Supplemental Information

Anonymous Author(s)
Affiliation
Address
`email`

1 Experiments on Network Architecture Search (NAS)

The fusion models for textual classification are initially looked for using NAS, where the primal focus was on identifying the optimal chain-structure length. A fusion depth of eight was maximum in the NAS, and the chain considered CNN, LSTM, BiLSTM as the initial layer. The NAS process considered a few alternative sequences of fusing DNN layers, which are as follows:

1. CNN + LSTM + CNN + LSTM + . . . + CNN
2. LSTM + CNN + LSTM + CNN + . . . + LSTM
3. BiLSTM + CNN + BiLSTM + CNN + . . .+ BiLSTM

The NAS looks for higher validation accuracy and lower validation loss for all the possible fusion models considered under the three subclasses mentioned. For each model structure, we look for optimal chain lengths between 1 and 8, giving each fusion model a minimum of five trials to produce the best parameters. From the six experiments, we collect the best validation loss and validation accuracy for all of the chain-length of the given fusion structure and plot as validation loss vs. validation accuracy (shown in Fig. 1 c, d, and e in the main manuscript). It is worthwhile to note that, given a chain length, the validation loss and validation accuracy do not necessarily come from the same set of parameters. Instead, we present the best possible loss and accuracy among all the chain configurations considered in the NAS. For instance, parameters that produce the highest validation accuracy for a chain length of four for CNN + LSTM + CNN + LSTM fusion structure may not be the best in terms of validation loss for the same fusion model.

1.1 Generalized Random Search

For searching the hyper-parameters of our fusion architecture, we have implemented a random search using Keras. Our goal is to make a general form of a random search function that can be used across all our experiments. The process needs only one parameter to be changed manually: the maximum word length of a sentence is connected with the shape of attention and LSTM layers. Other than this, the function can also be used by others in similar classification tasks. Each layer in the random search is accompanied by an activation layer, a batch normalization layer, and a dropout layer to reduce the overfitting error. The CNN and RNN layers also include kernel, bias, and activity regularizers.

1.2 Data availability

All the relevant data and codes are available in the Github repository at: `https://github.com/smaheen711/Low-Resource-Classifiation-Tasks`

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Table 1: Hyperparameters for the Generalized Random Search.

Hyperparameters	Hyperparameter Space)
Attention Unit	32 – 128; step 16
1st CNN Layer (layer 1) Unit	16 – 96 ; step 16
1st CNN Dropout	0.1 – 0.3; step 0.1
1st LSTM Layer(layer 2) Unit	64 – 256; step 32
1st LSTM Dropout	0.1 – 0.5; step 0.1
2nd CNN Layer(layer 3) Unit	64 – 256; step 32
2nd CNN Dropout	0.1 – 0.5; step 0.1
Cyclic Learning Rate Range	6e-3 – 1e-3
Optimizer	Adam, RMSprop
Epoch	5, 10, 20
Batch Size	32, 64, 100
Loss	Categorical Crossentropy

2 Fusion of various DNN layers

2.1 Word embedding input to CNN

In a chain-structured fusion model, the output of one DNN layer is cascaded down to the next and the cycle continues for the desired fusion depth. Before feeding it to DNN layers, each input sentence is transformed to a word-vector using the pre-trained word embedding fasttext [Joulin et al., 2016]. For a sentence (S) of n tokens $(s_1, s_2, \dots s_n)$, and a maximum sentence length L, fasttext generates an embedding matrix of dimension $W_{L\times d}$ for each sentence. Here, d is the fasttext word-vector size generated, and is fixed at $d = 300$ for all the studied fusion models. One-dimensional CNN layer [Kalchbrenner et al., 2014] extracts position specific local features in a sentence through a sliding filter of size p, and hence, the filter dimension for convolution becomes $\mathbf{u} \in \mathbb{R}^{p\times d}$. Let $\mathbf{x_j} = [r_1, r_2, \dots r_d],\ \forall r_i \in \mathbb{R}, \mathbf{x_j} \in \mathbb{R}^d$, be a d-dimensional word vector for the j-th word in a sentence S. The convolution operation considers a window vector (ω) over every k-th position in a sentence, which comprises of p consecutive word vectors $\mathbf{x}_l, l \in [1 \;\; p-1]$.

The exact window vector, expressed as a vector concatenation, for the k-th window is $\omega_k = [\mathbf{x}_k, \mathbf{x}_{k+1}, \dots, \mathbf{x}_{k+p-1}]$. A feature map $\mathbf{f} = [f_0, f_1, \dots, f_{L-p}]$, where $\mathbf{f} \in \mathbb{R}^{L-p+1}$ forms at each k-th position through convolution between the filter $\mathbf{u}$ and the window vector ω_k. The k-th element of the feature map f_k formulates as element-wise multiplication $f_k = g(\omega_k \odot \mathbf{u} + bias)$, where b is the bias, and $\odot$ denotes the element-wise multiplication. The nonlinear function $g(.)$ could be alternative function types, such as softmax, sigmoid, ReLU, etc. Upon calculation, each feature map $\mathbf{f}$ is subsequently applied to a pooling layer to generate potential features. For instance, the max-pooling of features from the feature map works as $\max f_l$, where $0 < l < L - p$.

2.2 Fusion of CNN and LSTM Layers

Generally, feature representation from the input sentence experiences max-over-pooling or other alternatives for feature extraction, which cascades down later toward the next layer of the CNN. However, when CNN fuses over with an LSTM layer, pooling may affect the sequential nature of the features, and hence, interfacing of CNN with an LSTM layer avoids pooling. So, the max-pooling-layer or the alternative pooling mechanism is absent in CNN when it combines an LSTM layer at its output end. This study uses the Keras implementation in all the studied fusion models to avoid max-pooling-layer at the juncture of CNN and LSTM layers.

2.3 Fusion of CNN and BiLSTM Layers

In this fusion, both word-level and sentence-level features are combined by using CNN output as the input to the forward and backward LSTM network of the BiLSTM architectures. For each word, the CNN layer does the word level feature extraction, which once fed to the BiLSTM layer, the sentence level dependencies of words being evaluated using memory cells available in the forward and backward LSTM network. Generally, a combination of a linear layer and a non-linear function of softmax, log-softmax, sigmoidal decode the LSTM networks output at each time step. Subsequently, the two vectors, evaluated from the forward and backward LSTM network, are summed together for the final output of the BiLSTM layer.

2.4 Metrics for comparison

The initial architectural search uses accuracy measure on the test dataset and the loss difference between the training loss and the validation loss as the two performance metrics. Accuracy is the ratio between the correctly predicted class and the total observations, and formulates as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1}$$

Where TP, TN, FP, FN stands for true positive, true negative, false positive, and false negative. The random search also considers loss difference (LD), along with early stopping, to control the overfitting error, which is defined as follows:

$$\text{LD} = \text{Validation loss} - \text{Training loss} \tag{2}$$

3 Experimental setup for the exploration of a few selected fusion models

3.0.1 Dropout

Different deep learning layers in the fusion architectures consider different dropout rates. For instance, the first four models, which include pruning, have a 0.2 dropout rate in the embedding layer, followed by 0.5 or 0.6 dropouts in the LSTM and CNN layers. Each hidden layer uses different dropouts, as found out, to ensure the lowest overfitting error. The BiLSTM layer considers a recurrent dropout rate of 0.3 in all the fusion architecture studied.

3.0.2 Layer details

The uniform layer size across different layers, as adopted here, makes the implementation of various layers and models simple. Precisely, initial trial runs demonstrate performance improvement for 128-layer units in CNN and LSTM models, whereas the BiLSTM layers contain 64 units. A GlobalMaxPooling layer connects the final output layer of 6-units with the hidden layers in all the models studied.

3.0.3 Batch Size and Epochs

The optimal batch size and epoch number for the CNN layer models are 64 and 20, respectively, as obtained from a rigorous screen. For LSTM and BiLSTM layer models, the epochs are limited to 10 as training errors do not reduce further and become computationally expensive.

3.0.4 Optimizer and activation function

All the models use Adam as the optimizer Kingma and Ba [2014] with cyclic learning rate (CLR) Smith [2017] as the scheduler. The maximum and minimum CLR ranges used do vary to maximize the accuracy and curbing the generalization error. For instance, the CNN + LSTM + CNN layer considers a range of $(0.006, 0.0015)$, whereas the LSTM + CNN + LSTM fusion model uses $(0.006, 0.0015)$ as CLR range. Each model's hidden layers include ReLu as the activation function, and the output layer employs softmax for classification.

3.0.5 Network Pruning and retraining

The polynomial decay with initial and final sparsity at 0.50 and 0.90 respectively perform better among the screen combinations. Retraining of the pruned model considers the same learning rate (known as learning rate rewinding [Renda et al., 2020]) used during training and continues for ten epochs to attain the lost accuracy. by retraining the models for about ten additional epochs.

3.1 Artificial data-scarcity experiment on IMDB dataset

This dataset is generated from single sentenced movie reviews for binary sentiment classification, popularly known as the IMDB dataset. The full dataset has 50000 labeled instances, 25000 positive, and 25000 negative reviews. In three different experiments, the dataset size considered was 5%, 10%, and 100%. As the dataset is balanced across the two provided labels, we maintain the same uniformity

Table 2: Hyperparameters for the exploration of models listed in Table 1 as in the paper.

Model	Size (L1, L2, L3)	Dropout (L1, L2, L3)	CLR	Batch Size	Epochs
CNN + CNN + CNN	(80, 256, 160)	(0.2, 0.2, 0.2)	1e-3 - 6e-3	64	10
LSTM + LSTM	(80, 256, NA)	(0.2, 0.2, NA)	1e-3 - 6e-3	64	10
CNN + LSTM + CNN	(80, 256, 160)	(0.2, 0.2, 0.2)	1e-3 - 6e-3	64	10
LSTM + CNN + LSTM	(128, 128, 128)	(0.6, 0.6, 0.6)	5e-4 - 3e-3	64	10
BiLSTM + BiLSTM	(64, 64, NA)	(0.5, 0.5, NA)	1e-3 - 6.5e-3	64	10
BiLSTM + CNN + CNN	(64, 64, 128)	(0.5, 0.5, 0.5)	1.5e-3 - 6e-3	32	20
CNN + BiLSTM + CNN	(128, 64, 128)	(0.5, 0.5, 0.5)	1.5e-3 - 6.5e-3	32	20
BiLSTM + LSTM	(64, 128, NA)	(0.5, 0.5, NA)	1.5e-3 - 6.5e-3	64	10
BiLSTM + LSTM + BiLSTM	(64, 128, 64)	(0.5, 0.5, 0.5)	1.5e-3 - 6.5e-3	64	10
BiLSTM + CNN + BiLSTM	(128, 128, 64)	(0.5, 0.5, 0.5)	1e-3 - 6e-3	64	10
CNN	(128, NA, NA)	(0.2, NA, NA)	8e-4 - 5e-3	64	10
LSTM	(128, NA, NA)	(0.6, NA, NA)	1.5e-3 - 6e-3	64	10
BiLSTM	(64, NA, NA)	(0.5, NA, NA)	1.5e-3 - 6e-3	64	10

with the reduced version using stratified while splitting the dataset into train and test datasets. The same random search module for all three experiments makes it easier to implement and observe the difference though the layer unit and dropout vary. The classification simulation on IMDB and its scaled datasets use cross-entropy loss function and optimizer were the same, and they are sparse categorical cross-entropy and RMSprop, respectively. Interestingly, RMSprop performed better than adam optimizer in our initial screening, which is heavily used in text classification problems.

3.2 Artificial data-scarcity experiment on Emotion dataset

This study also considers another multiclass classification data to illustrate the fusion model's efficacy in low-resource context. The dataset contains Twitter posts as single sentence and labeled with the basic six emotion categories: sadness, disgust, anger, joy, surprise, and fear; identical to our proposed Bangla dataset. The original dataset contains 416809 labeled texts. The dataset is not balanced across the all six emotion classes which perfectly match with our description as most of the low-resourced dataset are in the similar way. That makes this dataset a perfect candidate to experiment with our proposed fusion models. To tune for low-resource case, the dataset was scaled down to a significantly low size. Precisely, we randomly generates 0.01% and 0.02% of the dataset maintaining the data balance of the original data unchanged. For the CARER dataset, we find out that categorical crossentropy as loss and Adam as optimizer works better than the other available options such as RMSprop.

3.3 Attention Layer Position and LD Difference

As we have presented that using a basic attention layer in our fusion architecture can improve the accuracy and lower the overfitting error, the next question comes, where to use the attention layer. We conduct a total of four different runs, using the attention layer in four alternative places. Precisely, the attention layer is poised between embedding and first CNN layer, between the first CNN and first LSTM layer, between the first LSTM and second CNN layer, and between the second CNN layer and the final output layer. The first, second, and the last experiment produce very close results, where the second one has higher accuracy and lower overfitting error. However, further investigations on longer-chain architecture would be worthwhile.

4 Newly developed 6-class emotion dataset for Bengali

Bengali is the fifth largest mother tongue in the world. Approximately 228 million people speak Bengali as their first language, and around 37 million people use Bengali as their second language. Despite being the 5th largest language, natural language processing tasks for Bengali suffers because of the scarcity of resources, such as the unavailability of the standard corpus. To circumvent the problem of the NLP tasks, the newly developed sentiment dataset contains 36k texts, which are labeled as six primary categories of emotions: happiness, sadness, anger, fear, surprise, and disgust. An evaluation of the dataset with a Cohen's score of 0.920 shows agreement among annotators. The subsequent sections describe the underlying development process, including data tracing, preprocessing, marking, and verification. The newly developed corpus is made up of textual conversations, each of which is marked as angry, happy, sad, or others. The acute mixed text corpus uses Ekman's six basic emotions to master dual-language annotations and uses Cohen's Kappa coefficient to check the quality.

Emotions English	Emotions Bengali	Comments	Reasonable Words	Total Data	Total Words	Average Words
Happy	আনন্দিত	খুব মজা পাইলাম। মন খুলে হাসলাম।	মজা, হাসলাম	6000	88839	88839
Sad	দুঃখী, মন খারাপ, বিষন্ন	কবে ছিল, ছোটবেলা থেকে শুনছি সব সময় দেশের অবস্থা খুব খারাপ।	খুব খারাপ		106453	106453
Anger	ক্রোধ, রাগান্বিত	এই ধরনের অভিযোগগুলোর বিরুদ্ধে ব্যবস্থা নিয়ে সেই কেন্দ্রগুলো বাতিল করা জরুরী।	বিরুদ্ধে, বাতিল		96971	96971
Fear	ভীত, ভয়, শংকা	খুব ভয় পেয়ে গিয়েছিলাম...... আল্লাহ রক্ষা করেছে.	ভয়, রক্ষা		95013	95013
Surprise	বিস্ময়, বিস্মিত	আরো দেখার বাকি আছে!	বাকি আছে!		77355	77355
Disgust	ঘৃণা	আর কত নীচে নামবে..?? ছি: !!	ছি: !!, নীচে		107039	107039

Figure 1: Proposed chain-structured fusion models: Word embedding layer generate a word matrix for each input sentence, which acts as the input for the fusion of DNN layers. Finally, the output from the DNN models undergoes pruning and retraining to generate the deployable model version.

Table 3: Statistics of dataset and the percentage distribution of error types of words.

Classical Machine Learning Models		Classical Machine Learning Models	
Error types	**Percentage (%)**	**Attributes**	**Values**
Deacidification	41.40	Size on disk	1.95MB
Accent	39.60	Total number of expressions	36000
Abbreviation	26.18	Total number of words	571671
Separation	17.43	Maximum words in a single expression	550
International Word	14.70	Minimum words in a single expression	3
Social Media Phrase	5.93	Average words in a single expression	15.88
Adjacent	15.83	-	-

5 Data collection and processing

5.1 Properties of Emotional Expressions

Several Bengali textual expressions were investigated, and unique attributes were found for each of the six emotions, such as various types of happiness, sadness, anger, disgust, surprise, and fear. To identify the different attributes of each category, the sentence is investigated according to the following characteristics.

5.1.1 Emotion Seed Words and semantics dependency

Words that are commonly used in the context of a specific emotion. For example, the words ‘happy’, ‘enjoy’ and ‘pleased’ are considered the seed words of the happiness category. Therefore, specific seed words for specific Bengali emotions have been stored, and a few such examples are listed in the Table 1. Also, Observing the semantics of the text is one of the distinguishing features of in judging the type of emotion. In the example above, although the sentence started from the news of corrupted election in, the sentence became an ordinary sentence in news in human sadness. Therefore, the semantics of the sentence is an important parameter used to specify the expression of emotion.

5.1.2 Think Like a Person (TLTP)

Generally, emotional expression is a type that expresses someone's emotions in a specific context. Through TLTP, the rater imagined that he was in the same context as emotional expression. Through repeated pronunciation, the rater tried to imagine the situation and noticed the emotional level. Considering these characteristics, each emotion expression marks one of the six emotion categories: happy, sad, angry, disgust, surprise, and fear. The collected public dataset of 36k Bengali comments on news portals undergoes a few preprocessing steps, such as cleaning the lines with meta information like timestamp, URL, username, etc., to provide a sentence-per-line format. After that, we have 38 thousand phrases. Then, among these 38 thousand sentences, 2435 sentences were randomly selected, including at least one word outside the vocabulary. To check for words in a sentence, we first take each sentence in each line, check and set the normalized form of each sentence for analysis. This manages to provide analysis for a given sentence, which is then considered to be the correct sentence in Bengali. We consider such words in the rest of the document; otherwise, they are incorrect. In this way, we ensure that at least one word in each sentence obeys each other for alignment.

5.2 Preprocessing

We first filter the appropriate annotations for the annotation process. For the appropriateness of a given comment of we have three main criteria: 1. Write in Bengali, 2. Form a complete sentence, 3. Contain at least one misspelled word. There are 8,526 homonymic words in Bengali, of which 4,856-word meanings can only be inferred from a complete sentence. Similarly, due to the word sense disambiguation problem, some spelling errors is an unintentional character error can only be resolved in context. Therefore, we only accept full-sentence reviews on the dataset. Also, we removed comments that contained only hashtags or emojis from our analysis (no correction needed).

- The hash sign (#), because this symbol represents the hashtag in the comment. Also, this symbol is necessary to distinguish any word from the hashtag word.
- We kept the numbers that appeared and removed the hashtags unless they were misspelled by adding them to the numbers. The repeated characters did not change in a word that showed enthusiasm, which was considered a deliberate character error.
- There are spelling errors about compound words in the data set. For example, some words must be entered individually, while others must be entered side by side.

5.3 Data Annotation and Correction

After checking Bengali comments for different types of errors, the annotation of the dataset is completed. Then we refer to the authoritative dictionary and the Bengali spelling rules stipulated by Bangla Academy to make data corrections. The three scorers followed the scoring and correction process accordingly and then decided on the type of error by consensus. The types of errors used for annotations are mutually exclusive, covering all types of errors in the dataset; that is, no errors other than spelling can be found in the Bengali annotated words. There are syntactic and semantic errors. We identified eight different subgroups by considering misspellings, deliberate mistakes, non-verbal words, and slang from social media jargon. A statistic of the error type in the dataset is available in Table 2. The performance of the text correction model can be evaluated by the following indicators, the misspelled word correction rate, and the correct word non-corruption rate.

References

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. *arXiv preprint arXiv:2003.02389*, 2020.

Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE, 2017.