Adversarial Dropout for Recurrent Neural Networks

Sungrae Park,1* Kyungwoo Song,2 Mingi Ji,2 Wonsung Lee,3* Il-Chul Moon2

1Clova AI Research, NAVER Corp., Korea
2Industrial & Systems Engineering, KAIST, Korea
3AI Center, SK Telecom, Korea

Abstract
Successful application processing sequential data, such as text and speech, requires an improved generalization performance of recurrent neural networks (RNNs). Dropout techniques for RNNs were introduced to respond to these demands, but we conjecture that the dropout on RNNs could have been improved by adopting the adversarial concept. This paper investigates ways to improve the dropout for RNNs by utilizing intentionally generated dropout masks. Specifically, the guided dropout used in this research is called as adversarial dropout, which adversarially disconnects neurons that are dominantly used to predict correct targets over time. Our analysis showed that our regularizer, which consists of a gap between the original and the reconfigured RNNs, was the upper bound of the gap between the training and the inference phases of the random dropout. We demonstrated that minimizing our regularizer improved the effectiveness of the dropout for RNNs on sequential MNIST tasks, semi-supervised text classification tasks, and language modeling tasks.

Introduction
Many effective regularization methods have been introduced to address the issue of large-scale neural networks predisposed to overfitting. Among several regularization techniques (Bishop 1995; Ioffe and Szegedy 2015; Salimans and Kingma 2016), dropout (Srivastava et al. 2014; Kingma, Salimans, and Welling 2015) has become a common methodology because of its simplicity and effectiveness. The dropout randomly disconnects neural units during training to prevent the feature co-adaptation. Srivastava et al. (Srivastava et al. 2014) interpreted the dropout as an extreme form of a model ensemble by sharing the extensive parameters of a neural network. However, the naive application of the dropout to hidden states of recurrent neural networks (RNNs) failed to prove performance gains (Zaremba, Sutskever, and Vinyals 2015) because it interferes with abstracting long-term information. This issue is caused by the structural differences between feed-forward neural networks (FFNNs) and RNNs.

Recent works have investigated the applications of the dropout on the recurrent connections of RNNs. Gal and Ghahramani (Gal and Ghahramani 2016) suggested the application of the dropout on the same neural units through time steps, as opposed to dropping different neural units at each time step. Their approach is similar to the L2 regularization on the weight parameters in the recurrent connection. After this suggestion, Semeniuta et al. (Semeniuta, Severyn, and Barth 2016) and Moon et al. (Moon et al. 2015) explored the dropout applications within long-short term memory (LSTM) cells by regulating the gating mechanism. Recently, Merity et al. (Merity, Keskar, and Socher 2018) proposed the use of DropConnect (Wan et al. 2013) on the recurrent hidden-to-hidden weight matrices. This approach also allows the recurrent units to share the same dropout mask. These dropout techniques for RNNs have shown that recurrent units should have the same transition metric to process the sequential information. These results are used in extending the dropout techniques on RNNs.

Recently, dropout-based ensemble (DE) regularization (Bachman, Alsharif, and Precup 2015; Ma et al. 2017; Laine and Aila 2017; Tarvainen and Valpola 2017) has been established to improve the dropout techniques. DE regularization conceptually consists of two phases: the generation of the dropout masks and the comparison of the original and the perturbed networks by the dropout masks. Bachman et al. (Bachman, Alsharif, and Precup 2015) and Ma et al. (Ma et al. 2017) tried to minimize the distance between the output distributions of the original network and its randomly perturbed network. Laine and Aila (Laine and Aila 2017) suggested self-ensembling models, or the Π model, containing a distance term between two randomly perturbed networks. Park et al. (Park et al. 2018) proposed adversarial dropout that intentionally deactivates neurons that are dominantly used to predict the correct target. By utilizing the adversarial dropout, the regularization consists of the distance between the original and the adversarially perturbed networks. Additionally, they proved further improvements for supervised and semi-supervised image classification tasks. These DE regularizers are fundamentally developed for all types of neural networks, but these DE regularizations are not compatible to RNNs because of the sequential information abstraction.

To investigate the effectiveness of the DE regularization on RNNs, Zolna et al. (Zolna et al. 2018) proposed fraternal dropout (FD), an RNN version of self-ensembling (Laine and Aila 2017). Specifically, two randomly perturbed RNNs by DropConnect are simultaneously supervised from the true labels while maintaining the distance between their predic-
We denoted simple RNN models for brevity of notation because recurrent neurons are mainly used to predict target sequences in using a time-invariant dropout mask. According to our experiments, adversarial dropout on an RNN (Bachman, Alsharif, and Precup 2015; Ma et al. 2017), tries to reduce the network loss as well as the expected difference between the outputs with a deterministic network. However, the approximation causes the gap between the training and the inference phase of the dropout techniques.

Expectation-linearization (EL) regularization (Ma et al. 2017), which is similar to a pseudo-ensemble agreement (Bachman, Alsharif, and Precup 2015), tries to reduce the gap by adding a penalty to the objective function. Let $\epsilon$ be the dropout mask applied to the recurrent connection of RNNs, and $p_t(x, \epsilon; \theta)$ be the prediction of the model for the input sequence $x$ at time $t$. The EL regularizer can be represented as a generalized form as shown below:

$$
\mathcal{R}^{\text{EL}}(x; \theta) := E_{\epsilon} \left[ \sum_{t=1}^{T} \lambda_t D[p_t(x, \epsilon; \theta) || p_t(x, \epsilon; \theta)] \right]
$$

where $D[:\|:]$ indicates a non-negative function that represents the distance between two output vectors such as cross-entropy, $\epsilon$ is a Bernoulli random vector with the probability $p$, and $\lambda_t$ is a hyperparameter controlling the intensity of the loss at time $t$. The goal of the EL regularizer is to minimize network loss as well as the expected difference between the prediction from the random dropout mask and the prediction from the expected dropout mask, which is fully connected. However, the EL regularizer requires a Monte Carlo (MC) approximation because calculating the penalty is still intractable due to the expectation.

FD regularization (Zolna et al. 2018), which is an RNN version of II models (Laine and Aila 2017), indirectly reduces the inference gap by minimizing the variance of output distributions caused by random dropout masks. Its regularizer consists of the distance between two outputs perturbed by two sampled dropout masks as shown in the following:

$$
\mathcal{R}^{\text{FD}}(x; \theta) := E_{\epsilon_1, \epsilon_2} \left[ \sum_{t=1}^{T} \lambda_t D[p_t(x, \epsilon_1; \theta) || p_t(x, \epsilon_2; \theta)] \right],
$$

where $\epsilon_1$ and $\epsilon_2$ are Bernoulli random vectors with the probability $p$. Zolna et al. (Zolna et al. 2018) proved that the FD regularizer is related to the lower bound of the EL regularizer.
Adversarial dropout for recurrent neural networks

Adversarial dropout (Park et al. 2018) is a novel DE regularizer to reduce the distance between two output distributions of the original network and an intentionally perturbed network. Specifically, the adversarial dropout generate a perturbed network by disconnecting neural units that are dominantly used to predict correct targets. The perturbed network reconfigured by the adversarial dropout might not classify the correct target even though few neurons are deactivated from the full network. Therefore, its learning stimulates the useless or the incorrectly learned neurons to better contribute to more accurate predictions. The following is an RNN version of the adversarial dropout regularizer:

\[
R_{\text{AdD}}(x, e^0; \theta) := \sum_{t=1}^{T} \lambda_t D[p_t(x, e^0; \theta)||p_t(x, e^{\text{adv}}; \theta)]
\]  

(4)

where \( e^{\text{adv}} := \arg \max_{\epsilon, \|\epsilon - e^0\|_2 \leq \delta} \sum_{t=1}^{T} \lambda_t D[p_t(x, \epsilon; \theta)||p_t(x, e; \theta)] \).  

(5)

where \( \delta \) is the hyperparameter controlling the intensity of the noise. In this equation, \( e^0 \) is the base dropout mask, which represents a target network that supervises an adversarially dropped network. For a example, \( e^0 \) can be set as \( 1 \) vector that indicates the original network without any dropped neurons. At each training step, we identified the worst case dropout condition, \( e^{\text{adv}} \), against the current model, \( p_t(x, e^0; \theta) \), and trained the model to be robust to such dropout perturbations by minimizing the regularization term.

We note that the regularization term is equivalent to three statistical relationships between two output distributions by a dropout mask, \( e^0 \), and its adversarial dropout mask, \( e^{\text{adv}} \), as shown below (proof in the appendix).

**Remark 1.** Let \( \epsilon \) be an i.i.d. dropout mask; \( p_{t,i}(\epsilon) \) be a \( i \)-th element of \( p_t(x, e; \theta) \in \mathbb{R}^M \) where \( M \) is the size of the output dimension; \( R_{\text{AdD}}(x, e; \theta) \) be the distance term at time \( t \) in Eq.4; and \( e^{\text{adv}} \) be an adversarial dropout mask by setting the base dropout mask as \( e \) when the distance metric is L2 norm. Then,

\[
E_{\epsilon}[R_{\text{AdD}}^t(x, e; \theta)] = \sum_i V_{\epsilon}[p_{t,i}(\epsilon)]^{(1)} + \sum_i V_{\epsilon}[p_{t,i}(e^{\text{adv}})]^{(2)} - 2 \sum_i \text{Cov}_{\epsilon}[p_{t,i}(\epsilon), p_{t,i}(\epsilon^{\text{adv}})]^{(3)} + \sum_i \left[ E_{\epsilon}[p_{t,i}(\epsilon)] - E_{\epsilon}[p_{t,i}(e^{\text{adv}})] \right]^2,
\]

(6)

where \( V_{\epsilon} \) is the variance and \( \text{Cov}_{\epsilon} \) is the covariance. We note that \( e^{\text{adv}} \) depends on the base dropout mask, \( e \), so its output, \( p_t(x, e^{\text{adv}}; \theta) \), contains randomness derived from \( e \). Minimizing the regularization in Eq.5 pursues (1) invariant outputs over random dropout masks and adversarial dropout masks, (2) positive relationship between the two outputs, and (3) minimized distance between the means of the two outputs. The first terms (1) are consistent with the goal of the FD regularization that minimizes the variance of the output distributions over dropout masks (Zolna et al. 2018). The second term (2) indicates that outputs move in the same direction even though dominantly used neurons for the correct predictions are deactivated in the adversarially perturbed network. If the output of sub-network A is higher than the output of sub-network B, the output of adversarial sub-network from A becomes higher than the one from B. This means that a paired gap between a sub-network and its adversarial sub-network becomes similar to the other paired gaps caused by dropping dominantly used neurons. This interpretation provides the connection between one paired case to the others. The third term (3) minimizes the mean of the gaps. This might increase training error of the base network because the adversarially perturbed networks have poor performance. On the other hand, it can improve generalization performance by preventing the base network from overfitting to training data. Figure 2 shows a graphical description of the three statistical relationships.

When comparing our regularizer with the other EL regularizer, our regularization is the upper bound of the EL regularization when \( e^0 = E_{\epsilon}[\epsilon] \) without the MC approxi-
We note that our regularization is also positioned in the upper bound of the model (Park et al. 2018).

The IM can be calculated as follows:

\[ \tilde{h}_t(\tilde{e}) \text{ by } \tilde{\epsilon} \text{ they indicated the influences of the recurrent neurons.} \]

\[ \text{the discrete domain of } \epsilon \text{ invariant dropout mask. To get the influences, we first relax the dropout mask because the recurrent neurons have a time-varying dropout of } \epsilon \text{ term in Eq.4 with respect to } \epsilon \text{. In order to identify } \epsilon \text{ neural networks is intractable for modern neural networks because of non-linear activations of the network and the discrete domain of } \epsilon \text{. This challenge led us to approximate the worst case of } \epsilon \text{. In our experiment, we introduced a flip function, which changes elements of the dropout mask, } \epsilon^* \text{, in the descending order of } \epsilon \text{. This approach is simple but can be improved because } \epsilon \text{ values are unstable when an element is flipped.}

Finding adversarial dropout masks for recurrent neural networks

In order to identify \( \epsilon^{\text{adv}} \), we needed to minimize the distance term in Eq.4 with respect to \( \epsilon \). However, exact minimization is intractable for modern neural networks because of non-linear activations of the network and the discrete domain of \( \epsilon \). This challenge led us to approximate the worst case of the dropout masks by applying the following two steps: (1) identifying the influences of the recurrent neurons and (2) applying a greedy algorithm based on the influence scores (Park et al. 2018).

The influences of the recurrent neurons should be time-invariant because the recurrent neurons are repeatedly applied over every time step. Fortunately, we can derive the influences by using a gradient of the distance term with respect to the dropout mask because the recurrent neurons have a time-varying dropout mask. To get the influences, we first relax the discrete domain of \( \epsilon \) to \( \{0, 1\}^D \) to the continuous domain, \( \epsilon \in [0, 1]^D \), then we calculate the gradients with respect to \( \epsilon \). We named the gradient values as influence map (IM) because they indicated the influences of the recurrent neurons.

Let \( h_t(\tilde{e}) \) and \( \hat{h}_t(\hat{e}) \) be a hidden state and a dropped hidden state by \( \epsilon \) at time \( t \), and \( D_t(\epsilon) = D[p_t(x, \epsilon^0; \theta)|p_t(x, \hat{\epsilon}; \theta)] \).

The IM can be calculated as follows:

\[ \text{IM}_t(\epsilon^*) := \frac{\partial \sum_{t=1}^{T} \lambda_t D_t(\epsilon^*)}{\partial \epsilon_t} \bigg|_{\epsilon = \epsilon^*} \]

\[ = \sum_{t=1}^{T} \lambda_t \frac{\partial D_t(\epsilon^*)}{\partial h_t(\epsilon^*)} \left\{ \sum_{u=1}^{t-1} \frac{\partial h_u(\epsilon^*)}{\partial h_{u, i}(\epsilon^*)} h_{u, i}(\epsilon^*) \right\}, \]  

where \( \epsilon^* \) is the initial dropout mask of \( \hat{e} \). In the equation, \( h_{u, i}(\epsilon^*) \) and \( \hat{h}_{u, i}(\hat{e}) \) are the \( i \)-th components of \( h_t(\hat{e}) \) and \( \hat{h}_t(\hat{e}) \) respectively. The equation on the right shows an alternative view of the IM without any Jacobian matrices with respect to \( \epsilon \). The equation contains the gradient propagation, \( \frac{\partial h_u(\epsilon^*)}{\partial h_{u, i}(\epsilon^*)} \), which is calculated by the backpropagation of the recurrent directions from time \( u \) to time \( t \). This alternative equation shows that the IM provide the degree of the influence of the recurrent units over time. Additionally, the IM depends on the initial values of the dropout mask, \( \epsilon^* \). It is important to note that the initial dropout mask, \( \epsilon^* \), should be different from \( \epsilon^0 \) because the same conditions of dropout mask cause the zero values of the gradient if there is no another stochastic process in the model. In this paper, we initialized \( \epsilon^* \) by randomly flipping a element of the base dropout mask \( \epsilon^0 \) to set a initial dropout mask \( \epsilon^* \). Calculated IM values indicate the relations between the recurrent neurons and the distance term.

After calculating the IM, the adversarial dropout mask, \( \epsilon^{\text{adv}} \), should be identified under the constraint, \( \|\epsilon^{\text{adv}} - \epsilon^0\|_2 \leq \delta \). For the worst case dropout conditions, the neuron with a positive IM value should be activated and the neuron with a negative IM value should be deactivated. Park et al. (Park et al. 2018) proposed a greedy algorithm to find the worst case of the dropout mask utilizing the IM. The algorithm estimates IM values once and iteratively flips a dropout element shown to be highly effective to the distance term in view of the IM. The adversarial dropout with the greedy algorithm is shown below:

\[ \epsilon^{\text{adv}} = \text{Flip}_p\epsilon^* - \epsilon^0\|_2 \leq \delta \left( \epsilon^*, \text{IM}(\epsilon^*) \right), \]

where we introduce a flip function, which changes elements of the dropout mask, \( \epsilon^* \), in the descending order of \( (1 - 2\epsilon^*) \odot \text{IM}(\epsilon^*) \) until the constraint \( \|\epsilon^* - \epsilon^0\|_2 \leq \delta \) is satisfied (detail algorithm in appendix). In the flip function, \( (1 - 2\epsilon^*) \odot \text{IM}(\epsilon^*) \) indicates the influence of flipping a dropout element because the relation estimated by the IM depends on the state of the current dropout mask, \( \epsilon^* \). This approach is simple but can be improved because IM values are unstable when an element is flipped.

We constructed a straightforward way to extend the greedy algorithm. Specifically, we modified the greedy algorithm to update IM values multiple times, as shown here:

\[ \epsilon^{\text{adv}, (0)} = \epsilon^*, \]

\[ \epsilon^{\text{adv}, (k+1)} = \text{Flip}_p\epsilon^{\text{adv}, (k)} - \epsilon^0\|_2 \leq \frac{k+1}{K} \delta \left( \epsilon^{\text{adv}, (k)}, \text{IM}(\epsilon^{\text{adv}, (k)}) \right), \]

where \( K \) is the maximum number of the iteration. For a better approximation of the worst-case dropout mask, the IM values are updated using the current dropout mask and the bound of the constraint is increased through the iterations. There is a trade-off between the better approximation and the computational cost. The large number of the iteration, \( K \), leads to a better approximation, but it also increases the computational cost proportionately. In our experiment, we tested cases of \( K=1 \) and \( K=2 \) to investigate performance improvement obtained by sacrificing the computational cost.
Table 1: Test error rates of supervised learning on sMNIST and pMNIST. Each setting is repeated ten times.

<table>
<thead>
<tr>
<th>Model</th>
<th>sMNIST</th>
<th>pMNIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unregularized</td>
<td>1.018 (±0.165)</td>
<td>9.924 (±0.307)</td>
</tr>
<tr>
<td>VD (2016)</td>
<td>0.721 (±0.111)</td>
<td>5.218 (±0.132)</td>
</tr>
<tr>
<td>FD (2018)</td>
<td>0.720 (±0.061)</td>
<td>5.121 (±0.121)</td>
</tr>
<tr>
<td>AD (K=1)</td>
<td>0.705 (±0.047)</td>
<td>5.046 (±0.067)</td>
</tr>
<tr>
<td>AD (K=2)</td>
<td><strong>0.644 (±0.046)</strong></td>
<td><strong>5.030 (±0.065)</strong></td>
</tr>
</tbody>
</table>

Figure 3: Test accuracies of the perturbed RNNs by random dropout and adversarial dropout masks.

Experiments

Sequential MNIST

Sequential MNIST tasks, also known as pixel-by-pixel MNIST, process each image one pixel at a time and predicts the label of the image. The tasks can be categorized by the order of the pixels in a sequence: sMNIST in scanline order and pMNIST in a fixed random order. Because the size of an MNIST image is 28 × 28, the length of the sequence becomes 784.

Our baseline model included a single LSTM layer of 100 units with a softmax classifier to produce a prediction from the final hidden state. For the settings for the dropout, we set the dropout probability as 0.1 for the baseline models. In the case of the adversarial dropout, we adapted $\epsilon^0 = E_{\epsilon}[\epsilon]$, which indicates the expectation of the dropout mask, and $\delta = 0.03$, which represents the maximum changes from the base dropout mask as 3%. These hyperparameters of the baseline models as well as our models were retrieved in the validation phase. All models were trained with the same optimizer (detail settings in appendix).

Table 1 shows the test performances of the dropout-based regularizations. When applying variational dropout on the recurrent connections, the performance was improved from the unregularized LSTM. By adding regularization terms, the performance on the test dataset was improved in the following order: FD, and adversarial dropout (K=1, 2).

In order to investigate how adversarial dropout training affects the accuracy distribution of the dropped subnetworks, we sampled random dropout masks and adversarial dropout masks respectively 500 times and tested the performance of the subnetwork reconfigured by the sampled dropout masks. Figure 3 (a) shows the histogram of the results. In this analysis, we applied the dropout probability, $p=0.03$, and the hyperparameters, $\delta=0.03$ and $K=2$, to the adversarial dropout. In the case of the variational dropout, we can see that the test performances of adversarially dropped networks were worse than the test performance of randomly dropped networks. By adding the regularization term of the FD, the performances of adversarially dropped networks were improved and caused the performance distribution of randomly dropped networks to move the right and its variance to reduce. That is, regularization of the FD indirectly improved the performance of the subnetwork in its worst case scenario. On the other hand, adversarial dropout directly controlled the sub-network in the worst case scenario. As a result, the performances of the base subnetwork and the adversarially perturbed networks were improved more than the performances in the case of FD regularization.

We additionally investigated the adversarial dropout condition changes throughout the training phase. We calculated $E_{\text{test}}[\epsilon^{\text{adv}}]$ on every training epoch. Figure 3 (b) shows the visualized adversarial dropout masks. As can be seen, the adversarial dropout condition is equally spread over the test dataset in the case of adversarial dropout training, whereas the other cases show that a certain number of elements are selected more frequently for the adversarial dropout condition. These results show that adversarial dropout training stimulates the useless or the incorrectly learned neurons to better contribute to a more accurate prediction.

Semi-supervised text classification

The text classification task is one of the most important tasks utilizing RNN architecture. We evaluated our method on two text datasets, IMDB and Elec. IMDB is a standard benchmark
Table 2: Test error rates (%) on the IMDB and Elec classification tasks. All models were pre-trained by neural language models. Our experiments were repeated five times.

<table>
<thead>
<tr>
<th>Method</th>
<th>IMDB</th>
<th>Elec</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-hot CNN (Johnson and Zhang 2015; 2016)</td>
<td>6.05</td>
<td>5.87</td>
</tr>
<tr>
<td>One-hot bi-LSTM (Johnson and Zhang 2016)</td>
<td>5.94</td>
<td>5.55</td>
</tr>
<tr>
<td>Adversarial training (Miyato, Dai, and Goodfellow 2016)</td>
<td>6.21</td>
<td>5.61</td>
</tr>
<tr>
<td>Virtual adversarial training (Miyato, Dai, and Goodfellow 2016)</td>
<td>5.91</td>
<td>5.54</td>
</tr>
<tr>
<td>Adversarial + virtual adversarial training (Miyato, Dai, and Goodfellow 2016)</td>
<td>6.09</td>
<td>5.40</td>
</tr>
</tbody>
</table>

Our experiment

<table>
<thead>
<tr>
<th>Input and dropout perturbations</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Adversarial dropout (K=1) + virtual adversarial training</td>
<td>5.715</td>
<td>5.638</td>
</tr>
<tr>
<td>Adversarial dropout (K=2) + virtual adversarial training</td>
<td><strong>5.687</strong></td>
<td>5.621</td>
</tr>
</tbody>
</table>

Adversarial dropout on the recurrent connection (K=1) + virtual adversarial training achieved an impressive performance of 5.687% on IMDB.

Word-level language models

We investigated the performance of our regularization model in language modeling for two benchmark datasets, the Penn Treebank (PTB) (Mikolov et al. 2010) and WikiText-2 (WT2) dataset (Merity et al. 2017). We preprocessed both datasets: for PTB as specified by Mikolov et al. (2010) and for WT2 as specified by Koehn et al. (2007).

In this experiment, we used AWD-LSTM 3-layer architecture that was introduced by Merity et al. (2017). Specifically, the architecture contains one 400 dimensional embedding layer and three LSTM layers whose dimensions are 1150 for first two layers and 400 for the third layer. The embedding matrix and the weight matrix of the last layer for predictions were tied (Merity, McCann, and Socher 2017). In the training phase, Merity et al. (2017) applied DropConnect to improve model performances. In this experiment, we left a vast majority of hyperparameters used in the baseline model, i.e., embedding and hidden sizes, learning rate, and so on. Instead, we controlled the hyperparameters, $\gamma$ and $K$, related to the adversarial dropout. We did not use additional regularization terms except for the L2 norm for weight.
parameters. Our implementation code will be available at https://github.com/sungraepark/adversarial_dropout_lm.

Previous results with LSTM showed that fine-tuning is important to achieve state-of-the-art performances (Merity, Keskar, and Socher 2018; Li et al. 2018), so we fine-tuned model parameters once without regularization after learning was over with our regularization term. Table 3 shows the perplexity on both the PTB and WikiText-2 validation and test datasets. Our approach showed an advanced performance compared to existing benchmarks.

**Conclusion**

The improvement of generalization performance in RNNs is required for the successful application of sequential data processing. The existing methods utilizing the dropout depend on the random dropout mask without considering a guided sampling on the dropout mask. In contrast, we developed an RNN version of adversarial dropout, which is a deterministic dropout technique to find a subnetwork inferior to target predictions. Specifically, we proved that the regularizer with the adversarial dropout is the upper bound of the EL and FD regularizers. Additionally, we found a way to measure which recurrent neurons are mainly used for target predictions. Furthermore, we improved the algorithm to find the adversarial dropout condition. In our experiments, we showed that the adversarial dropout for RNNs improved generalization performance on the sequential MNIST tasks, the semi-supervised text classification tasks, and word-level language modeling tasks. More importantly, we achieved a highly advanced performance of 5.687% on IMDB when applying an adversarial perturbation on word embeddings and an adversarial dropout perturbation, together.

**Acknowledgement**

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(NRF-2018R1C1B6008652)

**References**


Maas, A. L.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the


In International Conference on Machine Learning, 1058–1066.

Zaremba, W.; Sutskever, I.; and Vinyals, O. 2015. Recurrent neural network regularization. ICLR.