summary of a sequence in their internal hidden representation h_t, at any time t as follows:

$$h_t = \phi(W_h h_{t-1} + U_h x_t) \tag{1}$$

where W_h is a weight matrix for the recurrent connections, U_h is an input-to-hidden weight matrix, and ϕ is usually a sigmoid function or hyperbolic tangent function. RNNs are trained to predict labels using the summary of the entire sequence, as follows:

$$p(y|x_1, x_2, x_3, ..., x_T) = f(h_T) \tag{2}$$

where f is the functional transformation between the hidden representation and the output probability distribution for labels. Training of RNNs is difficult due to the issue of vanishing and exploding gradients (Hochreiter and Schmidhuber 1997). To circumvent this issue, various modifications to recurrent networks are proposed such as GRU networks and LSTM (Hochreiter and Schmidhuber 1997) networks.

The GRU is a recently introduced recurrent network architecture Cho et al. (2014) that benefits from the use of two gates; designed to adaptively reset and update its memory content based on input and previous hidden representation. More precisely, the hidden representation h_t for timestep t can be computed as follows:

$$h_t = (1 - z_t) h_{t-1} + z_t \tilde{h_t} \tag{3}$$

where h_{t-1} and $\tilde{h_t}$ represents the previous and newly generated candidates for the hidden representation. z_t is the update gate which controls the extent of *memory* – how much should it forget from previous states and how much should it update based on the current input? The candidate hidden representations $\tilde{h_t}$ and the update gate z_t are computed as follows:

$$\tilde{h_t} = \texttt{tanh}(W_h h_{t-1} + r_t \odot U_h x_t) \tag{4}$$

$$z_t = \sigma(W_z h_{t-1} + U_z x_t) \tag{5}$$

$$r_t = \sigma(W_r h_{t-1} + U_r x_t) \tag{6}$$

where $\odot$ represents an element-wise multiplication and r_t is a reset gate computed according to Equation 6. Both the update and the reset gates allow to capture the long term dependencies. When the previous hidden representation is more vital, the update gate can block the input; and when the previous hidden representation is irrelevant than the reset gate releases information before availing the fresh information from the input x_t.

LSTM is yet another recurrent network with a gating mechanism (Hochreiter and Schmidhuber 1997). Unlike GRU, LSTM masks its memory content by introducing another representation in the form of a cell state representation c_t. Researchers have proposed many variants of LSTM. In this paper, we follow the one described as follows. While the cell state representation c_t tries to retain the summary of sequence, the other gates, namely, input i_t, forget f_t, and output o_t gates control the influence of the input signal x_t on c_t. More precisely, the computational flow is as follows:

$$i_t = \sigma(W_i h_{t-1} + U_i x_t) \tag{7}$$

$$f_t = \sigma(W_f h_{t-1} + U_f x_t) \tag{8}$$

$$\tilde{c_t} = \texttt{tanh}(W_c h_{t-1} + U_c x_t) \tag{9}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c_t} \tag{10}$$

$$o_t = \sigma(W_o h_{t-1} + U_o x_t) \tag{11}$$

$$h_t = o_t \odot \texttt{tanh}(c_t) \tag{12}$$

where $\tilde{c_t}$ and h_t represents a new candidate for cell state and the hidden representation, respectively. It is important to note that the presence of these gates and memory cell permits LSTM to adaptively write, forget, and reveal its memory content, in accordance with the learning problem. The memory cell enables to capture long term dependencies, as the forget gate can go off and create an uninterrupted pipeline to carry the information for a long time.

DeCov RNN Loss: Regularization of RNNs by Decorrelated Representations

For the current and following sections, let $h_n \in \mathbb{R}^D$ represent the hidden state of a recurrent layer in an RNN, where $n \in \{1, 2, 3, ..., N\}$ indicates an index of a sample in the training batch and D is dimension of the hidden representations. The covariance between a pair of i^{th} and j^{th} dimensions of hidden state representations is defined as follows:

$$C_{i,j} = \frac{1}{N} \sum_{n=1}^{N} (h_n[i] - \mu[i])(h_n[j] - \mu[j]) \qquad (13)$$

where $\mu[i]$ is computed as $\frac{1}{N}\sum_{i=1}^{N} h_n[i]$, which is equal to the mean of i^{th} dimension of hidden representations computed over all samples in the training batch. To decorrelate the dimensions of hidden representations from each other, one can bias the training procedure to minimize $\alpha \times \mathcal{L}_{\texttt{DeCovRnn}}$ in addition to the widely used cross-entropy cost function between the true and the predicted labels. The definition of $\mathcal{L}_{\texttt{DeCovRnn}}$ is provided below in Equation 14. Here, α provides a control over the relative importance of the $\mathcal{L}_{\texttt{DeCovRnn}}$ with respect to the cross-entropy error function.

$$\mathcal{L}_{\texttt{DeCovRnn}} = \frac{1}{D^2} \sum_{i=1}^{D} \sum_{j=1}^{i-1} C_{i,j} \qquad (14)$$

It is important to note that the $C_{i,j}$ for the case of i equal to j is not penalized. Decorrelating $h_n[i]$ and $h_n[j]$ can also be interpreted as a method to prevent co-adaptation between these dimensions, which has been the motivation for Dropout as well (Srivastava et al. 2014). The gradient of $\mathcal{L}_{\texttt{DeCovRnn}}$ with respect to $h_e[i]$, i^{th} dimension of hidden representation, for e^{th} sample as mentioned in Equation 15.

$$\frac{\partial \mathcal{L}_{\texttt{DeCovRnn}}}{\partial h_e[i]} = \frac{1}{N} \Big[\sum_{j \neq i} C_{i,j} \times (h_e[j] - \mu[j]) \Big] \qquad (15)$$

The presence of $C_{i,j}$ in the above mentioned gradient ensures that the i^{th} and the j^{th} dimension of hidden representation do not behave in a similar manner; and it will deter redundant solutions via large gradients. Furthermore, these gradient updates are weighted by the difference of $h_e[j]$ and $\mu[j]$. Such a difference helps in emphasizing the importance of j^{th} feature for example e^{th}. More precisely, if the j^{th} feature is not discriminative for the e^{th} example of a batch then it will be close to $\mu[j]$, hence, the gradient updates for the e^{th} example will be minimal.

One must notice that such a regularization method affects the learning of all layers upto the level where decorrelation is applied. This paper utilizes this property to limit the computation required for $\mathcal{L}_{\texttt{DeCovRnn}}$ only to the top-most recurrent layer of the network. Additionally, it imparts RNNs with an ability to assimilate more distinctive and informative summary of the sequences presented. With the use of this trick, the required number of computations does not amplify significantly. The hidden representations of top-most recurrent layer are manipulated to compute the $\mathcal{L}_{\texttt{DeCovRnn}}$ loss without even touching the other intermediate representations obtained by RNNs. Also, the backward gradient flow is not altered for a major part of the network, i.e., between the top-most recurrent layer and the input layer.

DeCov Ensemble Loss: Learning an Ensemble of RNNs by Decorrelated Representations

Learning an ensemble of classifiers has been a popular technique to obtain gain in the generalization performance. Research has shown that the key factor behind the success of ensemble learning is the diversity within the ensemble (Kuncheva and Whitaker 2003; Musehane et al. 2008). However, measuring diversity is still a non-trivial task; as there is not one precise definition that firmly encapsulates its notion. The intuition though is consistent behind the most of the previously suggested methods; which is to encourage each individual to learn complementary abilities within the ensemble.

With similar motivations, a novel cost function $\mathcal{L}_{\texttt{DeCovEnsm}}$ is proposed and mentioned below in Equation 16. The cost function $\mathcal{L}_{\texttt{DeCovEnsm}}$ utilizes the notion of decorrelated representations to encourage diversity while learning an ensemble of RNNs. The cost function $\mathcal{L}_{\texttt{DeCovEnsm}}$ consists of two terms to estimate the diversity: the first is inter-network diversity, which is between the individual networks of the ensemble, and the latter is intra-network diversity which is for the network; with the associated weights, α and β, respectively. The purpose of weights is to control the relative importance of both the diversity terms; with respect to the cross-entropy error function of the learning cost function.

$$\mathcal{L}_{\texttt{DeCovEnsm}} = \frac{\alpha}{D} \sum_{i=1}^{D} \sum_{j=1}^{i-1} \overline{E}_{i,j} + \frac{\beta}{D^2} \sum_{i=1}^{D} \sum_{j=1}^{i-1} \overline{C}_{i,j} \qquad (16)$$

Keeping consistency with the notations introduced in previous section, $\overline{E}_{i,j}$ measures the decorrelation between the mean of hidden representation of a training batch, obtained by the individual RNNs in the ensemble; as mentioned below in Equation 17. On the other hand, $\overline{C}_{i,j}$ measures the decorrelation between the hidden representation of individual RNNs; as mentioned below in Equation 18.

$$\overline{E}_{i,j} = \frac{1}{M} \sum_{m=1}^{M} (\mu^m[i] - \mu^g[i])(\mu^m[j] - \mu^g[j]) \qquad (17)$$

$$\overline{C}_{i,j} = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} (h_n^m[i] - \mu^m[i])(h_n^m[j] - \mu^m[j]) \quad (18)$$

where $\mu^m[i]$ is computed for m^{th} network and $\mu^g[i]$ is computed as $\frac{1}{M}\sum_{i=1}^{M} \mu^m[i]$, which is global mean of hidden representations. M, N, and $[i]$ represent number of networks in ensemble, number of training samples in a batch, and an operator to get i^{th} dimension of a vector respectively.

One of the reasons that makes it difficult to simultaneously learn an ensemble of networks, is the explosion in the number of model parameters. To mitigate this issue, we sum the hidden representations from all individual networks and apply one final non-linear layer to make the joint predictions; as mentioned below in Equation 19. The advantage with this method is that the model parameters between the top hidden layer and output layer can be shared across the networks with in the ensemble. Sharing parameters can lead to a significant reduction in the model parameters, thereby making it less susceptible to overfiting, specially while dealing with a huge number of target classes (Vincent 2014).

$$p(y|x_1, x_2, x_3, ..., x_T) = \texttt{softmax}(\sum_{m=1}^{M} h^M) \qquad (19)$$

Experiments and Results

To ascertain the effectiveness of the proposed methods, we have conducted experiments on the following tasks:

- Digits classification on sequential MNIST.
- Sentiment classification for movie reviews.
- Ticket classification for enterprise support.
- Ensemble learning for all three above mentioned tasks.

For first three tasks, we considered vanilla RNNs and the other two most prominent variants, namely, LSTMs, and GRUs. To study the dynamics of hidden representations, we analyse the separability achieved at the level of hidden representations, for different classes in sequential MNIST (sMNIST), by injecting the proposed loss on the widely used L2-norm method of regularization. We perform dimensionality reduction using t-SNE in order to visualize the hidden state representations (van der Maaten and Hinton Nov 2008). For ensemble learning, we limit this study to only LSTMs in order to demonstrate the efficacy of proposed method to enforce diversity through hidden representations. Because, LSTMs are found to be performing better in comparison to other variants in our earlier experiments; and also because ensemble learning methods are computationally expensive.

Sequential MNIST Digit Classification

Task & Data. In sMNIST, an instance of a digit (0 to 9) is represented as a sequence of pixels in a lexicographic order, (i.e, left to right, top to bottom). The goal of this task is to predict the digit of the image used to generate the sequence of pixels. The dataset consists of 50K, 10K, and 10K samples for training, validation, and testing. Each image is of size 28×28 pixels resulting into a sequence of 784 values.

Setup. In our sMNIST experiments, we use all the networks using a single layer with 200 hidden units. Parameters of all the networks are initialized using a normal distribution with a standard deviation of 0.1. We have used RMSProp with a batch size of 64, to optimize the cross-entropy error function (Tieleman and Hinton 2012). Learning rate is

	RNN		GRU		LSTM	
Methods	Valid	Test	Valid	Test	Valid	Test
Vanilla	98.21	97.99	98.18	98.04	98.04	98.05
L2 Norm	98.31	97.92	98.28	98.09	98.24	98.15
RDrop	98.42	98.07	98.36	98.18	98.44	98.29
$\mathcal{L}_{\texttt{DeCovRnn}}$	98.25	97.97	98.25	**98.23**	98.32	98.14
$\mathcal{L}_{\texttt{DeCovRnn}}$ + RDrop	98.44	**98.19**	98.29	**98.29**	98.56	**98.31**

Table 1: Percentage validation and test accuracy for different methods on the sMNIST digit classification task.

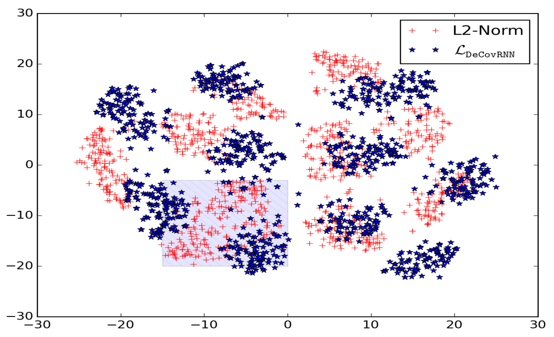

Figure 1: Visualization of 1,000 digits from the sMNIST dataset after performing dimensionality reduction using t-SNE method on the learned hidden representations; obtained by L2-Norm and the proposed method. Unlike, L2-Norm, the proposed method $\mathcal{L}_{\texttt{DeCovRnn}}$ separates better even for the confounded pair of digits; as shown by the shaded region.

fixed to 10^{-3}. Maximum gradient clipping and L2-norm are optimized between $[10^{-1}, 1]$ and $[10^{-4}, 10^{-3}, 10^{-2}]$ respectively. We have compared our results with vanilla architecture, L2-Norm, and Recurrent Dropout (RDrop), a recently proposed and improved version of Dropout for RNNs (Semeniuta, Severyn, and Barth 2016). Best value of probability for Rdrop and coefficient of proposed method is searched in the sets $[0.1, 0.2, 0.5]$ and $[10^{-3}, 10^{-2}, 10^{-1}]$ respectively. Models are trained until 150 epochs and parameters corresponding to maximum validation accuracy are used to test.

Results. Table 1 reports the results for RNN, GRU and LSTM networks. For GRU, the results offered are able to generalize significantly better and makes 50 less erroneous predictions, when compared against RDrop. On combining with RDrop, it boosts up the performance and performs better than the RDrop alone for all three networks.

Visualizing Hidden Representations. To investigate the effect of $\mathcal{L}_{\texttt{DeCovRnn}}$ closely, we compute the hidden representations for 1,000 digits from the test set. The hidden representations are computed using the best model parameters obtained by L2-Norm and the $\mathcal{L}_{\texttt{DeCovRnn}}$, for all choices of hyper-parameters for LSTMs. t-SNE was used to reduce the number of dimensions of hidden representations to two; as depicted in Figure 1. We mitigate the effect of other factors of variations by keeping all other hyper-parameters same for both, with GRUs as the model. Visualization makes it ap-

	RNN		GRU		LSTM	
Methods	Valid	Test	Valid	Test	Valid	Test
Vanilla	59.15	54.03	76.12	74.32	76.56	72.66
L2 Norm	59.93	55.11	77.34	74.32	76.79	73.44
RDrop	59.92	56.45	76.45	73.83	77.57	74.41
$\mathcal{L}_{\texttt{DeCovRnn}}$	59.49	**56.93**	76.67	**75.10**	78.24	74.51
$\mathcal{L}_{\texttt{DeCovRnn}}$ + RDrop	59.82	56.55	76.67	74.88	78.35	**75.20**

Table 2: Percentage validation and test accuracy for different methods on sentiment classification for movie reviews.

	RNN		GRU		LSTM	
Methods	Valid	Test	Valid	Test	Valid	Test
Vanilla	59.38	46.88	64.19	46.88	57.81	53.81
L2 Norm	64.06	51.26	67.19	45.88	76.56	54.39
RDrop	64.36	51.56	70.31	46.98	76.76	54.69
$\mathcal{L}_{\texttt{DeCovRnn}}$	58.0	56.06	68.75	**60.31**	73.44	58.81
$\mathcal{L}_{\texttt{DeCovRnn}}$ + RDrop	59.38	**56.94**	75.0	54.69	73.54	**59.81**

Table 3: Percentage validation and test accuracy for different methods on ticket classification for enterprise support.

parent that the confusion between two digits for L2-Norm (shaded region in Figure 1.) is diminished on injecting the $\mathcal{L}_{\texttt{DeCovRnn}}$ loss function with the coefficient equal to 10^{-3}.

Sentiment Classification for Movie Reviews

Task & Data. Sentiment classification for movie reviews is a widely accepted public benchmark (Pang and Lee 2005). It contains snippets of sentences with positive and negative sentiments expressed by reviewers. The goal is to predict the sentiment expressed in a given sentence. The number of training, validation, and test samples are 8637, 959, and 1066, respectively. The maximum number of words in a sentence and length of vocabulary are 59 and 21426.

Setup. Network consists of two layers: an embedding layer with a dimension of 256, followed by another recurrent layer with 200 hidden units, for all three networks . To reduce the number of parameters to learn, we have utilized pre-trained vectors from Word2Vec (Mikolov et al. 2013). Network weights' initialization, the probability of RMSprop, batch size, learning rate, maximum gradient clipping, coefficient of $\mathcal{L}_{\texttt{DeCovRnn}}$ method are selected similar to sMNIST experiments. L2-Norm is searched in $[10^{-4}, 10^{-3}, 10^{-2}]$. Number of epochs is set to 100; as due to the pre-trained vectors, training has converged faster.

Results. Table 2 reports the results for three recurrent networks. The $\mathcal{L}_{\texttt{DeCovRnn}}$ method outperforms for RNN and GRU; even the combination of $\mathcal{L}_{\texttt{DeCovRnn}}$ method with RDrop. For LSTM, the $\mathcal{L}_{\texttt{DeCovRnn}}$ method combined with Rdrop yields the best performance. Improvement shown in results make it clearly evident that the $\mathcal{L}_{\texttt{DeCovRnn}}$ either supersedes or complements the existing approaches.

Ticket Classification for Enterprise Support

Task & Data. Modern enterprise services are committed to provide good application support to their employees while using the applications under all kinds of network and data requirements from various geographical locations (Zinner et al. 2015). In order to ensure the proper functioning of these applications and to be able to provide suitable countermeasures, ticketing systems are employed to collect complaints and other service or performance related issues. When a large number of tickets are raised in an enterprise, manually categorizing the tickets into various support categories becomes a big workload. We explore the automatic categorization of tickets using machine learning techniques. Our dataset consists of 61252 tickets with these split as Training set of 49002 tickets, Validation set of 6125, and Test set of 6125 tickets. We have to categorize these tickets into 6 categories, namely, configuration/installation issues, web access issues, notes desktop client application issues, password related problems, database related queries, and email related issues. The goal is to predict ticket classes. The maximum length of such tickets is 279 words

Setup. Network consists of two layers: first is an embedding layer with a dimension of 256 and the second is a recurrent layer with 200 hidden units for all three networks. To reduce the number of parameters to learn, we have utilized pre-trained vectors from Word2Vec (Mikolov et al. 2013). The initialization procedure, probability of RMSprop, batch size, learning rate, max grad clip, coefficient to our method are selected similar to sMNIST experiments. L2-Norm is searched within $[10^{-4}, 10^{-3}, 10^{-2}]$. We have set the number of epochs to 100; as due to the pre-trained vectors, training has converged relatively much earlier.

Results. Table 3 reports the results for RNN, GRU and LSTM networks. Either $\mathcal{L}_{\texttt{DeCovRnn}}$ or a combination of $\mathcal{L}_{\texttt{DeCovRnn}}$ with RDrop yields superior performance for all three networks. Especially, for GRU, the $\mathcal{L}_{\texttt{DeCovRnn}}$ method shows an improvement of absolute 13.33%. For RNN and LSTM, a boost up of 5.38% and 5.12% in the performance of our method combined with RDrop is observed when compared to Rdrop alone, respectively.

Experiments on Ensemble Learning

Task & Data. The aim of ensemble learning task is to train multiple base learners and then obtain improvements over them; by predicting jointly. Here, we have fixed base learners to be LSTMs and we utilize all three datasets mentioned in the previous sections, namely, sequential MNIST digit classification, sentiment classification for movie reviews, and ticket classification for enterprise support.

Setup. The architecture for all three tasks is the same as mentioned in their respective sections. Word2Vec pretrained vectors are used for the latter two tasks. Network weights are initialized using zero mean normal and a standard deviation of 0.1. Probability of RMSprop, batch size, learning rate, L2-Norm, maximum gradient clip, α and β of $\mathcal{L}_{\texttt{DeCovEnsm}}$ are $[0.1]$, $[64]$, $[10^{-2}]$, $[10^{-3}]$, $[10]$, $[10^{-2}, 10^{-3}]$, $[10^{-2}, 10^{-3}, 10^{-4}]$, respectively. Maximum number of epochs is 100 and the number of LSTM based learners for ensemble learning are 5.

Baselines. For a comparison to asses the improvements, we first consider best models obtained in our initial experiments with LSTMs. Next, we build a very strong baseline by

Tasks	Single Learner		Avg. Ensm.		$\mathcal{L}_{\texttt{DeCovEnsm}}$	
	Valid	Test	Valid	Test	Valid	Test
sMNIST	98.56	98.31	98.67	98.65	98.76	**98.74**
Sentiment Class.	78.35	75.20	75.67	75.78	79.58	**79.10**
Ticket Class.	73.54	59.81	67.19	**60.93**	65.62	**60.87**

Table 4: Percentage validation and test accuracy for base LSTM models (Single Learner), average prediction ensemble method (Avg. Ensm.), and $\mathcal{L}_{\texttt{DeCovEnsm}}$, on three tasks.

simultaneously training an ensemble of LSTM based models. Prediction is made using an average prediction of all base learners (hence we call it Avg. Ensm.) and network parameters between the top hidden layer and the output layer are shared, as outlined in the proposed $\mathcal{L}_{\texttt{DeCovEnsm}}$ method.

Results. Table 4 depicts the results for three of the above mentioned tasks. On the task of sentiment classification, the proposed method for ensemble learning yields an absolute improvement of 3.9%, when compared with the baseline. Furthermore, for sMNIST, $\mathcal{L}_{\texttt{DeCovEnsm}}$ outperforms the baseline and yields a relative improvement of 25.44% and 6.66% over LSTM and the ensemble baseline respectively.

Related Work

RNNs are among the most prominent models of machine learning due to their ability to remember and to embed information into a fixed length vector (i.e, hidden representation) from a sequence by complicated transformations. Therefore, dynamics of hidden state (or representations) is a fairly studied topic of research in RNNs (Chung et al. 2014; Krueger and Memisevic 2015). In principle, RNNs are Turing-complete and are capable of simulating arbitrary procedures. However, what seems plausible in theory is not always easy to apply successfully. This paper performs analysis and evaluates its learning abilities (via generalization).

In the recent past, a notable interest in the regularization of RNNs is witnessed. As a result, a large number of methods are proposed to tackle this problem (Cooijmans et al. 2016; Semeniuta, Severyn, and Barth 2016). Previous works on regularization of RNNs can be broadly divided into two categories, namely, normalizing representations and network sampling. The methods based on first category of normalizing representations aim to control the distributions of activations in order to mitigate the issue of the internal covariance shift. The internal covariance shift adapts parameters of all layers above the layer at which the input distribution has changed and hence makes it difficult to learn (Cooijmans et al. 2016; Jimmy Lei Ba 2016). The major focus of these methods is on speeding up the training by stabilizing the dynamics of hidden state representations. Network sampling based methods learn by dropping units stochastically, along with their network connections (Srivastava et al. 2014; Semeniuta, Severyn, and Barth 2016; Krueger et al. 2016). This idea was originally proposed in Dropout which has become the de-facto method of regularization for feed-forward deep neural networks. However, Dropout does not result in significant benefits on naive application to RNNs. As dropping units between hidden to hidden connections lead to the problem of memory loss, this issue is latter circumvented by RDrop (Semeniuta, Severyn, and Barth 2016). Due to these reasons, we chose RDrop as a suitable baseline method along with the classical L2-Norm method of regularization.

Reducing redundancy to learn efficient representations has a long history of more than five decades (Barlow 1961). Schmidhuber (1991) proposed to lower redundancy by reducing the predictability of one unit given all of the other units in a layer. Recently, Cogswell et al. (2015) have advocated decreasing redundancy by explicitly minimizing a cost function similar to the proposed method. However, their study was limited to feed-forward networks and images.

Earlier work on ensemble learning such as bagging predictors suggest to generate multiple copies of same predictor; by learning on a sequence of different sets from the same underlying distribution (Breiman 1996). These methods aggregate by averaging out the predictions of multiple versions. The success of these methods is also critically dependent on the perturbation of the learning set so that it induces diversification of the resultant learners (Sollich and Krogh 1995). Unlike the proposed method, these methods do not consider learning simultaneously in an ensemble.

The idea of using neural networks in an ensemble has been proposed previously in Hansen and Salamon; Perrone and Cooper (1990; 1993). The simultaneous training of negatively correlated networks in an ensemble was later introduced by Liu and Yao (1999). Similar to this paper, these methods also encourage different individual networks in an ensemble to learn different parts or aspects of training data so that the ensemble can learn the complete data better. However, these pioneering works were most limited to feed-forward networks, without many of the modern design choices (e.g. Dropout, maximum gradient clipping, etc.).

Conclusion and Future Research

Looking at the widespread applications of RNNs and their inherent tendency to overfit while training, this paper highlights the need for better regularization methods and introduces an effective regularization method for RNNs. This paper suggests that the use of decorrelated hidden representations will impart RNNs with an ability to encapsulate more informative summary of the sequences. The proposed $\mathcal{L}_{\texttt{DeCovRnn}}$ method or a combination of the $\mathcal{L}_{\texttt{DeCovRnn}}$ with RDrop yields significant improvement in generalization performance. Secondly, this paper proposes a novel cost $\mathcal{L}_{\texttt{DeCovEnsm}}$ function to train an ensemble of RNNs; with a strong emphasis on diversifying the abilities of base learners in the ensemble. Extensive experiments are presented to empirically evaluate the efficacy of the proposed methods. While the method looks conceptually similar to the classical learning methods of ensemble learning by negative correlation (Wang, Chen, and Yao 2010); this paper tries to encourage diversity in the intermediate hidden representations, not using the prediction errors made by the base learners.

In future, it will be an interesting direction to investigate the possibility of incorporating supervised information in the proposed methods of regularization and also in the learning of an ensemble. Furthermore, we plan to explore RNNs with higher depth and more volume of text data; to see the impact of decorrelating representations on separability for the tasks requiring information processing from sequences.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Barlow, H. B. 1961. Possible principles underlying the transformations of sensory messages. *MIT press* 217–234.

Breiman, L. 1996. Bagging predictors. *Mach. Learn.* 24(2):123–140.

Cho, K.; van Merrienboer, B.; Gülçehre, Ç.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

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

Cogswell, M.; Ahmed, F.; Girshick, R. B.; Zitnick, L.; and Batra, D. 2015. Reducing overfitting in deep networks by decorrelating representations. *arXiv preprint arXiv: 1511.06068*.

Cooijmans, T.; Ballas, N.; Laurent, C.; and Courville, A. C. 2016. Recurrent batch normalization. *arXiv preprint arXiv:1603.09025* abs/1603.09025.

Hansen, L. K., and Salamon, P. 1990. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence* 12:993–1001.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Comput.* 9(8):1735–1780.

Jimmy Lei Ba, Jamie Ryan Kiros, G. E. H. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Krueger, D., and Memisevic, R. 2015. Regularizing rnns by stabilizing activations. *arXiv preprint arXiv:1511.08400*.

Krueger, D.; Maharaj, T.; Kramár, J.; Pezeshki, M.; Ballas, N.; Ke, N. R.; Goyal, A.; Bengio, Y.; Larochelle, H.; Courville, A. C.; and Pal, C. 2016. Zoneout: Regularizing rnns by randomly preserving hidden activations. *arXiv preprint arXiv:1606.01305*.

Kuncheva, L. I., and Whitaker, C. J. 2003. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning* 51(2):181–207.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *Nature* 521(7553):436–444.

Liu, Y., and Yao, X. 1999. Simultaneous training of negatively correlated neural networks in an ensemble. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 29(6):716–725.

Mikolov, T.; Karafiát, M.; Burget, L.; Cernocký, J.; and Khudanpur, S. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, 1045–1048.

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

Musehane, R.; Netshiongolwe, F.; Nelwamondo, F. V.; Masisi, L. M.; and Marwala, T. 2008. Relationship between diversity and perfomance of multiple classifiers for decision support. *arXiv preprint arXiv:0810.3865*.

Pang, B., and Lee, L. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, 115–124. Stroudsburg, PA, USA: Association for Computational Linguistics.

Perrone, M. P., and Cooper, L. N. 1993. When networks disagree: Ensemble methods for hybrid neural networks. 126–142. Chapman and Hall.

Schmidhuber, J. 1991. Learning factorial codes by predictability minimization. Technical Report CU-CS-565-91, Dept. of Comp. Sci., University of Colorado at Boulder.

Semeniuta, S.; Severyn, A.; and Barth, E. 2016. Recurrent dropout without memory loss. *arXiv preprint arXiv:1603.05118*.

Sollich, P., and Krogh, A. 1995. Learning with ensembles: How overfitting can be useful. 190–196.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, 3104–3112.

Tieleman, T., and Hinton, G. 2012. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

van der Maaten, L., and Hinton, G. Nov 2008. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research* 9: 2579-2605.

Vincent, P. 2014. Efficient exact gradient update for training deep networks with very large sparse targets. *arXiv preprint arXiv:1412.7091*.

Vinyals, O., and Le, Q. V. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Wang, S.; Chen, H.; and Yao, X. 2010. Negative correlation learning for classification ensembles. In *The 2010 International Joint Conference on Neural Networks*, 1–8. IEEE.

Zhou, B.; Lapedriza, A.; Xiao, J.; Torralba, A.; and Oliva, A. 2014. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems*, 487–495.

Zinner, T.; Lemmerich, F.; Schwarzmann, S.; Hirth, M.; Karg, P.; and Hotho, A. 2015. *Text Categorization for Deriving the Application Quality in Enterprises Using Ticketing Systems*. Cham: Springer International Publishing. 325–336.