ProSeqo: Projection Sequence Networks for On-Device Text Classification

Zornitsa Kozareva
Google
Mountain View, CA, USA
zornitsa@kozareva.com

Sujith Ravi
Google Research
Mountain View, CA, USA
sravi@google.com

Abstract

We propose a novel on-device sequence model for text classification using recurrent projections. Our model ProSeqo uses dynamic recurrent projections without the need to store or look up any pre-trained embeddings. This results in fast and compact neural networks that can perform on-device inference for complex short and long text classification tasks.

We conducted exhaustive evaluation on multiple text classification tasks. Results show that ProSeqo outperformed state-of-the-art neural and on-device approaches for short text classification tasks such as dialog act and intent prediction. To the best of our knowledge, ProSeqo is the first on-device long text classification neural model. It achieved comparable results to previous neural approaches for news article, answers and product categorization, while preserving small memory footprint and maintaining high accuracy.

1 Introduction

In the last decade, research in deep neural networks has lead to tremendous advances and state-of-the-art performance on wide range of Natural Language Processing (NLP), speech and vision applications. Coupled with the tremendous growth and adoption of smart devices such as mobile phones, watches, Internet of Things, conversational AI devices like Alexa and Google Home, these deep learning improvements have resulted in text and speech becoming the primary modes of communication (Cohen, 2008). To date, the majority of the models run on the server side by taking user input, sending the data to the server (or cloud) for processing and returning the results back to the user. This often results in latency delays, inconsistent user experiences and concerns over user privacy.

One way to surmount these challenges is to develop and deploy text and speech models that run inference entirely on-device and return accurate predictions in real-time. To make this work, the on-device models have to be very small in size (few kilobytes or megabytes) to fit on-devices like mobile phone, watch and IoT; to have low latency and be as accurate as server side models. These on-device challenges opened up new active area of research, which recently has shown promising results for speech (Lin et al., 2018), wake word detection (He et al., 2017), dialog act (Ravi and Kozareva, 2018) and intent prediction short text classification (Ravi and Kozareva, 2019).

Previous attempts for on-device text classification used hashed <*input_text*, *output_class*> pairs. Unfortunately, such models cannot handle examples that are not part of the lookup table and cannot generalize to more complex tasks like the ones we solve in this paper – long text classification, conversational intent prediction. (Bui et al., 2018) combined graphs with neural networks to improve the robustness of the model, but this resulted in large model sizes that cannot fit on-device. The most successful work is the self-governing neural network (SGNN) from (Ravi and Kozareva, 2018) and (SGNN++) from (Ravi and Kozareva, 2019), which use locality-sensitive projections (Ravi, 2017, 2019) to encode short text into low-dimensional representations. SGNN reached state-of-the-art results on dialog acts outperforming recurrent networks. While successful, SGNN's static projections make it hard to generalize to longer sequences.

In this work, we take one step further by developing a novel embedding free on-device neural model that combines *dynamic projections* with recurrent operations to learn powerful representations that can more powerfully encode text representations and can be equally efficient at short and long text classification tasks. The main contributions of our work are:

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 3894–3903,
Hong Kong, China, November 3–7, 2019.

- We propose novel on-device dynamic projection sequence network ProSeqo for short and long text classification; To the best of our knowledge, this is the first on-device neural network for long text classification;
- Unlike SGNN which uses static projections and fully connected networks, ProSeqo is the first work to introduce embedding-free LSTMs that combine *dynamic projections* with recurrent operations to learn powerful yet compact neural networks applicable for on-device scenarios;
- We conduct exhaustive evaluations and comparisons against state-of-the-art on-device and deep learning models for short and long text classification;
- Our results show that ProSeqo outperformed state-of-the-art on-device SGNN network (Ravi and Kozareva, 2018) with up to +8.9% accuracy on short text classification and +35.9% accuracy on long document classification. This shows that ProSeqo's dynamic recurrent projections are more powerful than the static ones used in SGNN;
- Similarly, our results show that ProSeqo outperformed the non-on-device neural models like RNN (Khanpour et al., 2016), CNN (Lee and Dernoncourt, 2016) for dialog acts and intent prediction; as well as LSTMs (Zhang et al., 2015), character CNNs (Zhang et al., 2015; Bui et al., 2018) for long document classification of news, answers and product reviews. This is a significant advancement given ProSeqo's small model size compared to these widely used approaches;
- Finally, we conduct a series of ablation studies to demonstrate the effect of ProSeqo's projection size on accuracy and the ability of ProSeqo to produce small model sizes while retaining high accuracy.

2 ProSeqo: Projection Sequence Networks for On-device Text Classification

We introduce a neural architecture, *ProSeqo*, a projection-based sequence network, that is designed to (a) be efficient and learn compact *embedding-free* neural representations of text at the semantic level, (b) capture contextual and morphological information at the character-level, and (c) model longer context across words and sentences that is suited for both short and long-text classification tasks.

The key differences to prior work (e.g., RNNs, CNNs) are that we use context information to *dynamically* compute text representations and learn compact neural networks for text classification that are suited for on-device applications. Our work departs significantly from recent on-device work (Ravi and Kozareva, 2018) and (Ravi and Kozareva, 2019) which leverage projection operations to learn efficient networks that can even be transferred to other tasks (Sankar et al., 2019b). However, they are limited to short-text classification tasks. In contrast, our model effectively uses contextual information and combines *recurrent* and *projection* operations to achieve efficiency and enable learning more powerful neural networks that generalize well and can solve more complex language classification tasks. As a result, our projection sequence network is able to dynamically project and learn efficient neural classifier models that are competitive with state-of-the-art RNNs and CNNs without the need to store or lookup any pre-trained embeddings.

2.1 Model Overview

The overall architecture of ProSeqo is shown in Figure 1. It consists of multiple parts: a word context projector, a recurrent projector, a projection sequence encoder, and a classification layer. Next, we describe in details each component.

2.2 Word Context Projector

Given an input text x_i (sentence or document) with sequence of words w_{it}, where w_{it} refers to t-th word in the input, we first project the input x_i to a vector representation $\mathbb{P}(x_i)$. Majority of the neural models used in NLP rely on embedding matrices (word or character level) to achieve this. Instead, we dynamically compute word vector representations using *context* and locality-sensitive projections (Ravi, 2017).

2.2.1 Character and Morphology Projection

Each word contains characters and morphological information that can be informative for the task. For example, the words `amaze` and `amazing`, may appear slightly different but capture similar semantics for a review classification task. It

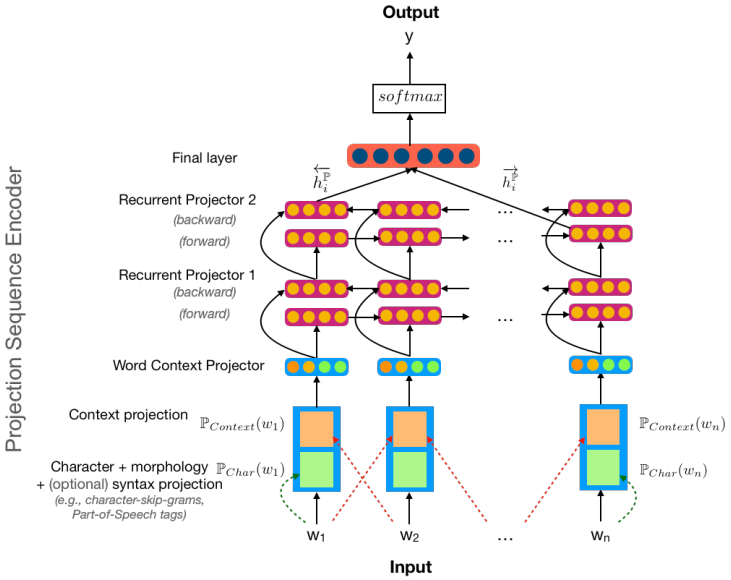

Figure 1: Model architecture for ProSeqo: On-device Projection Sequence Neural Networks.

is particularly challenging for NLP tasks to handle large vocabularies and model *unknown* words missing from training data. Previous methods have proposed to overcome this by relying on character-level embeddings and other neural models like character-CNNs. However, these methods are often complex and slow to compute for long text (e.g., convolution kernels on devices without significant computational capacity) and still require explicitly storing character or sub-word sequences. For example, even simple character trigrams over 26 characters in the English alphabet requires storing and looking up $V = 26^3 = 18K$ entries in character embedding table and complexity of $O(V \cdot d)$, where d is the dimensionality of the embedding. For word embeddings, V can be as large as 100K to millions of entries.

We extract character-level sequence information (i.e., character sequences) and use locality-sensitive projections (Ravi, 2017, 2019) to dynamically project them to a vector representation $\mathbb{P}_{Char}(w_{it})$ with just $O(T \cdot d)$ complexity, where $T << V$.

$$\mathbb{P}_{Char}(w_{it}) = \mathbb{P}(w_{itc_{1\ldots C}}) \quad (1)$$
$$= \mathbb{P}(\overrightarrow{w_{it}}) \quad (2)$$

where, $w_{itc_{1\ldots C}}$ is the sequence of characters $c \in [1, C]$ in word w_{it} and $\overrightarrow{w_{it}}$ is a sparse vector comprised of character-level features (character skip-grams) extracted from the word w_{it}. We can also extend this to include other morphological (e.g., stemming) or syntax features (e.g., Part-of-Speech) but this adds further complexity and requires lexicons or further pre-processing steps (e.g., stemmers, taggers) which may not be available on device during inference. Hence, we only use simple raw features like character skip-grams in the projector.

$$\mathbb{P}(\overrightarrow{w_{it}}) = \texttt{LSH}_{proj}(\overrightarrow{w_{it}}, \overrightarrow{\mathbb{P}_{T,d}}) \quad (3)$$
$$= [\ sgn(\overrightarrow{w_{it}} \cdot \overrightarrow{\mathbb{P}_{1,1}}), ..., \quad (4)$$
$$..., sgn(\overrightarrow{w_{it}} \cdot \overrightarrow{\mathbb{P}_{T,d}})\ ]$$

We then apply LSH projections that dynamically compute locality-sensitive projections and transform the extracted sparse vector $\overrightarrow{w_{it}}$ (character sequence features) into a compact, low-dimensional binary representation $\mathbb{P}(\overrightarrow{w_{it}})$. The projection functions $\overrightarrow{\mathbb{P}}$ used to compute the binary vector are dynamically generated from observed features in $\overrightarrow{w_{it}}$ and a random seed. T, d represents the number of projection functions and dimensonality of each binary vector produced by each projection function. So, the output of the character-level projector is a $T \cdot d$ binary vector.

The locality-sensitive nature of the projections help learn a compact representation that capture semantic similarity (Sankar et al., 2019a) (i.e., words with similar character or morphology are mapped to the same binary representation) with a small memory footprint. For more details on LSH projections, refer to (Ravi, 2017).

2.2.2 Context Projection

In addition to characters, the surrounding context also contains important information useful to represent in the word vector. Unlike characters though, word context is typically not fixed in language and changes across sentences and documents. To address this, we extend the projection mechanism to capture word meaning using contextual information.

$$\begin{aligned}
\mathbb{P}_{Context}(w_{it}) &= \mathbb{P}(context(w_{it})) && (5)\\
&= \mathbb{P}(w_{it-\Delta}...w_{it-1}, && (6)\\
&\quad\ w_{it+1}...w_{it+\Delta})\\
&= \mathbb{P}(\overrightarrow{w_{it,\Delta}}) && (7)\\
&= \texttt{LSH}_{proj}(\overrightarrow{w_{it,\Delta}}, \overrightarrow{\mathbb{P}_{T,d}}) && (8)
\end{aligned}$$

where, Δ represents the context size around the current word w_{it}, $\overrightarrow{w_{it,\Delta}}$ is the context features extracted from neighboring context. We use simple raw features like word unigrams to model the context and apply dynamic projections on the sparse context feature vector to transform this to a binary context projection vector.

2.3 Recurrent Projector

We use LSTM to model the state of word projection sequences within a sentence or document. This allows the model to capture words appearing in different parts of the input text, and use the projection space to learn an efficient neural representation for the task. At each time-step t, we compute the word projection by combining both character and context information, and transforming them via projections.

$$\mathbb{P}(x_t) = [\ \mathbb{P}_{Char}(x_t),\ \mathbb{P}_{Context}(x_t)\ ] \quad (9)$$

The LSTM-projector $\mathbb{P}_{\text{RNN}}$ then computes the new state $h_t^{\mathbb{P}}$ using the projector output as follows

$$\begin{aligned}
f_t^{\mathbb{P}} &= \sigma(W_f \cdot [h_{t-1}, \mathbb{P}(x_t)] + b_f) && (10)\\
i_t^{\mathbb{P}} &= \sigma(W_i \cdot [h_{t-1}, \mathbb{P}(x_t)] + b_i) && (11)\\
\widetilde{C_t^{\mathbb{P}}} &= \tanh(W_C \cdot [h_{t-1}, \mathbb{P}(x_t)] + b_C) && (12)\\
C_t^{\mathbb{P}} &= f_t^{\mathbb{P}} * C_{t-1}^{\mathbb{P}} + i_t^{\mathbb{P}} * \widetilde{C_t^{\mathbb{P}}} && (13)\\
o_t^{\mathbb{P}} &= \sigma(W_o \cdot [h_{t-1}, \mathbb{P}(x_t)] + b_o) && (14)\\
h_t^{\mathbb{P}} &= o_t^{\mathbb{P}} * \tanh(C_t^{\mathbb{P}}) && (15)
\end{aligned}$$

where, $f^{\mathbb{P}}, i^{\mathbb{P}}, o^{\mathbb{P}}$ represent the projection forget, input and output gates respectively, σ is a non-linearity (ReLu) and W, b are the weight and bias parameters. $h_t^{\mathbb{P}}$ is the new state computed from previous state $h_{t-1}^{\mathbb{P}}$. Note that we can replace LSTM with GRU (Cho et al., 2014) or other variants to control the gating mechanism and derive other recurrent projector model variants.

2.4 Projection Sequence Encoder

We use a bidirectional recurrent projector $\mathbb{P}_{\text{RNN}}$ that encodes the input word sequence by summarizing information from both directions for word projections. The sequence encoder captures both character-level and contextual information across time steps through the projection space $\Omega_{\mathbb{P}}$.

The projection sequence encoder contains a forward encoder $\overrightarrow{\mathbb{P}_{\text{RNN}}}$ that processes the sentence (or document) from w_{i1} to w_{iT} and the backward encoder $\overleftarrow{\mathbb{P}_{\text{RNN}}}$ which reads the input text in reverse.

$$\begin{aligned}
\overrightarrow{xh_i^{\mathbb{P}}} &= \overrightarrow{\mathbb{P}_{\text{RNN}}}(x_i) && (16)\\
\overleftarrow{h_i^{\mathbb{P}}} &= \overleftarrow{\mathbb{P}_{\text{RNN}}}(x_i) && (17)\\
h_i &= [\ \overrightarrow{h_i^{\mathbb{P}}},\ \overleftarrow{h_i^{\mathbb{P}}}\ ] && (18)
\end{aligned}$$

We concatenate the two states to model the state h_i and transition of word projection sequences within a sentence or document.

Stacked Recurrent Projections: We also stack the recurrent projections to create deeper, multi-layered ProSeqo. Each layer is a bidirectional recurrent projection encoder with information from the forward and backward states passed to the next layer. In our work, we train ProSeqo with 2 layers.

2.5 Text Classifier

The final state of the projection sequence encoder h_{iF} is combined with an output layer to learn a classifier for a given task.

$$y_i = \texttt{softmax}(W_F \cdot h_{iF} + b_F) \quad (19)$$

We train ProSeqo via backpropagation using stochastic gradient descent with negative log likelihood of correct labels as the loss.

3 Text Classification Tasks

In this section, we describe the tasks and datasets for our experimental evaluation. We choose the datasets to compare against prior state-of-the-art on-device work (Ravi and Kozareva, 2018), and also to test the generalizability of our on-device model on short and long texts.

Text Classification Task		#Classes	Vocabulary	Avg. Length	Train	Test
SWDA	Dialog Act	42	20,000	7	193,000	5,000
MRDA	Dialog Act	6	12,000	8	78,000	15,000
ATIS	Intent Prediction	21	950	11	4,478	893
SNIPS	Intent Prediction	7	11,241	10	13,084	700
AG	News Categorization	4	156,000	38	120,000	7,600
Y!A	Yahoo! Answers Categorization	10	1,554,607	108	1,400,000	60,000
AMZN	Product Review Prediction	5	1,919,336	92	3,000,000	650,000

Table 1: Text Classification Tasks and Dataset Characteristics

3.1 Dataset Description

• **SWDA:** Switchboard Dialog Act Corpus is a popular open domain dialog corpus between two speakers with 42 dialog acts (Godfrey et al., 1992; Jurafsky et al., 1997). The dataset is used in the on-device work of (Ravi and Kozareva, 2018).

• **MRDA:** Meeting Recorder Dialog Act is a dialog corpus of multiparty meetings annotated with 6 dialog acts (Adam et al., 2003; Shriberg et al., 2004). The dataset was also used by on-device work of (Ravi and Kozareva, 2018). We used the same data split as (Lee and Dernoncourt, 2016; Ortega and Vu, 2017; Ravi and Kozareva, 2018).

• **ATIS:** The Airline Travel Information Systems dataset (Tür et al., 2010) is widely used in dialog and speech research. The dataset contains audio recordings of people making flight reservations (Tür et al., 2010; Goo et al., 2018).

• **SNIPS:** To test the generalizability of our model, we use another NLU dataset with custom intent-engines collected by Snips personal voice assistant. We used the data from (Goo et al., 2018). Compared to the single-domain ATIS dataset, Snips is more complicated mostly because of the rich and diverse intent repository combined with the larger vocabulary.

• **AG:** AG News corpus is a collection of news articles on the web, where each document has a title and a description field. We used the dataset from (Zhang et al., 2015).

• **Y!A:** Yahoo! Answers is a text classification task with 10 diverse classes: *Society & Culture*, *Science & Mathematics*, *Health*, *Education & Reference*, *Computers & Internet*, *Sports*, *Business & Finance*, *Entertainment & Music*, *Family & Relationships* and *Politics & Government*. Each document contains a question title, question context and best answers. We obtained the data from (Zhang et al., 2015). Note that (Yang et al., 2016) used a smaller test sample for evaluation. To present fair results, we will not compare to (Yang et al., 2016).

• **AMZN:** Amazon review dataset is obtained from (Zhang et al., 2015). Resolving this task can be very helpful for product categorization (Kozareva, 2015). The corpora has reviews with ratings ranging from 1 to 5. Following prior work, we use 3,000,000 reviews for training and 650,000 reviews for testing.

3.2 Dataset Characteristics

Table 1 shows the datasets characteristics such as the type of the task, number of classes, vocabulary size, average text length, train and test sizes. As shown in Table 1, the datasets are very diverse, some have few thousand training examples, while others have millions. The test sizes also differ. Similarly, there are tasks with small and huge vocabularies. These diverse characteristics help validate and capture the ability of our on-device model ProSeqo to generalize to different tasks, data constraints, settings and vocabulary. With respect to the text length, we have two major categories: short text tasks such as dialog act and intent prediction, and long text tasks such as news, product reviews and answer categorization.

3.3 Experimental Setup & Model Parameters

We setup our experiments as given a classification task and dataset, we apply ProSeqo to train an on-device model. For each task, we report *Accuracy* on the test set. Unlike typical NLP approaches including baseline neural methods, we do **not** apply any vocabulary pruning or pre-processing at all to the input sentences and documents, except for splitting tokens by space.

We use a 2-layer *ProSeqo* neural network with recurrent projections of size $T = 60, d = 14$ and 256 hidden dimensions to represent state in each bidirectional LSTM-projection layer. We use character 7-grams (with 1-skip) and context size of 1 to model the projector described in Section 2.2. ProSeqo network is trained with SGD and Adam

Model	SWDA	MRDA	ATIS	SNIPS
ProSeqo (our on-device model)	**88.3**	**90.1**	**97.8**	**97.9**
SGNN(Ravi and Kozareva, 2018)(**on-device**)	83.1	86.7	*88.9*	*93.4*
RNN(Khanpour et al., 2016)	80.1	86.8	-	-
RNN+Attention(Ortega and Vu, 2017)	73.8	84.3	-	-
CNN(Lee and Dernoncourt, 2016)	73.1	84.6	-	-
GatedIntentAtten.(Goo et al., 2018)	-	-	94.1	96.8
GatedFullAtten.(Goo et al., 2018)	-	-	93.6	97.0
JointBiLSTM(Hakkani-Tur et al., 2016)	-	-	92.6	96.9
Atten.RNN(Liu and Lane, 2016)	-	-	91.1	96.7

Table 2: Short Text Classification On-device Results & Comparisons to Prior Work

optimizer (Kingma and Ba, 2014) over shuffled mini-batches of size 100. We did not do any additional dataset-specific tuning or processing.

4 STC: Short Text Classification Results

This section focuses on the multiple short text classification experiments we have conducted. Table 2 shows the results on the dialog act and intent prediction tasks for SWDA, MRDA, ATIS and SNIPS datasets. Overall, ProSeqo reached the highest performance and significantly improved upon prior state-of-the-art on-device neural approaches (Ravi and Kozareva, 2018). Similarly, ProSeqo outperformed non-on-device neural models like RNN (Khanpour et al., 2016), CNN (Lee and Dernoncourt, 2016) and joint neural approaches (Goo et al., 2018; Hakkani-Tur et al., 2016).

4.1 STC: Comparison with On-Device Work

We compare our on-device model against state-of-the-art on-device short text classification approach SGNN (Ravi and Kozareva, 2018). SGNN was evaluated only on the SWDA and MRDA dialog act tasks (Ravi and Kozareva, 2018) and reached state-of-the-art performance against prior non-on-device neural approaches (Khanpour et al., 2016). We directly compare performance on the same SWDA and MRDA datasets. As shown in Table 2 ProSeqo reaches +5.3% improvement for SWDA and +3.4% accuracy improvement for MRDA.

Since we want to compare on-device performance on a wider spectrum of tasks and datasets, we re-implemented SGNN with the same parameters (Ravi and Kozareva, 2018). We run experiments on ATIS and SNIPS intent prediction tasks and reported results in Table 2 in italic to indicate that this is a re-implementation of (Ravi and Kozareva, 2018) and previously these on-device results were not reported. As shown in Table 2, SGNN consistently performs well on dialog act and intent prediction tasks. Overall, ProSeqo reached +8.9% accuracy improvements on ATIS and +4.5% accuracy improvements on SNIPS compared to SGNN. This shows that ProSeqo's recurrent dynamic projections learn more powerful representations than the static SGNN ones, this also leads to significant performance improvements on multiple tasks.

4.2 STC: Comparison with Non-On-Device

The main objective of on-device work is to develop small and efficient models that fit on devices with limited memory and capacity. In contrast, the non-on-device models do not have any constraints and can use all resources available on server side. Therefore, it is not fair to directly compare the on-device and non-on-device models. Taking these differences in consideration, we show in Table 2 results from non-on-device models with the objective to highlight the power of on-device models and their capability to produce small memory footprint models, which are competitive in accuracy to those using pre-trained embeddings and unconstrained resources.

As shown in Table 2, prior work focused on developing the best approach for a specific task and dataset, resulting in not having a single model across all tasks. ProSeqo is the only approach spanning across all tasks and datasets. In terms of SWDA and MRDA, ProSeqo significantly improves with +8.2% accuracy and +3.3% accuracy the best performing non-on-device neural approach (Khanpour et al., 2016). For ATIS and SNIPs, the most recent state-of-art approaches use joint intent and slot prediction model (Hakkani-Tur et al., 2016; Liu and Lane, 2016; Goo et al., 2018), where the slot model recognizes the enti-

Model	AG	Y!A	AMZN
ProSeqo (our on-device model)	**91.5**	**72.4**	**62.3**
SGNN (Ravi and Kozareva, 2018)(**on-device**)	*57.6*	*36.5*	*39.3*
FastText-full(Joulin et al., 2016)	**92.5**	72.3	60.2
CharCNNLargeWithThesau.(Zhang et al., 2015)	90.6	71.2	59.6
CNN+NGM(Bui et al., 2018)	86.9	-	-
LSTM-full(Zhang et al., 2015)	86.1	70.8	59.4
Hier.-Attention(Yang et al., 2016)	-	-	**63.6**
Hier.-AVE(Yang et al., 2016)	-	-	**62.9**

Table 3: Long Text Classification On-device Results & Comparisons to Prior Work

ties and their semantic categories in the text, and the intent prediction model uses the slot information. (Liu and Lane, 2016) showed that joint intent and slot model improves upon the individual ones. While the entities and their semantic types are useful, ProSeqo does not rely on such additional information. ProSeqo simply uses the text and different context to learn compact semantic projection representations on-the-fly to make the intent prediction. ProSeqo improves with +3.7% the best approach for ATIS, which is a gated intent attention (Goo et al., 2018), and with +1% the best approach for SNIPS, which is gated full attention (Goo et al., 2018). (Goo et al., 2018) developed two complimentary approaches one with full attention and one with gated intent attention, yet they do not have consistent improvements on both datasets and tasks. In our case, our on-device ProSeqo model consistently improved upon both tasks, and reached state-of-the-art performance over prior non-on-device neural approaches. This is a significant improvement given the small size of ProSeqo and the fact that it does not store or lookup any pre-trained embedding matrices.

5 LDC: Long Document Classification Results

This section focuses on long document classification results. Table 3 shows the results on three tasks and datasets (AG, Y!A, AMZN). Overall, ProSeqo significantly improved upon the on-device neural model SGNN (Ravi and Kozareva, 2018) with +23% to +35.9% accuracy. ProSeqo also reached comparable performance to prior non-on-device neural LSTMs and character CNNs approaches (Zhang et al., 2015; Bui et al., 2018).

5.1 LDC: Comparison with On-Device Work

To the best of our knowledge, ProSeqo is the first on-device neural approach for long document classification. Since prior on-device work focused mostly on short text, we re-implemented the on-device state-of-the-art neural approach SGNN (Ravi and Kozareva, 2018) and run it on multiple long text tasks and datasets. Similarly to the short text section, we reported SGNN results in Table 3 in italic to indicate that this is a re-implementation of (Ravi and Kozareva, 2018) and previously results on these datasets and tasks were not reported.

As shown in Table 3, SGNN performed well on short texts, but it did not reach high performance on long documents. It it important to highlight that ProSeqo maintained consistent performance across all short and long document classification tasks. For long texts, ProSeqo improved SGNN with +33.9% accuracy on AG news, with +35.9% accuracy on Y!A and with +23% accuracy on Amazon product review classification. These are significant improvements and further show the importance of ProSeqo's recurrent dynamic projections for short and long text classification.

5.2 LDC: Comparison with Non-On-Device

Similarly to the short text classification, we also conduct a comparison of non-on-device model for long document classification. As highlighted before, such comparison is not fair due to the fact that non-on-device models do not have to comply with small memory, small size and low latency constraints. Bearing this in mind, we show results in Table 3. As it can be seen, only FastText (Joulin et al., 2016), CharCNN with Thesaurus (Zhang et al., 2015) and LSTM-full (Zhang et al., 2015) have been evaluated on all datasets, while CNN+NGM (Bui et al., 2018) focused on reaching the best performance for AG news. ProSeqo outperforms (Joulin et al., 2016; Zhang et al., 2015,?; Bui et al., 2018), and reaches comparable results to FastText (Joulin et al., 2016) on AG news. The hierarchical attention and hierarchi-

cal average models of (Yang et al., 2016) were evaluated only on the Amazon reviews and Y!A datasets. However, we noticed that the test set of the Y!A dataset in (Yang et al., 2016) was smaller compared to the data used in (Zhang et al., 2015; Joulin et al., 2016) and our experiments. To make consistent and fair comparisons, we did not include (Yang et al., 2016). For Amazon reviews, ProSeqo reached comparable performance to hierarchical average and hierarchical attention models (Yang et al., 2016). Overall, ProSeqo reached similar performance to non-on-device neural approaches, which use pre-trained embeddings while ProSeqo does not store or lookup any pre-trained embedding matrices.

6 ProSeqo Ablation Studies

This section shows two ablation studies focusing on the: (1) effect of the projection size on accuracy and (2) model size on accuracy.

6.1 Effect of Projection Size on Accuracy

During series of evaluations on multiple short and long text classification tasks, and comparing results against prior state-of-the-art neural networks and on-device neural models, ProSeqo achieved robust performance irrespective of the task, vocabulary size, training and test datasets. This is significant improvement given the small memory size of the model and the fact that we used the same model, configuration and model parameters across all seven NLP tasks unlike prior work which fine-tuned on specific data.

We also evaluated ProSeqo with different number of parameters T, d used in the recurrent projection. Table 4 shows the performance of ProSeqo on the AG News classification task and indicates that even if we reduce the projection size by 3x, we still achieve 90% accuracy with a small drop in performance of 1.4% (91.5 $\rightarrow$ 90.09).

Projection Size ($T \cdot d$)	300	600	800
Compression Ratio ($\frac{S_1}{S_2}$)	15×10^4	7.5×10^4	5.6×10^4
Accuracy	90.0	91.0	**91.5**

Table 4: Effect of ProSeqo's Recurrent Projection Size on Accuracy for AG News Classification. *Compression Ratio* is computed wrt ProSeqo model size (S_1) vs. LSTM baseline (S_2) with word embeddings (vocabulary size=150K, embedding size=300).

6.2 ProSeqo's Model Size

In addition to the quality evaluations, comparisons and results, we also highlight how compact and fast our on-device model is, which is very important for fitting the model on small memory-constrained devices such as mobile phones and watches. For this purpose, we measure the current size of our ProSeqo model. We do not have the exact parameter counts and model sizes from the previous neural approaches such as RNNs, BiLSTMs, and Hierarchical Attention models. But we make an observation that number of parameters in the recurrent layers of these baseline networks are similar to the parameters encoded in the recurrent projector layers in the ProSeqo model. Hence we can approximately compute the difference in model sizes between ProSeqo and other neural models by taking into consideration the size of embeddings fed into the recurrent layers along with the training data vocabulary size.

As highlighted in the main contributions of our approach ProSeqo is an embedding-free method, which means that it does not have to store or look up any embedding matrices. This makes the model small in comparison to existing neural approaches. For instance, to calculate the model size of ProSeqo as explained in Section 2 we need to consider $(T \cdot d)$ where d is the number of LSH bits specified for each projection vector, and T is the number of projection functions used resulting in $(T \cdot d) = 60 \times 14$ parameters shared across all time-steps. Independent of the task, length of the text (*short* vs *long*) and vocabulary sizes, ProSeqo's model size remains constant $(T \cdot d)$ across all seven tasks and datasets. Comparing ProSeqo's model size against a neural approach that uses Glove (or word2vec) word embeddings with 300 dimensions and vocabulary size V for the datasets, we can easily see that the neural models will have bigger sizes and their model size also varies depending on the task and vocabulary size. For instance, the vocabulary size in AG dataset is $150K$ resulting in $150K \times 300$ more model parameters than ProSeqo. For Yahoo this is $1.5M \times 300$ and for Amazon model size is $1.9M \times 300$. As it can be seen, ProSeqo produces models that are orders of magnitude smaller compared to prior neural approaches. On AG dataset, ProSeqo model has $5 \cdot 10^4$x fewer parameters for the first layer which is equivalent to just 3.3KB in size using `float32` parameters compared to 180MB for other neural methods. ProS-

eqo's size remains constant independent of the text lengths, vocabulary size and classification tasks. Even with neural models like charCNN that use character-level embeddings which is typically applied in conjunction with word embeddings, ProSeqo still yields an order or two magnitude smaller model sizes. The computation within ProSeqo is fast, $O(T \cdot d \cdot n)$ `multiply-add` operations for n-length words regardless of word/character vocabulary sizes.

7 Conclusion

We proposed a novel Projection Sequence Neural Network (ProSeqo) for on-device text classification. Evaluations on multiple text classification tasks show that ProSeqo significantly improved accuracy compared to state-of-the-art on-device neural network SGNN (Ravi and Kozareva, 2018) on short text with +3.4% for MRDA, +4.5% on SNIPS, +5.3% on SWDA and +8.9% on ATIS; and long documents with +23% on Amazon product reviews, +33.9% on AG news, +35.9% on Y!A. Similarly, ProSeqo improved non-on-device neural RNN, CNN and joint approaches (Khanpour et al., 2016; Lee and Dernoncourt, 2016; Hakkani-Tur et al., 2016; Goo et al., 2018) on short text and reached comparable results on long documents (Joulin et al., 2016; Zhang et al., 2015; Bui et al., 2018). This is significant improvement given ProSeqo's small network size, and the fact that we used the same architecture and parameters across all seven NLP tasks, while prior work tuned depending on the task and dataset.

References

Janin Adam, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Nelson Morgan, Barbara Peskin, Thilo Pfau, Elizabeth Shriberg, Andreas Stolcke, and Chuck Wooters. 2003. The icsi meeting corpus. In *Proceedings of the 5TH SIGdial Workshop on Discourse and Dialogue*, pages 364–367.

Thang D. Bui, Sujith Ravi, and Vivek Ramavajjala. 2018. Neural graph learning: Training neural networks using graphs. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 64–71.

Kyunghyun Cho, Bart van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Jordan Cohen. 2008. Embedded speech recognition applications in mobile phones: Status, trends, and challenges. pages 5352 – 5355.

John J. Godfrey, Edward C. Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech and Signal Processing - Volume 1*, ICASSP'92, pages 517–520. IEEE Computer Society.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 753–757. Association for Computational Linguistics.

Dilek Hakkani-Tur, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Vivian Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Proceedings of The 17th Annual Meeting of the International Speech Communication Association (INTERSPEECH 2016)*.

Yanzhang He, Rohit Prabhavalkar, Kanishka Rao, Wei Li, Anton Bakhtin, and Ian McGraw. 2017. Streaming small-footprint keyword spotting using sequence-to-sequence models. *CoRR*, abs/1710.09617.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hervé Jégou, and Tomas Mikolov. 2016. Fasttext.zip: Compressing text classification models. *CoRR*, abs/1612.03651.

Daniel Jurafsky, Rebecca Bates, Rachel Martin Noah Coccaro, Marie Meteer, Klaus Ries, Elizabeth Shriberg, Audreas Stolcke, Paul Taylor, and Van Ess-Dykema. 1997. Automatic detection of discourse structure for speech recognition and understanding. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 88–95.

Hamed Khanpour, Nishitha Guntakandla, and Rodney Nielsen. 2016. Dialogue act classification in domain-independent conversations using a deep recurrent neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2012–2021.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Zornitsa Kozareva. 2015. Everyone likes shopping! multi-class product categorization for e-commerce. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1329–1333.

Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 515–520.

Zhong Qiu Lin, Audrey G. Chung, and Alexander Wong. 2018. Edgespeechnets: Highly efficient deep neural networks for speech recognition on the edge. *CoRR*, abs/1810.08559.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *Proceedings of The 17th Annual Meeting of the International Speech Communication Association (INTERSPEECH 2016)*.

Daniel Ortega and Ngoc Thang Vu. 2017. Neural-based context representation learning for dialog act classification. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 247–252.

Sujith Ravi. 2017. Projectionnet: Learning efficient on-device deep networks using neural projections. *CoRR*, abs/1708.00630.

Sujith Ravi. 2019. Efficient on-device models using neural projections. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5370–5379.

Sujith Ravi and Zornitsa Kozareva. 2018. Self-governing neural networks for on-device short text classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 804–810.

Sujith Ravi and Zornitsa Kozareva. 2019. On-device structured and context partitioned projection networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3784–3793.

Chinnadhurai Sankar, Sujith Ravi, and Zornitsa Kozareva. 2019a. On the robustness of projection neural networks for efficient text representation: An empirical study. *ArXiv*, abs/1908.05763.

Chinnadhurai Sankar, Sujith Ravi, and Zornitsa Kozareva. 2019b. Transferable neural projection representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3355–3360.

Elizabeth Shriberg, Raj Dhillon, Sonali Bhagat, Jeremy Ang, and Hannah Carvey. 2004. The icsi meeting recorder dialog act (mrda) corpus. In *Proceedings of the 5th SIGdial Workshop on Discourse and Dialogue*, pages 97–100, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Gökhan Tür, Dilek Hakkani-Tür, and Larry P. Heck. 2010. What is left to be understood in atis? In *Proceedings of 2010 IEEE Spoken Language Technology Workshop (SLT)*, pages 19–24.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 649–657. MIT Press.