End to End Binarized Neural Networks for Text Classification

Kumar Shridhar[1]*, **Harshil Jain**[2]*, **Akshat Agarwal**[3]*, **Denis Kleyko**[4,5]
[1]NeuralSpace, London
[2]Computer Science and Engineering, IIT Gandhinagar, Gujarat, India
[3]Electrical Engineering, Delhi Technological University, Delhi, India
[4]Redwood Center for Theoretical Neuroscience, University of California, Berkeley
[5]Intelligent Systems Lab, Research Institutes of Sweden
kumar@neuralspace.ai, jain.harshil@iitgn.ac.in,
akshat.agarwal0311@gmail.com, denis.kleyko@ri.se

Abstract

Deep neural networks have demonstrated their superior performance in almost every Natural Language Processing task, however, their increasing complexity raises concerns. A particular concern is that these networks pose high requirements for computing hardware and training budgets. The state-of-the-art transformer models are a vivid example. Simplifying the computations performed by a network is one way of addressing the issue of the increasing complexity. In this paper, we propose an end to end binarized neural network for the task of intent and text classification. In order to fully utilize the potential of end to end binarization, both the input representations (vector embeddings of tokens statistics) and the classifier are binarized. We demonstrate the efficiency of such a network on the intent classification of short texts over three datasets and text classification with a larger dataset. On the considered datasets, the proposed network achieves comparable to the state-of-the-art results while utilizing $\sim$ 20-40% lesser memory and training time compared to the benchmarks.

1 Introduction

In recent years, deep neural networks have achieved great success in a variety of domains, but the networks are becoming more and more computationally expensive due to their ever-growing size. This tendency has been noticed in (Strubell et al., 2019; Schwartz et al., 2019) and it has been recommended that academia and industry researchers should draw their attention towards more computationally efficient methods. At the same time, many important application areas such as chatbots, IoT devices, mobile devices, and other types of power-constrained and resource-constrained platforms require solutions that would be highly computationally and memory efficient. Such use-cases limit the potential use of the state-of-the-art deep networks. One viable solution is the transformation of these high-performance neural networks to a more computationally efficient architecture. Recently, Binarized Convolutional Neural Networks (BNN) (Hubara et al., 2016) have been developed where both weights and activations are restricted to $\{+1, -1\}$. BNN is a highly computationally efficient network with a much lower memory footprint. Tasks like language modeling (Zheng and Tang, 2016) were performed using binarized neural networks, but, to the best of our knowledge, in the area of text classification, no end to end trainable binarized architectures have been demonstrated yet.

In this paper, we introduce an architecture for the tasks of intent and text classifications that fully utilizes the power of binary representations. The input representations are tokenized and embedded in binary high-dimensional (HD) vectors forming distributed representations using the paradigm known as hyperdimensional computing (Kanerva, 2009). The binary input representations are used for training an end to end BNN classifier for intent classification. Classification performance-wise, the binarized architecture achieves results comparable to the state-of-the-art on several standard intent classification datasets. The efficiency of the proposed architecture is shown in terms of its time and memory complexity relative to non-binarized architectures.

2 Proposed Method

Figure 1 presents a schematic overview of the architecture. Given an input text document D, we first pre-process the document. The pre-processed document is then tokenized into the corresponding tokens $< T_1, T_2, ..., T_n >$, which are used as an

* The authors contributed equally to this research and work was done at NeuralSpace

Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing, pages 29–34
Online, November 20, 2020.

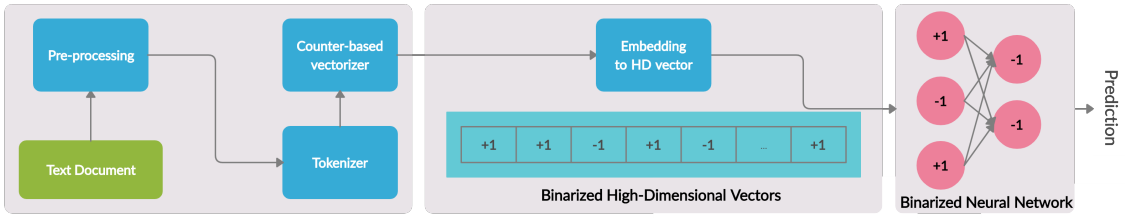


Figure 1: A schematic diagram of the end to end binarized classification architecture for text classification.

input to a count-based vectorizer. The representation of vectorizers, which is sparse and localist, is embedded into an HD vector (distributed representation) using hyperdimensional computing. HD vector representing the counter's content can be binarized. It is used as an input to a classifier. The primary classifier studied in this work is BNN, but other classifiers are also considered for benchmarking.

2.1 High-Dimensional embedding of vectorized representations

In order to reduce the dimensionality of representations, we use hyperdimensional computing (Kanerva, 2009). First, each unique token T_i is assigned with a random d-dimensional bipolar HD vector, where d would be a hyperparameter of the method. HD vectors are stored in the item memory, which is a matrix $\mathbf{H} \in [d \times n]$, where n is the number of tokens. Thus, for a token T_i there is an HD vector $\mathbf{H}_{T_i} \in \{-1, +1\}^{[d \times 1]}$. To construct composite representations from the atomic HD vectors stored in $\mathbf{H}$, hyperdimensional computing defines three key operations: permutation (ρ), binding ($\odot$, implemented via element-wise multiplication), and bundling ($+$, implemented via element-wise addition) (Kanerva, 2009). The bundling operation allows storing information in HD vectors (Frady et al., 2018). The three operations above allow embedding vectorized representations based on n-gram statistics into an HD vector (Joshi et al., 2016).

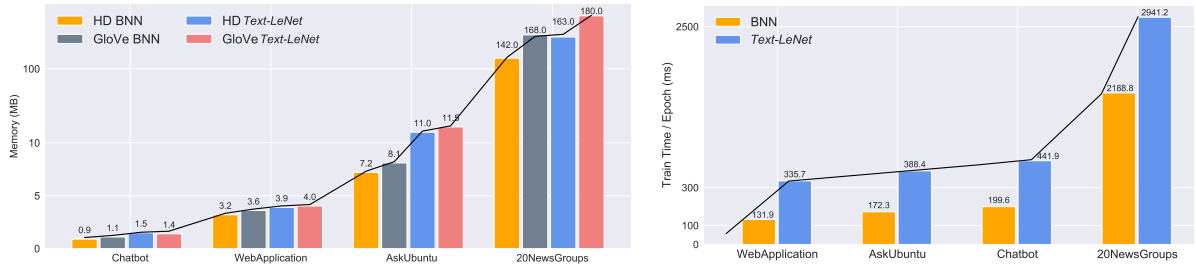
We first generate $\mathbf{H}$, which has an HD vector for each token. The permutation operation ρ is applied to $\mathbf{H}_{T_j}$ j times ($\rho^j(\mathbf{H}_{T_j})$) to represent a relative position of token T_j in an n-gram. A single HD vector corresponding to an n-gram (denoted as $\mathbf{m}$) is formed using the consecutive binding of permuted HD vectors $\rho^j(\mathbf{H}_{T_j})$ representing tokens in each position j of the n-gram. For example, the trigram '#he' will be embedded to an HD vector as follows: $\rho^1(\mathbf{H}_\#) \odot \rho^2(\mathbf{H}_\mathrm{h}) \odot \rho^3(\mathbf{H}_\mathrm{e})$. In general, the process of forming HD vector of an n-gram is $\mathbf{m} = \prod_{j=1}^{n} \rho^j(H_{T_j})$, where T_j is token in jth position of the n-gram; the consecutive binding operations applied to n HD vectors are denoted by $\prod$. Once it is known how to form an HD vector for an individual n-gram, embedding the n-gram statistics into an HD vector $\mathbf{h}$ is achieved by bundling together all n-grams observed in the document:

$$\mathbf{h} = [\sum_{i=1}^{k} f_i \mathbf{m}_i = \sum_{i=1}^{k} f_i \prod_{j=1}^{n} \rho^j(H_{T_j})],$$

where k is the total number of unique n-grams; f_i is the frequency of ith n-gram and $\mathbf{m}_i$ is the HD vector of ith n-gram; $\sum$ denotes the bundling operation when applied to several HD vectors; $[*]$ denotes the binarization operation, which is implemented via the sign function. The usage of $[*]$ is optional, so we can either obtain binarized or non-binarized $\mathbf{h}$. If $\mathbf{h}$ is non-binarized, its components will be integers in the range $[-k, k]$, but these extreme values are highly unlikely since HD vectors for different n-grams are quasi-orthogonal, which means that in the simplest (but not practical) case when all n-grams have the same probability the expected value of a component in $\mathbf{h}$ is 0. Due to the use of $\sum$ for representing n-gram statistics, two HD vectors embedding two different n-gram statistics might have very different amplitudes if the frequencies in these statistics are very different. When HD vectors $\mathbf{h}$ are binarized, this issue is addressed. In the case of non-binarized HD vectors, we address it by using the cosine similarity, which is imposed by normalizing each $\mathbf{h}$ by its ℓ_2 norm; thus, all $\mathbf{h}$ have the same norm, and their dot product is equivalent to their cosine similarity.

2.2 Binarized Neural Networks

Based on the work of (Hubara et al., 2016), we construct BNNs capable of working with representations of texts. To take the full advantage of binarized HD vectors, we constraint the weights and



(a) Memory Comparison

(b) Train Time per epoch Comparison

Figure 2: (a) shows the memory comparisons for all 4 datasets using HD *Text-LeNet*, HD BNN, GloVe *Text-LeNet*, GloVe BNN and (b) shows the training time per epoch comparison for all 4 datasets using BNN and *Text-LeNet*

	Chatbot		AskUbuntu		WebApplication		20NewsGroups	
Tokenizers	*Text-LeNet*	BNN	*Text-LeNet*	BNN	*Text-LeNet*	BNN	*Text-LeNet*	BNN
Word	**0.80**	0.73	0.51	**0.79**	0.56	**0.78**	0.54	**0.56**
SemHash	**0.94**	0.90	**0.87**	0.84	0.79	**0.83**	**0.78**	0.69
BPE	**0.80**	0.58	0.54	**0.67**	0.52	**0.75**	0.38	**0.42**
Char BPE	**0.92**	0.81	**0.76**	**0.76**	**0.55**	0.53	**0.55**	0.48
SentencePiece	0.80	**0.99**	0.70	**0.72**	0.50	**0.70**	0.41	**0.43**
BERT	**0.89**	0.88	**0.72**	0.71	0.70	**0.77**	**0.60**	**0.60**

Table 1: F_1 performance comparison of binarized *Text-LeNet* (BNN) architecture with non-binarized *Text-LeNet* for the task of intent classification on various datasets.

activations of the network layers to be $\{+1, -1\}$. This constraint is highly efficient in terms of hardware and memory, as bit-wise operations are used instead of multiply-accumulate operations. For example, a multiplication on binary values can be performed using an XNOR logical operation.

The vectorized representations of tokens embedded into HD vectors are binarized with all values $\{+1, -1\}$. In the case of HD vectors, we binarize the result of the bundling operation using the sign function.

Similarly, the sign function is used in the BNN for every weight or activation to restrict them into $\{+1, -1\}$ as follows:

$$b(x) = [x] = \text{sign}(x) = \begin{cases} +1 & \text{if } x \geq 0, \\ -1 & \text{otherwise} \end{cases} \quad (1)$$

where, x can be any weight or activation value.

We further define a convolutional 1D layer that creates a convolution kernel that is convolved with the input HD vector over a single spatial dimension to produce a tensor of outputs. Since gradient descent methods make small changes to the value of the weights, which cannot be done with binary values, we use the straight-through estimator idea, as mentioned in (Yin et al., 2019). We also define a value over which we clip the gradients in the backward pass:

$$\frac{\delta b(x)}{\delta x} = \begin{cases} +1 & \text{if } |x| < \text{clip value}, \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

This ensures that the entire architecture is end to end trainable using gradient descent optimization.

3 Empirical Analysis

3.1 Datasets

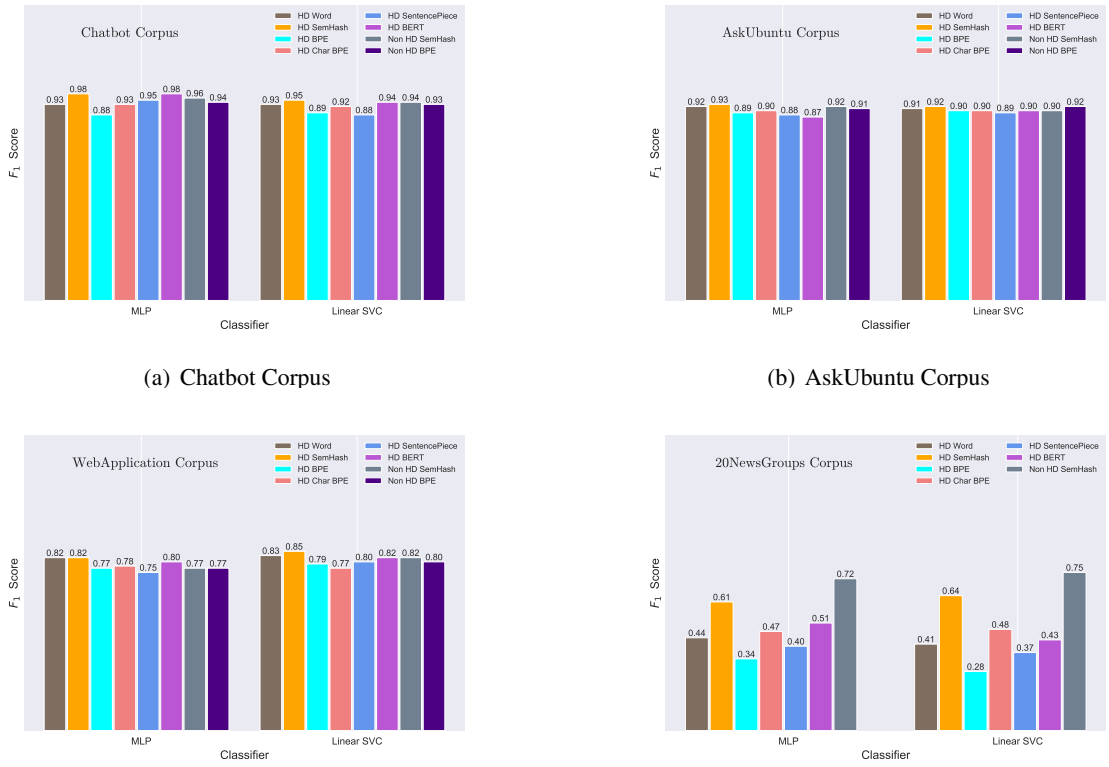
All the experiments are performed on four datasets, namely: the *Chatbot Corpus* (Chatbot), the *Ask Ubuntu Corpus* (AskUbuntu), the *Web Applications Corpus* (WebApplication), and the *20 News Groups Corpus* (20NewsGroups) (Braun et al., 2017).

3.2 Results and Discussions

For CNN-based architecture, 5 hidden layers were used: 3 convolutional 1D layers followed by 2 dense layers. Due to its resemblance to the original LeNet architecture (LeCun et al., 1998), we refer to this architecture as *Text-LeNet*. We compare the results of binarized HD vectors with the binarized *Text-LeNet* (BNN) architecture as the classifier against non-binarized HD vectors with non-binarized *Text-LeNet*. The F_1 scores are compared in Table 1 where BNN performed equally well to a *Text-LeNet* architecture while being 20%

Datasets	Binarized GloVe	Binarized SemHash	Binarized HD vectors
Chatbot	0.74	0.91	**0.99**
AskUbuntu	0.86	**0.87**	0.84
WebApplication	0.66	0.80	**0.83**
20NewsGroups	0.62	0.64	**0.69**

Table 2: F_1 performance comparison of Binarized GloVe vectors, Binarized SemHash vectors and Binarized HD vectors. All vectorizers use the same binarized *Text-LeNet* architecture as classifier.



(a) Chatbot Corpus

(b) AskUbuntu Corpus

(c) WebApplication Corpus

(d) 20NewsGroups Corpus

Figure 3: (a), (b), (c) and (d) show the F_1 score comparison of MLP and Linear SVC classifier with HD and non-HD based tokenizers on Chatbot, AskUbuntu, WebApplication and 20NewsGroups corpus respectively.

to 40% more memory efficient, as shown in Figure 2 (a). Note that due to the specifics of implementation, BNNs use 32 bit float values as *Text-LeNet*. The memory efficiency of BNNs can be further improved by 4x when 8-bit representations are used and up to 32x if a single bit representations are used. However, the hardware limitations prevented us from going to that extreme. On the performance side, BNNs outperforms the *Text-LeNet* for AskUbuntu and WebApplication datasets on 4 out of 6 tokenizers. The results reported in Table 1 used 512 dimensional HD vectors for Chatbot, AskUbuntu, and WebApplication corpus, while 1,024 dimensional HD vectors were used for the 20NewsGroups dataset.

One thing to note here is that *Text-LeNet* also used HD vectors with the mentioned tokenizers, but the HD vectors were non-binarized. HD vectors in itself are already faster and much more efficient than counter-based representations, as shown in (Alonso et al., 2020). When experimenting with other embedding methods like GloVe, the training was significantly slower; therefore, HD vectors were used for all the experiments. In addition to that, using the binarized classifier (BNN) further improved the training time up to 50% per epoch when compared to non-binarized classifier on all four datasets, as shown in Figure 2 (b). Furthermore, when compared to GloVe embeddings with *Text-LeNet*, HD BNN used around 20 - 40% lesser

Platform	Chatbot	AskUbuntu	WebApplication	Average
Botfuel	0.98	0.90	0.80	0.89
Luis	0.98	0.90	0.81	0.90
Dialogflow	0.93	0.85	0.80	0.86
Watson	0.97	0.92	0.83	0.91
Rasa	0.98	0.86	0.74	0.86
Snips	0.96	0.83	0.78	0.86
Recast	**0.99**	0.86	0.75	0.87
TildeCNN	**0.99**	0.92	0.81	0.91
FastText	0.97	0.91	0.76	0.88
SemHash (Shridhar et al., 2019)	0.96	0.92	**0.87**	**0.92**
BPE	0.95	**0.93**	0.85	0.91
HD vectors (Alonso et al., 2020)	0.97	0.92	0.82	0.90
Binarized HD vectors with the best classifier	0.98	**0.93**	0.84	**0.92**
HD *Text-LeNet*	0.94	0.87	0.79	0.88
HD BNN	**0.99**	0.84	0.83	0.91

Table 3: F_1 score comparison of various platforms on intent classification datasets of short texts with methods used in the paper. Some results are taken from (Alonso et al., 2020)

memory for all the intent classification datasets.

We also benchmarked the binarized HD vectors with binarized 300-dimensional GloVe vectors and the binarized version of counter-based representation for SemHash tokenizer (Alonso et al., 2020) for all the datasets. Table 2 summarizes the results of the comparison. All the binarized representations were trained with the same BNN classifier. Binarized HD vectors performed significantly better than other binarized methods outperforming binarized GloVe by 4 - 25% and binarized SemHash by 2 - 8% on 2 out of 3 smaller intent classification datasets and achieved comparable results for AskUbuntu dataset. The trend continued for 20NewsGroups with binarized HD achieving 5 - 7% better F_1 scores. Note that for the SemHash counter-based vectorizer, we put a sign function $\text{sign}(x) = +1$ for $x > 0$ and -1 otherwise.

In Figure 3, MLP and Linear SVC with all the tokenizers with HD vectors as representation are compared with MLP and Linear SVC classifiers with SemHash tokenizers and counter-based vectorizer as representation from (Alonso et al., 2020). The F_1 score is comparable to the state-of-the-art for both MLP and SVC. For all small intent classification datasets, binarized HD vectors have achieved better results than non-HD vectors. The proposed architecture beats the non-HD baselines by $+2\%$ for AskUbuntu and Chatbot Corpus, and $+5\%$ for WebApplication Corpus. However, for 20NewsGroups, the results of binarized HD Vectors are lower than non-HD Vectors. This is mainly due to the large size of the dataset, and simple classifiers like LinearSVC failed to perform with just binarized values. The results for all the other classifiers are provided in the Appendix.

Table 3 compares the F_1 scores of various platforms on the intent classification datasets. We report the results of binarized HD vectors with the best classifiers from one of the nine classifiers mentioned (Binarized HD vectors with the best classifier), non-binarized HD vectors with *Text-LeNet* (HD *Text-LeNet*) and binarized HD vectors with binarized *Text-LeNet* (HD BNN). Our end to end binarized architecture (HD BNN) achieved the state-of-the-art results for the Chatbot dataset. The approach where only HD vectors were binarized (binarized HD vectors with the best classifier) achieved the state-of-the-art results for the AskUbuntu dataset. The results on the WebApplication dataset are comparable to the state-of-the-art (0.87 with SemHash): 0.84 for binarized HD vectors with the best classifier and 0.83 for HD BNN. The average performance of both binarized HD vectors with the best classifier (0.92) and HD BNN (0.91) was also comparable to the best non-binarized approach (0.92).

4 Conclusion

In this work, we show that it is possible to achieve comparable to the state-of-the-art results while using the binarized representations of all the components of the text classification architecture. This allows exploring the effectiveness of binary representations both for reducing the memory footprint of the architecture and for increasing the energy-efficiency of the inference phase due to the effectiveness of binary operations. This work takes a step towards enabling NLP functionality on resource-constrained devices.

References

P. Alonso, K. Shridhar, D. Kleyko, E. Osipov, and M. Liwicki. 2020. HyperEmbed: Tradeoffs between Resources and Performance in NLP Tasks with Hyperdimensional Computing Enabled Embedding of n-gram Statistics. *arXiv:2003.01821*.

D. Braun, A. Hernandez-Mendez, F. Matthes, and M. Langen. 2017. Evaluating Natural Language Understanding Services for Conversational Question Answering Systems. In *Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 174–185.

E. P. Frady, D. Kleyko, and F. T. Sommer. 2018. A Theory of Sequence Indexing and Working Memory in Recurrent Neural Networks. *Neural Computation*, 30:1449–1513.

L. Geiger and P. Team. 2020. Larq: An Open-Source Library for Training Binarized Neural Networks. *Journal of Open Source Software*, 5(45):1746.

G. Hinton, N. Srivastava, and K. Swersky. 2012. Neural Networks for Machine Learning Lecture 6a Overview of Mini-batch Gradient Descent.

I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. 2016. Binarized Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1–9.

S. Ioffe and C. Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167.

A. Joshi, J. T. Halseth, and P. Kanerva. 2016. Language Geometry Using Random Indexing. In *Quantum Interaction (QI)*, pages 265–274.

P. Kanerva. 2009. Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors. *Cognitive Computation*, 1(2):139–159.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

R. Schwartz, J. Dodge, N. Smith, and O. Etzioni. 2019. Green ai. *arXiv preprint arXiv:1907.10597*.

K. Shridhar, A. Dash, A. Sahu, G. Grund Pihlgren, P. Alonso, V. Pondenkandath, G. Kovacs, F. Simistira, and M. Liwicki. 2019. Subword Semantic Hashing for Intent Classification on Small Datasets. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–6.

E. Strubell, A. Ganesh, and A. McCallum. 2019. Energy and Policy Considerations for Deep Learning in NLP. In *57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 3645–3650.

P. Yin, J. Lyu, S. Zhang, S. Osher, Y. Qi, and J. Xin. 2019. Understanding Straight-through Estimator in Training Activation Quantized Neural Nets. *arXiv:1903.05662*.

W. Zheng and Y. Tang. 2016. Binarized Neural Networks for Language Modeling. *Technical Report cs224d, Stanford University*.