On-device Structured and Context Partitioned Projection Networks

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

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

Abstract

A challenging problem in on-device text classification is to build highly accurate neural models that can fit in small memory footprint and have low latency. To address this challenge, we propose an on-device neural network SGNN++ which dynamically learns compact projection vectors from raw text using structured and context-dependent partition projections. We show that this results in accelerated inference and performance improvements.

We conduct extensive evaluation on multiple conversational tasks and languages such as English, Japanese, Spanish and French. Our SGNN++ model significantly outperforms all baselines, improves upon existing on-device neural models and even surpasses RNN, CNN and BiLSTM models on dialog act and intent prediction. Through a series of ablation studies we show the impact of the partitioned projections and structured information leading to 10% improvement. We study the impact of the model size on accuracy and introduce quantization-aware training for SGNN++ to further reduce the model size while preserving the same quality. Finally, we show fast inference on mobile phones.

1 Introduction

Over the last years, the usage of conversational assistants has become extremely popular. On a daily basis, people request weather information, check calendar appointments, perform calls. Large part of the conversational and natural language understanding happens on the server side and then fulfilled resulting in response delays, inconsistent experience and privacy concerns. Therefore, there is a huge demand for developing on-device natural language models that work entirely on-device such as mobile phones, tablets, watches and any internet of things (IoT) devices. On-device computation can circumvent the latency delays, can increase the user privacy and further enable new capabilities for real time interaction.

One way to develop on-device natural language understanding is to leverage the power of deep neural networks, which over the years have shown tremendous progress and have improved upon state-of-the-art machine learning methods in Natural Language Processing (NLP) (Sutskever et al., 2014), Speech (Hinton et al., 2012) and Vision (Krizhevsky et al., 2012). These advancements were byproducts of the availability of large amounts of data and high performance computing, enabling the development of more complex and robust neural network architectures. However, despite their success, yet it remains challenging to deploy deep networks on-device such as mobile phone, smart watch and IoT. The limited memory and computation power combined with the need of fast latency require the development of novel on-device neural networks.

Inspired by (Ravi and Kozareva, 2018), we propose a novel on-device neural network (SGNN++) that uses joint structured (word+character) information and context partitioned projections to learn robust models for short text classification. We employ a modified version of the locality sensitive hashing (LSH) to reduce input dimension from millions of unique words/features to a short, fixed-length sequence of bits (Ravi, 2017, 2019). This allows us to compute a projection for an incoming text very fast, on-the-fly, with a small memory footprint on the device without storing any incoming text and word embeddings.

Unlike prior work that focused on developing the best neural network for a specific NLP task and language, we develop one SGNN++ architecture with the same parameters and apply it to wide range of tasks and languages such as En-

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3784–3793
Florence, Italy, July 28 - August 2, 2019.

glish, French, Spanish and Japanese. Our experimental results show that SGNN++ improves upon baselines, prior on-device state-of-the-art and even non-on-device RNN, CNN and BiLSTM methods. The main contributions of our paper are:

- Novel embedding-free SGNN++ on-device neural model with quantization, and joint structured and context partitioned projections;
- Novel context partitioned projections result in small memory footprint with better performance and speedup.
- First on-device model evaluated on a *wide range of applications* such as dialog act, intent prediction, customer feedback.
- First on-device model evaluation on English, Spanish, French and Japanese languages demonstrating the language agnostic power of SGNN++ .
- Comparison against prior on-device state-of-the-art neural models, which SGNN++ significantly improves upon across multiple tasks.
- Ablation studies that show the impact of word vs joint word and character representation on accuracy; the power of the partitioned projection vectors on speed and inference; and the ability of SGNN++ to compress large models while still maintaining high accuracy; the fast latency of the on-device model.

2 On-device Partitioned Projection Networks (SGNN++)

We propose new on-device neural network architectures for NLP inspired by *projection* model architectures (Ravi, 2017, 2019). The *projection* model is a neural network with dynamically-computed layers that encodes a set of efficient-to-compute operations which can be performed directly on device for inference.

Unlike prior work that employs projections (Ravi and Kozareva, 2018), our new model defines a set of efficient *structured* and *context-dependent* "projection" functions $\mathbb{P}_C(x_i)$ that progressively transform each input instance x_i to a different space $\Omega_{\widetilde{\mathbb{P}}_C}$ and then performs learning in this space to map it to corresponding outputs y_i. The model applies dynamically-computed projection functions that are conditioned on context in multiple ways to achieve higher discriminative power (for classification tasks) and better efficiency wrt memory footprint and speedup. Firstly, we introduce a **joint structured projection model** that uses language structure to project *word* and *character* information in each input instance separately ($\Omega_{\widetilde{\mathbb{P}}=\Omega_{\widetilde{\mathbb{P}}_w} \bigcup \Omega_{\widetilde{\mathbb{P}}_c}}$) and combines them during learning. Secondly, we introduce **context-partitioned** projection functions $\mathbb{P}_{C_k}(x_i)$ that leverage feature-context hierarchy to partition the projection space $\Omega_{\widetilde{\mathbb{P}}}$ based on context type. Both these methods enable learning powerful compact neural networks that achieve high performance and fast inference with low memory footprint.

2.1 SGNN++ Architecture

Our on-device projection partitioned neural network architecture is a deep multi-layered context-dependent locality-sensitive projection model. Figure 1 shows the model architecture. The neural model uses projections (Ravi, 2017, 2019) making it an embedding-free approach, i.e., the model can be learned without the need to initialize, load or store any feature or vocabulary weight matrices. This is different from the majority of the widely-used state-of-the-art deep learning techniques in NLP whose performance depends on embeddings pre-trained on large corpora. In this work, we also introduce a novel joint *structured projections* and *context partitioned projection spaces* that result in highly efficient and compact neural network models for on-device applications. We will also show how SGNN++ yields significant improvements over prior work (Ravi and Kozareva, 2018) and reaches state-of-the-art on multiple NLP tasks and languages.

2.2 Model Overview

In this work, we focus on short text classification. Each input x_i contains a sequence of tokens, where x_{it} represents the t-th token in the input. The proposed SGNN++ model progressively projects each raw input text x_i to an efficient vector representation $\widetilde{i_p}$ and then learns a classifier to map x_i to output class y_i.

The raw input text x_i is first converted to an intermediate feature vector $\mathbb{F}(x_i)$ using raw text features such as skip-grams.

$$\vec{x_i} = \mathbb{F}(x_i) \quad (1)$$

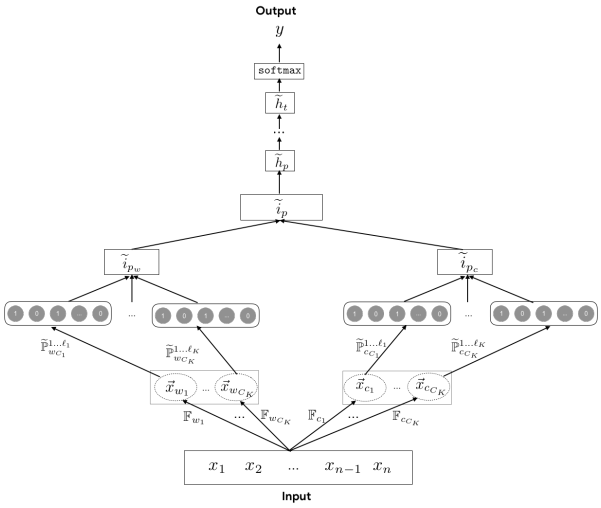

Figure 1: SGNN++ Model Architecture: On-Device Joint Structured & Context Partitioned Projection Neural Network

The projection $\widetilde{i}_p$ for x_i is then computed by applying a series of T context-partitioned projection functions $\widetilde{\mathbb{P}}^1, ..., \widetilde{\mathbb{P}}^T$ on the intermediate sparse feature vector $\vec{x_i}$. Details of the projections and computation for SGNN++ are described as follows.

$$\widetilde{\mathbb{P}}^j(x_i) = \texttt{projection}(\vec{x_i}, \widetilde{\mathbb{P}}^j) \quad (2)$$

$$\widetilde{i}_p = \widetilde{\mathbb{P}}^{1...T}(x_i) = [\,\widetilde{\mathbb{P}}^1(x_i), ..., \widetilde{\mathbb{P}}^T(x_i)\,] \quad (3)$$

where $\widetilde{\mathbb{P}}^j(x_i)$ refers to output from the j-th projection function. This is followed by a stack of additional layers and non-linear activation to create deep, non-linear combinations of projections that permit the network to learn complex mappings from inputs x_i to outputs y_i.

$$\widetilde{h}_p = \sigma(W_p \cdot \widetilde{i}_p + b_p) \quad (4)$$

$$\widetilde{h}_t = \sigma(W_t \cdot \widetilde{h}_{t-1} + b_t) \quad (5)$$

$$y_i = \texttt{softmax}(W_o \cdot \widetilde{h}_k + b_o) \quad (6)$$

where $\widetilde{h}_p$ is computed directly from the projection output, h_t is applied at intermediate layers of the network with depth k followed by a final `softmax` activation layer at the top. In an L-layer SGNN++ , h_t, where $t = p, p+1, ..., p+L-1$ refers to the L subsequent layers after the projection layer. W_p, W_t, W_o and b_p, b_t, b_o represent trainable weights and biases respectively. The projection transformations use pre-computed parameterized functions, i.e., they are not trained during learning, and their outputs are concatenated to form the hidden units for subsequent operations.

2.3 Joint Structured Projection Network

Unlike prior work that employs projections (Ravi and Kozareva, 2018), we make an important observation that input instances x_i are drawn from natural language rather than random continuous vectors and thereby encode some inherent structure—for example, sentences contain sequence of words, and words contain characters. This motivates us to leverage the underlying linguistic structure in the input and build a hierarchical projection model from the raw text in a *progressive* fashion rather than taking a one-shot projection approach. We define a joint structured projection model (SGNN++). The model jointly combines *word* and *character* level context information from the input text to construct the language projection layer.

2.3.1 Word Projections

Given an input x_i with t words, we first project sequence x_i to word projection vectors. We use word-level context features (e.g., phrases and word-level skip-grams) extracted from the raw text to compute the intermediate feature vector $\vec{x}_w = \mathbb{F}_w$ and compute projections.

$$\widetilde{\mathbb{P}}^j_w(x_i) = \texttt{projection}(\vec{x_{i_w}}, \widetilde{\mathbb{P}}^j_w) \quad (7)$$

$$\widetilde{i}_{p_w} = \widetilde{\mathbb{P}}^{1...\ell}_w(x_{i_w}) = [\,\widetilde{\mathbb{P}}^1_w(x_i), ..., \widetilde{\mathbb{P}}^\ell_w(x_{i_w})\,] \quad (8)$$

We reserve ℓ bits to capture the word projection space computed using a series of ℓ functions $\widetilde{\mathbb{P}}^1_w, ..., \widetilde{\mathbb{P}}^\ell_w$. The functions project the sentence structure into low-dimensional representation that

captures similarity in the word-projection space (Sankar et al., 2019).

2.3.2 Character Projections

Given the input text x_i, we can capture morphology (character-level) information in a similar way. We use character-level context features (e.g., character-level skip-grams) again extracted directly from the raw text to compute $\vec{x_c} = \mathbb{F}_c$ and compute character projections $\widetilde{i}_{p_c}$.

$$\widetilde{\mathbb{P}}_c^j(x_i) = \texttt{projection}(\vec{x_{i_c}}, \widetilde{\mathbb{P}}_c^j) \quad (9)$$

$$\widetilde{i}_{p_c} = \widetilde{\mathbb{P}}_c^{\ell+1...T}(x_{i_c}) \quad (10)$$
$$= [\,\widetilde{\mathbb{P}}_c^{\ell+1}(x_i), ..., \widetilde{\mathbb{P}}_c^{T}(x_{i_w})\,]$$

The character feature space and hence projections $\widetilde{i}_{p_c}$ are reserved and computed separately. Note that even though we compute separate projections for character-level context, the SGNN++ model re-uses the remaining $T - \ell$ functions for this step and hence keeps the overall space and time complexity for projections directly $\propto T$.

2.3.3 Joint Structured Model and Extension

We then combine these into $\widetilde{i}_p$ for the joint structure projection model as shown in Figure 1. The projection functions dynamically transform each input text x_i to a low-dimensional representation i_p via context-dependent projection spaces that jointly capture word and character information in a succinct representation. The joint structured projections are followed by a stack of additional layers that jointly learn non-linear combinations of these projection spaces to build the classifier.

$$\widetilde{h}_p = \sigma(W_p \cdot [\widetilde{i}_{p_w}, \widetilde{i}_{p_c}] + b_p) \quad (11)$$

The choice of intermediate features used for projections can be flexible and different for $\mathbb{F}_w$ and $\mathbb{F}_c$. For example, we could apply stemming or extract other morphological features for computing $\widetilde{i}_{p_c}$. Similarly, we can use syntax information from Part-of-Speech tags or constituency parses at the sentence-level for computing $\widetilde{i}_{p_w}$. However, these features might not be available on device to perform inference—e.g., syntax features require an additional tagging or parsing model to be loaded on device, which incurs additional complexity and latency. Hence, for efficiency and simplicity, we only use the same type of raw features (e.g., skip-grams) for word and character-level projections.

2.4 Context Partitioned Projection Network

In the SGNN++ model, we further leverage the feature-context *type* information to introduce an additional level of hierarchy in the network. The motivation is as follows—we use locality-sensitive projections for `projection(.)` step to transform input text to a low-dimensional representation. Incorporating global information, via context-dependent projections, enables the model to vary the language projections and encode them separately based on feature-type. We use this to avoid collisions in the projected space between different feature types (e.g., $unigrams$ vs. $bigrams$) and also help the neural network learn the importance of specific types of projections based on the classification task rather than pooling them together and fixing this apriori.

We achieve this by introducing *context-partitioned* projections in SGNN++ , i.e., we partition the overall projection space into sub-partitions based on context-type. Let C_K denote the type of intermediate features extracted via $\mathbb{F}$, where $C_1 = unigrams$, $C_2 = bigrams$, and so on. Both word and character-level outputs i_{p_w}, i_{p_c} (describe earlier) are generated using context-partitioned projections, i.e., each projection space $\Omega_{\widetilde{\mathbb{P}}}$ is partitioned into sub-spaces $\Omega_{\widetilde{\mathbb{P}}_{C_k}}$ based on context type. The *type* of context used to represent the input text determines the *function choice* and *size* of the sub-partitions and thereby the number of corresponding bits reserved in the projection outputs i_{p_w} and i_{p_c}.

$$\widetilde{i}_p = [\,\widetilde{\mathbb{P}}_{C_1}^1(x_i), ..., \widetilde{\mathbb{P}}_{C_1}^{\ell_1}(x_i)\,] \quad (12)$$
$$\| [\,\widetilde{\mathbb{P}}_{C_2}^1(x_i), ..., \widetilde{\mathbb{P}}_{C_2}^{\ell_2}(x_i)\,]$$
$$...$$
$$\| [\,\widetilde{\mathbb{P}}_{C_K}^1(x_i), ..., \widetilde{\mathbb{P}}_{C_K}^{\ell_K}(x_i)\,]$$

$$M = \frac{\max_K \cdot (\max_K + 1)}{2} \quad (13)$$

$$\ell_K = T \cdot \frac{K}{M} \quad (14)$$

where, C_K denotes a specific type of context-feature extracted from the input and $\mathbb{P}_{C_K}^1 ... \mathbb{P}_{C_K}^{\ell_K}$ denote the projection functions applied to the input for context type C_K. $\max_K$ is the total number of context types and ℓ_K is the number of projection functions in the partition reserved for C_K and hence the number of output bits reserved in projection output.

Effect of Partitioned Projections: Partitioning the projection space has a significant effect on both memory and time-complexity. This results in a significant speedup for the projection network both during training and inference since the overall size of intermediate feature context vectors $\mathbb{F}$ (per type) is smaller and hence fewer operations are required to compute each projection output and these can be computed in parallel. Also, in SGNN++ the overall projection complexity does not increase since we keep T fixed $\sum_{j_{w,c}} \ell_j = T$. Moreover, the context partitioned SGNN++ neural network uses the global context information to efficiently decompose and learn projections from different contexts and combine them effectively for the classification task.

2.5 q-SGNN++ : Compressing Model Further

We also learn hardware-optimized variants of SGNN++ using **quantized** training similar to (Jacob et al., 2017). This permits fast 8-bit arithmetic operations in the model achieving 4x further reduction in overall model size and improved latency. Both SGNN++ and q-SGNN++ can run efficiently on edge devices and support inference through **TensorFlow Lite** (tfl) open-source library.

2.6 Computing Projections on-the-fly

We employ an efficient randomized projection method for each `projection(.)` step. We use locality sensitive hashing (LSH) (Charikar, 2002) to model the underlying projection operations in SGNN++ . Equation 1 applies $\mathbb{F}$ to dynamically extract features from the raw input text. Text features (e.g., skip-grams) at word and character-level are converted into 64-bit feature-ids f_j (via hashing) to generate a sparse feature representation $\vec{x_i}$ of feature-id, weight pairs (f_m, w_m). For the `projection(.)` step (Equation 4), a projection vector $\widetilde{P}^j$ is first constructed on-the-fly using a hash function with feature ids $f_m \in \vec{x_i}$ and fixed seed j as input, then dot product of the two vectors $< \vec{x_i}, \widetilde{\mathbb{P}}^j >$ is computed and transformed into binary representation $\widetilde{\mathbb{P}}^j(\vec{x_i})$ using $sgn(.)$ of the dot product.

As shown in Figure 1, both $\mathbb{F}_{w,c}$ and $\widetilde{\mathbb{P}}_{w,c}$ steps are computed on-the-fly, i.e., no word/character-embedding or vocabulary/feature matrices need to be stored and looked up during training or inference. Instead feature-ids and projection vectors are dynamically computed via hash functions. For intermediate feature weights w_m, we use observed counts in each input text and do not use precomputed statistics like idf. Hence the method is embedding-free.

2.7 Model Parameters

SGNN++ uses a total of T different projection functions $\widetilde{\mathbb{P}}^{j=1...T}$, each resulting in d-bit vector that is concatenated to form the projected vector i_p in Equations 11. T and d can be tuned to trade-off between prediction quality and model size of the SGNN++ network. For the intermediate feature step $\mathbb{F}$ in Equations 1, 9, 11, we use skip-gram features (3-grams with skip-size=2) extracted from raw text both for word and character projections. We set $\ell = \frac{T}{2}$ in Equation 9, i.e., the joint structured model (described in Section 2.3) reserves half the projection space ($\frac{T}{2} \cdot d$ bits) for word projections and remaining half for character projections. The choice of features also determines the size of the context-dependent sub-partitions within each projection space—for example, if we choose features with upto 3-gram context, then $max_K = 3$ and we compute 3 projection sub-partitions for C_1, C_2, C_3 in Equation 14.

2.8 Training, Inference and Optimization

SGNN++ is trained from scratch on the task data using a supervised loss defined wrt ground truth $\hat{y}_i$ $\mathcal{L}(.) = \sum_{i \in N} \texttt{cross} - \texttt{entropy}(y_i, \hat{y}_i)$. During training, the network learns to choose and combine context-dependent projection operations that are more predictive for a given task. SGNN++ uses language projections to transform the input into compact bit vectors. This yields a drastically lower memory footprint both in terms of number and size of parameters as well as computation cost.

During training, the network learns to move the gradients for points that are nearby to each other in the projected bit space $\Omega_{\widetilde{\mathbb{P}}}$ in the same direction. SGNN++ is trained end-to-end using backpropagation. Training can progress efficiently with stochastic gradient descent with distributed computing on high-performance CPUs or GPUs.

2.9 Complexity

Overall complexity for inference with the SGNN++ model depends on the projection layer, $O(n \cdot T \cdot d)$ where n is the observed feature size (*not* overall vocabulary size) which is linear in input size, d is the number of LSH

bits specified for each projection vector $\widetilde{\mathbb{P}}^j$, and T is the number of projection functions used. However, each partitioned projection operation in the model is much faster in practice than non-partitioned projection since it depends on size of intermediate vectors which are partitioned by context and smaller in size. The model size (in terms of number of parameters) and memory storage required for the *projection* inference step is $O(T \cdot d \cdot C)$, where C is the number of hidden units in $\widetilde{h}_p$ in the multi-layer projection network and typically smaller than $T \cdot d$.

3 NLP Datasets and Experimental Setup

3.1 Datasets & Tasks

We evaluate our on-device SGNN++ model on four NLP tasks and languages such as English, Japanese, Spanish and French. The datasets were selected so we can compare against prior on-device work (Ravi and Kozareva, 2018) and also test the language agnostic capabilities of SGNN++

- **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).
- **SwDA: Switchboard Dialog Act** is a popular open domain dialog corpus between two speakers with 42 dialog acts (Godfrey et al., 1992; Jurafsky et al., 1997).
- **ATIS: Intent Understanding** is a widely used corpus in the speech and dialog community (Tür et al., 2010) for understanding different intents during flight reservation.
- **CF: Customer Feedback** is a multilingual customer feedback analysis task (Liu et al., 2017) that aims at categorizing customer feedback as "*comment*, "*request*, "*bug*, "*complaint*, "*meaningless*, or "*undetermined*. The data is in English (EN), Japanese (JP), French (FR) and Spanish (SP) languages.

Table 1 shows the characteristics of each task: language, number of classes, training and test data.

3.2 Experimental Setup & Parameter Tuning

We setup our experiments as given a classification task and a dataset, generate an on-device model. For each task, we report *Accuracy* on the test set.

NLP Task		Lang.	#Classes	Train	Test
MRDA	Dialog Act	EN	6	78K	15K
SwDA	Dialog Act	EN	42	193K	5K
ATIS	Intent Prediction	EN	21	4,478	893
CF-EN	Cust. Feedback	EN	5	3,065	500
CF-JP	Cust. Feedback	JP	5	1,526	300
CF-FR	Cust. Feedback	FR	5	1,950	400
CF-SP	Cust. Feedback	SP	5	1,631	299

Table 1: NLP Tasks and Datasets Statistics

Unlike prior work that aims at finding the best configuration for a given datasets or task, we use the same on-device architecture and settings across all datasets and tasks. We use 2-layer SGNN++ ($\mathbb{P}_{T=80,d=14}$ × FullyConnected$_{256}$ × FullyConnected$_{256}$), mini-batch size of 100, dropout rate of 0.25, learning rate initialized to 0.025 with cosine annealing decay (Loshchilov and Hutter, 2016). We do not do any additional dataset-specific tuning or processing. Training is with SGD over shuffled mini-batches with Adam optimizer (Kingma and Ba, 2014).

4 Experimental Results

This section focuses on the multiple experiments we have conducted. Table 2 shows the results on the different NLP tasks and languages. Overall, SGNN++ consistently outperformed all baselines, reached state-of-the-art against prior on-device state-of-the-art work (Ravi and Kozareva, 2018) and even outperformed non-on-device state-of-the-art RNN, CNN and BiLSTM models for MRDA, SWDA, ATIS and CF tasks.

4.1 Comparison with Baselines

For each task, we compared SGNN++ against well established baselines. MRDA and SWDA use Naive Bayes classifier (Lee and Dernoncourt, 2016), which our SGNN++ model outperformed with 14 to 41%. ATIS uses a majority baseline, which SGNN++ outperformed with 21.51%. CF (Liu et al., 2017) uses trigrams to find the most similar annotated sentences to the input and assigns their label as final prediction. SGNN++ consistently outperformed CF similarity baselines with 16.2%, 17.66%, 16.18 and 6.69% for EN, JP, FR and SP respectively.

4.2 Comparison with On-Device State-of-Art

One of the most important studies in this work is the comparison of our on-device model against prior state-of-the-art on-device NLP model called self-governing neural networks (SGNN) (Ravi and

Model	MRDA	SwDA	ATIS	CF-EN	CF-JP	CF-FR	CF-SP
SGNN++ (our on-device)	**87.30**	**88.43**	**93.73**	65.00	**74.33**	70.93	83.95
SGNN(Ravi and Kozareva, 2018)(sota on-device)	86.70	83.10	-	-	-	-	-
RNN(Khanpour et al., 2016)	86.80	80.10	-	-	-	-	-
RNN+Attention(Ortega and Vu, 2017)	84.30	73.90	-	-	-	-	-
CNN(Lee and Dernoncourt, 2016)	84.60	73.10	-	-	-	-	-
GatedAtten.(Goo et al., 2018)	-	-	93.60	-	-	-	-
JointBiLSTM(Hakkani-Tur et al., 2016)	-	-	92.60	-	-	-	
Atten.RNN(Liu and Lane, 2016)	-	-	91.10	-	-	-	-
ADAPT-Run1(Dzendzik et al., 2017)	-	-	-	63.40	67.67	69.50	83.61
Bingo-logistic-reg(Elfardy et al., 2017)	-	-	-	55.80	60.67	59.00	72.91
Baseline	74.60	47.30	72.22	48.80	56.67	54.75	77.26

Table 2: On-device Results and Comparison on Multiple Datasets and Languages

Kozareva, 2018). SGNN learns compact projection vectors with local sensitive hashing and has previously reached state-of-the-art results on MRDA and SWDA tasks. While both methods share the ideology of projections, SGNN++ uses more powerful representations via joint structured and context partitioned projections. As shown in Table 2, SGNN++ outperformed SGNN with 1% on MRDA and 5% on SWDA. These significant performance improvements are due to SGNN++ 's joint structure representations coupled with partitioned projections. Section 5.1 shows detailed ablation study.

4.3 Comparison with Non-On-Device Work

The characteristics of on-device models are low memory footprint and low latency. Therefore, a direct comparison of an on-device model against cloud based neural networks might not be fair, due to the resource constraints for on-device models. But we wanted to showcase that despite such constraints, yet our SGNN++ learns powerful neural networks that are competitive and can even outperform widely used approaches like RNNs and CNNs with huge parameters and pre-trained word embeddings. Another aspect to consider on why such a comparison might not be fair, is that prior work focused mostly on creating the best model for a specific task with lot of fine tuning and additional resources like pre-trained embedding, whereas we use the same SGNN++ architecture and parameters across multiple tasks and languages.

Taking these major differences into consideration, we still compare results against prior non-on-device state-of-art neural networks. As shown in Table 2 only (Khanpour et al., 2016; Ortega and Vu, 2017; Lee and Dernoncourt, 2016) have evaluated on more than one task, while the rest of the methods target specific one. We denote with – models that do not have results for the task. SGNN++ is the only approach spanning across multiple NLP tasks and languages.

On the **Dialog Act** MRDA and SWDA tasks, SGNN++ outperformed deep learning methods like CNN (Lee and Dernoncourt, 2016), RNN (Khanpour et al., 2016) and RNN with gated attention (Tran et al., 2017) and reached the best results of 87.3% and 88.43% accuracy.

For **Intent Prediction**, SGNN++ also improved with 0.13% 1.13% and 2.63% over the gated attention (Goo et al., 2018), the joint slot and intent biLSTM model (Hakkani-Tur et al., 2016) and the attention slot and intent RNN (Liu and Lane, 2016) on the ATIS task. This is very significant, given that (Goo et al., 2018; Hakkani-Tur et al., 2016; Liu and Lane, 2016) used a joint model to learn the slot entities and types, and used this information to better guide the intent prediction, while SGNN++ does not have any additional information about slots, entities and entity types.

On **Customer Feedback**, SGNN++ reached better performance than Logistic regression models (Elfardy et al., 2017; Dzendzik et al., 2017).

Overall, SGNN++ achieves impressive results given the small memory footprint and the fact that it did not rely on pre-trained word embeddings like (Hakkani-Tur et al., 2016; Liu and Lane, 2016) and used the same architecture and model parameters across all tasks and languages. We believe that the dimensionality-reduction techniques like locality sensitive context projections jointly coupled with deep, non-linear functions are effective at dynamically capturing low dimensional semantic text representations that are useful for text classification applications.

5 Ablation Studies

In this section, we show multiple ablation studies focusing on: (1) impact of partitioned projections and joint structured representation on accuracy; (2) impact of model size on accuracy; quantized version of SGNN++ which reduces model size while preserving same quality; and (3) SGNN++ latency.

5.1 Impact of Joint Structured & Context Partitioned Projections on Accuracy

Our SGNN++ model uses joint structured (word+character) and context partitioned projections. We want to show the impact of the joint structure (word+character) vs word only; as well as the impact of the partitioned vs non-partitioned projections. Table 3 shows the obtained results on the ATIS intent prediction dataset. First, using joint structured (word+character) information leads to significantly better performance compared to word only. For instance, +9% for non-partitioned projections and +3.9% for partitioned projections. Second, significant improvement is seen when using partitioned vs non-partitioned projections, +6.14% for word and +1% for word+character. Overall, the novel joint structured and context partitioned projections we introduced in our SGNN++ model improve +10.06% performance compared to models using only word and non-partitioned projections.

ATIS	**Partitioned SGNN++**	**Non-Partitioned SGNN++**
Word+Char	**93.73**	92.72
Word	89.81	83.67

Table 3: Impact of SGNN++ Partitioning on Accuracy

It is important to note that in addition to the accuracy improvements, SGNN++ partitioned projection models are also significantly faster for inference and training (upto $3.3X$). For example, using $T = 80, d = 14$ and bigram word features ($max_K = 2$) for a 10-word sequence requires $80 \times 14 \times 6 = 6720$ `multiply-add` operations for partitioned projections compared to $80 \times 14 \times 19 = 21280$ for non-partitioned model.

5.2 Accuracy vs Model Size

It is easy to customize our model for different devices such as watches, phones or IoT with different size constraints. To showcase this, we show results on varying projection sizes and network parameters. Furthermore, we also trained quantized versions of our SGNN++ model denoted by qSGNN++ which achieves additional model size reduction while maintaining high accuracy. Figure 2 shows the obtained results on the ATIS dataset. Each data point in the figure represents a SGNN++ or qSGNN++ model trained with specific partition projection parameter configuration. We show the model size and the accuracy achieved for that size.

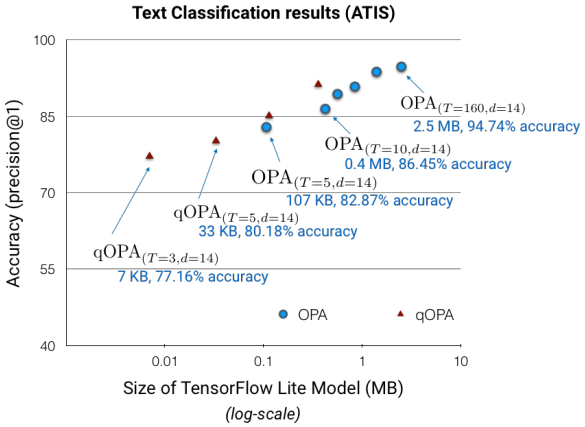

Figure 2: Model Size vs. Accuracy

Overall, SGNN++ models achieve high accuracy even at low sizes. For instance, 100KB model yields 82.87% accuracy compared to 2.5MB model that yields 94.74%. For a given SGNN++ model we can further reduce the size with little performance degradation by applying the quantization-aware training. For instance, SGNN++ 107KB model ($T = 5, d = 14$) yields 82.87%, but can be further compressed to qSGNN++ with 33KB and 80.18% accuracy. We also take our model to the extreme, we are able to train qSGNN++ model with extremely tiny size of 7KB ($T = 3, d = 14$), while still achieving 77.16%.

5.3 Model Latency

In addition to being small and highly accurate, on-device model has to be fast. We measure the latency of our on-device SGNN++ model on a Pixel phone. Given an input text, we measure inference time on the Pixel device and report average latency. On ATIS dataset, SGNN++ accuracy is 93.73% with average latency of 3.35 milliseconds. This shows that our SGNN++ model is compact, highly accurate and with low latency (i.e. very fast).

6 Conclusion

We proposed embedding-free on-device neural network that uses joint structured and context partitioned projections for short text classification. We conducted experiments on wide range of NLP applications such as dialog acts, intent prediction and customer feedback. We evaluated the approach on four languages, showing the language agnostic capability of our on-device SGNN++ model. We used the same model architecture and parameter settings across all languages and tasks, which demonstrates the generalizability of this approach compared to prior work that built custom models. Overall, our SGNN++ approach outperformed all baselines from 14 to 41%, improved upon state-of-the-art on-device work (Ravi and Kozareva, 2018) with up to 5%, and also outperformed non-on-device neural approaches (Hakkani-Tur et al., 2016; Liu and Lane, 2016; Dzendzik et al., 2017; Elfardy et al., 2017). Through multiple ablation studies, we showed the impact of partitioned projections on accuracy and the impact of model size on accuracy. We trained quantized versions of SGNN++ showing that we can further reduce the model size while preserving quality. Finally, we showed SGNN++ fast latency on Pixel phone.

Acknowledgments

We would like to thank the organizers of the customer feedback challenging for sharing the data and the anonymous reviewers for their valuable feedback and suggestions.

References

TensorFlow Lite. https://www.tensorflow.org/lite/.

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.

Moses S. Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 380–388, New York, NY, USA. ACM.

Daria Dzendzik, Alberto Poncelas, Carl Vogel, and Qun Liu. 2017. Adapt centre cone team at ijcnlp-2017 task 5: A similarity-based logistic regression approach to multi-choice question answering in an examinations shared task. In *Proceedings of the IJCNLP 2017, Shared Tasks*, pages 67–72. Asian Federation of Natural Language Processing.

Heba Elfardy, Manisha Srivastava, Wei Xiao, Jared Kramer, and Tarun Agarwal. 2017. Bingo at ijcnlp-2017 task 4: Augmenting data using machine translation for cross-linguistic customer feedback classification. In *Proceedings of the IJCNLP 2017, Shared Tasks*, pages 59–66. Asian Federation of Natural Language Processing.

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, NAACL-HLT*, pages 753–757.

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)*.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew G. Howard, Hartwig Adam, and Dmitry Kalenichenko. 2017. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *CoRR*, abs/1712.05877.

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.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105.

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.

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)*.

Chao-Hong Liu, Yasufumi Moriya, Alberto Poncelas, and Declan Groves. 2017. Ijcnlp-2017 task 4: Customer feedback analysis. In *Proceedings of the IJCNLP 2017, Shared Tasks*, pages 26–33. Asian Federation of Natural Language Processing.

Ilya Loshchilov and Frank Hutter. 2016. SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983.

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 International Conference on Machine Learning (ICML 2019)*.

Sujith Ravi and Zornitsa Kozareva. 2018. Self-governing neural networks for on-device short text classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, (EMNLP 2018)*.

Chinnadhurai Sankar, Sujith Ravi, and Zornitsa Kozareva. 2019. Transferable neural projection representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2019)*.

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.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112.

Quan Hung Tran, Gholamreza Haffari, and Ingrid Zukerman. 2017. A generative attentional neural network model for dialogue act classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 524–529.

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.