Generalization of Neural Networks in NLP

Harsh Vishwakarma
MTech CSA
21532
vharsh@iisc.ac.in

Abstract

This report presents significant work on the Generalization of Neural Networks for NLP. We will look at various related works regarding motivation and different techniques to achieve generalization, the advantages and disadvantages of each technique, and how each is rectified with time. We will also look at other ideas to quantify the generalization.

This report will also include the implementation of ProtoBERT, a language model to achieve maximum generalization in low-resource settings.

1 Introduction

Neural Networks have made significant progress in various tasks in reaching accuracy comparable to Human Results. But the main problem arises while generalization of the model, whether the model is learning for a generalized task or it is memorizing the examples leading to overfitting. Generalization is essential to catch up with the patterns in the data to get better performance. But various obstacles in real-world scenarios hinder this. The presence of Noise is a significant hindrance in the model's performance, as the model won't be able to generalize better as the noise increases. Another scenario can be where the data is imbalanced. In the presence of minority examples, the model either treats them as noise or memorizes them rather than generalization.

2 Related Work

Hupkes et al. (2023) proposed the taxonomy of generalization. It showed the motivation behind generalization, its types, and how different shifts in the dataset can affect generalization.

He also showed that little work is done on fairness and inclusivity of generalization. Tanzer et el. (2022) proposed a model in which they gave an architecture of BERT for low resource setting, improving the performance of BERT in few-shot. This addressed the fairness motivation of generalization.

Hoffer et el. (2018) found out that hyper-parameters also play a crucial role in generalization performance. Taking smaller batch sizes is beneficial as compared to larger batch sizes. So, they proposed that we train the model for more epochs and take smaller virtual batches for normalization rather than the original bigger batches. The advantage of such normalization is that even if we consider larger batch sizes, the actual updates in each iteration are done according to the small batch size, thus giving us similar performance in smaller and larger batch sizes.

But the catch here is that the performance depends not on the number of epochs but on the number of updates. Also, the learning rate should be more significant in starting phase as we want to make a stride as long as possible from the initial point to reach the local minima of smooth curvature (related to better generalization). After this, Naganuma et el. (2022) found out that not only just hyperparameters but also the distribution of the dataset used and the optimizer plays a crucial role.

I Can't Believe It's Not Better Workshop III at NeurIPS 2022.

He also discovered that for out-of-distribution data, non-adaptive optimizers(SGD, momentum) perform better than adaptive optimizers(Adam, RMSprop). Also, for in-distribution cases, there is no significant difference seen.

Some researchers also proposed their work on the evaluation techniques of generalization. The main problem of generalization occurs when we have a smaller number of examples for a category, so models find it hard to generalize for such non-major entities. There can be several scenarios, like a particular category of examples seen in the test sample that are not seen in the training set, an example occurring in more than one category in the training set, etc. To address this, Zhang et el. (2020) proposed the Entity Coverage Ratio, which quantifies whether an example occurring in a test set is seen in a training set. This paper also introduced the concept of $Consistency$, quantifying the relationship between the entities in the dataset. The advantage is that we can train the model considering the relationship between the entities, which are positively related as gradient step taken w.r.t one entity reduces the loss on another entity, similarly with negatively related. But the main disadvantage of such an evaluation process is that generally, models are pre-trained on the vast corpus, and finding such relationships between the entities in large datasets is computationally heavy and will take time. This problem is then addressed by Yang et el. (2022), who proposed evaluating NLP models with generalization metrics that do not need training or testing data access. They used the concept of heavy-tail self-regularization. HTSR considers Empirical Spectral Density. Now suppose for several models, we have their Density curve and the output BLEU score for a specific dataset. Plotting the BLEU score and this density function gives the best model. This idea only considers the weight parameters and the final evaluation score to measure the model's generalization performance. Nowhere it needs the dataset. The explanation behind such behavior is that the weight matrices become increasingly correlated during training, and the eigenvalues also become similar, making the ESD heavy-tailed. And with the experiment, it is found that models that have better BLEU scores show heavier-tailed ESDs.

Various techniques are adopted as state-of-the-art to improve generalization. The feature fusion technique dramatically affects the generalization of Neural Networks. Neelesh Mungoli (2023) proposed the idea of Adaptive Feature Fusion (AFF) for enhancing Generalization in Deep Learning Models.

The feature fusion process can be used by looking at the data and the model and then aggregating the extracted features for better performance. AFF leverages a combination of data-driven and model-based fusion strategies. It adaptively fuses features based on the underlying data characteristics and model requirements. The framework of AFF is that the adaptive fusion layer receives features from multiple sources and combines them using fusion functions. These functions are learned during the training process. A meta-learning step is introduced at the end, allowing the fusion layer to learn how to optimally combine the outputs of the fusion functions for a given task.

Since the pre-trained models are trained on a large dataset, they are generally trained on some general domain. To use these models, we must fine-tune them to perform tasks for a specific domain. Despite this, adapters are fine-tuned, so there will be no updates in the model parameters after pretraining.

Chronopoulou et el. (2023) proposed the idea of AdapterSoup, which averages the weights of the adapters most relevant(using Attention-Mechanism) to the new domain to improve the out-of-domain performance of pre-trained models at test time.

In contrast, one of the most critical results of Statistical Learning Theory states that the discrepancy between training and generalization error diminishes with increasing training examples. Ferreira et al. (2023) proposed the idea of Data Augmentation(DA) for increasing the diversity of training examples, thus expanding the generalization power of the model.

This method is very much exploited in Computer Vision to generalize the model, like providing the CNN with a rotatedMNIST dataset for domain adaptation, etc. In NLP, different data augmentation methods can be applied to both Data Space and Feature Space. One other way can be noise injection.

Authors have also used cGAN to generate synthetic data while fine-tuning the model. This helps in low-resource settings. The most promising technique for DA is **mixup**. The main advantage of this is that it is label-free augmentation. So we can do the data augmentation via a linear combination of examples.

Fisch et al.(2019) proposed the results of the Machine Reading for Question Answering shared on evaluating the generalization capabilities of the reading comprehension system. They adapted the idea of answering the questions based on two extractions format method

- The answer to each question must appear as a span of tokens in the passage. The idea behind here is that we can extract the main token from the question and search for similar ones in the passage to find a more accurate answer
- The passage of truncated to first 800 tokens. This will reduce the computational complexity in case of the large dataset as we have to search in a limited number of tokens

The token of the question and passage is concatenated with some special tokens to obtain a join context which is then encoded with a baseline BERT, the encoding of the input is then passed to the MultiQA model to predict the answer.

Miller et al. (2021) showed the correlation between the in-distribution and out-of-distribution data regarding generalization. Their research is based on the idea that it is rare in real-world scenarios that the training dataset and test dataset are from the same distribution. The model should handle out-of-distribution cases between the training and test more robustly. They found out that there may not be the case always that high in-distribution accuracy does not necessarily translate to good out-of-distribution generalization, and vice versa. They also found that if there is a linear trend between in-distribution and out-of-distribution like in CIFAR 10, all the datasets with similar shifts follow the same linear trend.

3 Experimental Analysis

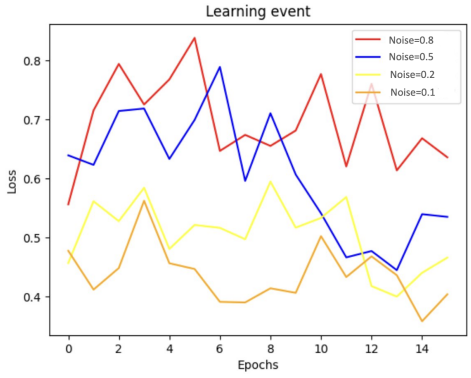
3.1 Learning Events of pre-trained BERT

Pretrained BERT is a robust model that is not prone to forgetting events. Forgetting events are the events if, for step $t-1$, the model learns an example, and for tth step, it forgets. The loss curve for the BERT is monotonically decreasing in Fig 1(a). The learning process of the model is backed by the pretraining on a large dataset.

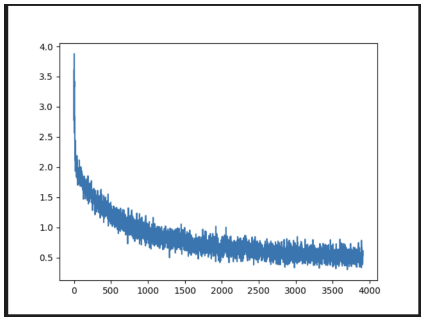
The results in Fig 1(b) are similar to the training behavior of BERT proposed by Michael Tänzer et. el. 2022. This behavior is explained by the memorization phase of BERT training which is after a certain number of training iterations, the model starts over-fitting for the noise. The above model used is protoBERT for the $SNLI$ dataset where some of the examples for certain entities are removed to make the dataset imbalanced and add some noise.

3.2 Behaviour of BERT in the presence of Noise

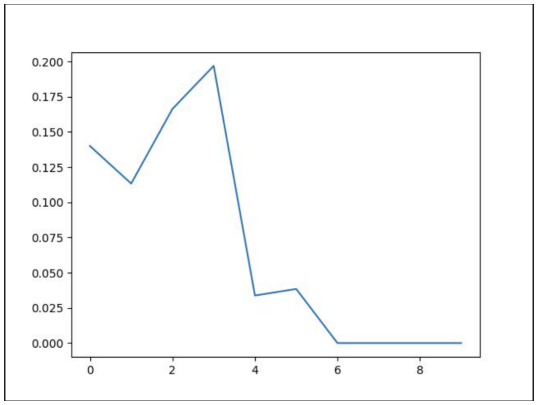
The BERT model is robust to the presence of noise, it first learns the examples and ignores the noise, after that a settling phase kicks in where the loss of the training process is stabilized, and at the end it memorizes the noise.



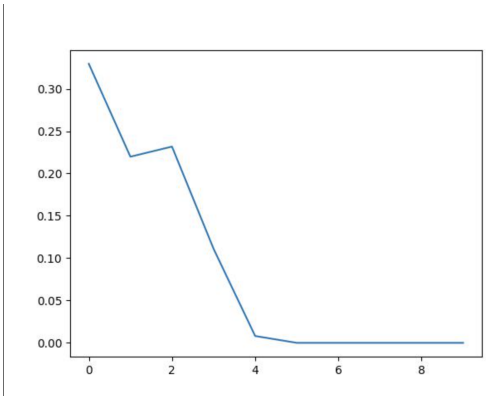
: Fig 2: BERT's behavior on noisy dataset



(a) Fig 1:
The above graph between the loss and number of learning steps shows the learning behavior of
BERT for $MNLI$ dataset.



(b) The behavior of BERT on $SNLI$ dataset for training for Loss vs Epochs



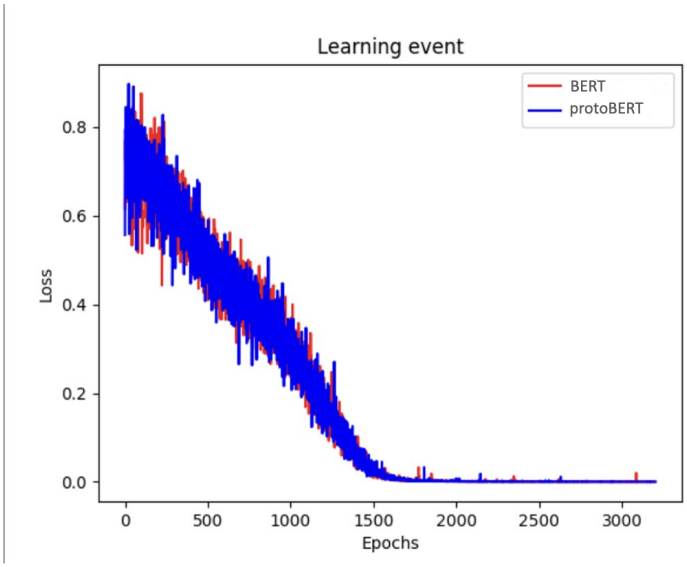
(c) The behavior of BERT on $SNLI$ dataset for test for Loss vs Epochs

Fig 2 shows BERT's behavior in the presence of noise to different levels in the $MNLI$ dataset. Similar behavior can also be seen in the Tanzer et el. (2022) paper. He also showed that in the training process, the behavior of BERT is robust to noise, that the training loss decreases with the same pattern in the presence of noise.

3.3 ProtoBERT

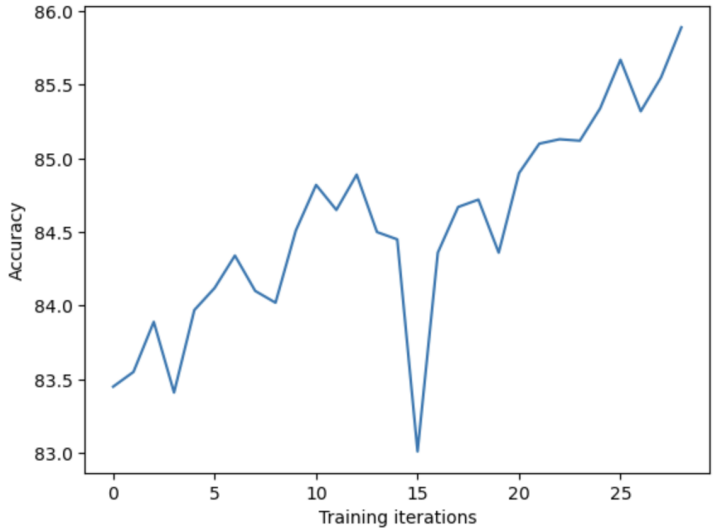
One of the most crucial findings by Tanzer et el. (2022) is BERT's behavior in a low-resource setting, basically in a Few-Shot learning scenario. The modification of ProtoBERT includes the idea of the random sampler, which keeps in check the ratio of minority to non-minority class. Also, class centroids are used to classify examples, and softmax is applied to the distances between the predicted class and class centroid.

The below graph compares BERT and protoBERT for vision the model visualBERT(pre-trained model at hugging face) is used here.



: Fig 3: Training loss on $CIFAR10$ for BERT vs protoBERT

Fig 3's result between BERT and protoBERT is similar because CIFAR10 is the entire dataset. There are no unbalanced classes. The protoBERT outperforms BERT in the dataset that is unbalanced i.e., contains a mix of minority and majority classes.

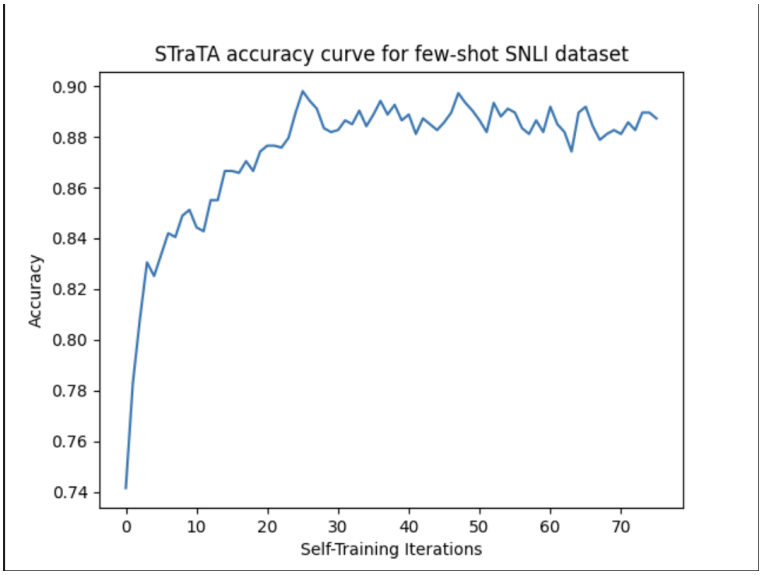


: Fig 4: Acurracy curve of protoBERT for $SNLI$ dataset

Fig 4 depicts the accuracy curve for protoBERT for the raw $SNLI$ dataset. The above results are generated by taking 70% of the dataset and training the model.

3.3.1 Drawbacks of protoBERT

ProtoBERT is a model that performs well in few-shot learning, but base-BERT exceeds in the case of the full dataset. Also, it doesn't leverage the idea of data augmentation, which can be extremely useful in the case of a few-shot. Luong et al. proposed a model(STraTA) that includes self-training with task augmentation to achieve better results.



: Fig 5: Accuracy curve of STraTA

	Full dataset	Few-shot
protoBERT	88.36	87.38
STraTA	89.21	88.72

: Table 1: Acuuracies of both models

Table 1 shows the accuracies of protoBERT and STraTA on the SNLI dataset for both few-shot and complete examples. (The actual results in the paper were slightly higher as shown in Table 1, I am not able to regenerate the same results as I did not run the training for original hyperparameters.)

References

[1] Dieuwke Hupkes, Mario Giulianelli , Verna Dankers, Mikel Artetxe Yanai Elazar , Tiago Pimentel , Christos Christodoulopoulos, Karim Lasri Naomi Saphra , Arabella Sinclair, Dennis Ulmer , Florian Schottmann Khuyagbaatar Batsuren, Kaiser Sun, Koustuv Sinha, Leila Khalatbari Maria Ryskina, Rita Frieske, Ryan Cotterell, Zhijing Jin. 2023. State-of-the-art generalization research in NLP: A taxonomy and review.

[2] Hanie Sedghi, Google Research and Preetum Nakkiran, Harvard University. A New Lens on Understanding Generalization in Deep Learning.

[3] Hiroki Naganuma, Kartik Ahuja, Ioannis Mitliagkas. 2022. Empirical Study on Optimizer Selection for Out-of-Distribution Generalization.

[4] Michael Tänzer, Sebastian Ruder, Marek Rei. 2022. Memorization versus Generalisation in Pre-trained Language Models,

[5] Jinlan Fu, Pengfei Liu, Qi Zhang, Xuanjing Huang. 2020. Rethinking Generalization of Neural Models: A Named Entity Recognition Case Study

[6] Elad Hoffer , Itay Hubara , Daniel Soudry. 2018. Train longer, generalize better: closing the generalization gap in large batch training of neural networks,

[7] Yaoqing Yang , Ryan Theisen , Liam Hodgkinson , Joseph E. Gonzalez , Kannan Ramchandran , Charles H. Martin , Michael W. Mahoney. 2022. Evaluating natural language processing models with generalization metrics that do not need access to any training or testing data,

[8] Neelesh Mungoli. 2023. Adaptive Feature Fusion: Enhancing Generalization in Deep Learning Models

[9] Alexandra Chronopoulou, Matthew E. Peters, Alexander Fraser, Jesse Dodge. 2023. AdapterSoup: Weight Averaging to Improve Generalization of pre trained Language Models

[10] Lucas Francisco Amaral Orosco Pellicer , Taynan Maier Ferreira, Anna Helena Reali Costa. 2023. Data augmentation techniques in natural language processing

[11] Tu Vu1, Minh-Thang Luong , Quoc V. Le , Grady Simon , Mohit Iyyer. 2022. STraTA: Self-Training with Task Augmentation for Better Few-shot Learning