Assessing Pruning methods on NLP models

Morgan Nañez

January 18, 2024

Abstract

Text data, generally characterized by high-dimensionality and sparsity, poses unique challenges when applying pruning techniques to text-based models. This paper investigates the impact of pruning on the sequential and contextual nature of text and the loss of linguistic nuances and patterns. The study focuses on two pruning methods: GraSP (Gradient Signal Preservation) pruning and random pruning. The experiments involve multiclass classification using the AG news dataset and language translation using the Multi30k dataset. The results demonstrate that as the compression ratio increases, both GraSP and random pruning methods experience a decrease in accuracy. However, GraSP pruning outperforms random pruning, preserving the network's performance even at higher compression rates. Furthermore, GraSP pruning excels in capturing correlated features in tasks like text classification.

1 Introduction

Text data analysis is crucial to many new areas of machine learning and natural language processing. NLP tasks can range from basic text classification, such as ham/spam email checker, to language translation, to novel text generators such as ChatGPT. However, text data is often processed using word embeddings with vast vocabularies, which lead to high dimensional and sparse data to feed into ML models. Due to large vocabularies and varying word (token) frequencies, pruning in text-based models is particularly interesting. Pruning methods are used to reduce the size and computational complexity of a model and I'm curious to see how this affects connections within text models

Text data relies heavily on sequential and contextual information. The meaning of a word in a sentence depends on the words that precede and follow it. Pruning methods have the potential to disrupt the flow of information and context within the network, which can significantly affect the model's ability to understand and generate meaningful representations of text. Moreover, text data often contains intricate linguistic nuances and patterns that are essential for accurate understanding and interpretation. Pruning methods may inadvertently remove connections or neurons responsible for capturing these nuances, leading to a degradation of the model's ability to capture fine-grained linguistic features.

In contrast, image data often exhibits more localized dependencies, where the interpretation of a pixel is less reliant on the context of neighboring pixels. Image data, while also containing patterns and structures, may be more resilient to the loss of individual pixels or features.

To address these issues, I investigate the impact of pruning on text-based models. I focus on two pruning methods: random pruning and GraSP (Gradient Signal Preservation) pruning. Random pruning involves randomly removing connections or neurons, while GraSP pruning identifies important connections based on gradients. We conduct experiments on two NLP tasks: multiclass classification and language translation.

This study aims to contribute to a better understanding of the impact of pruning on text-based models and provide insights into the challenges and considerations when applying pruning techniques towards NLP tasks.

2 Background

2.1 Text Data

Text data holds unique qualities and characteristics that should be taken into account applying pruning to NLP tasks.

High-dimensional and Sparse Nature of Text Data: Due to large vocabularies and the persistent infrequencies of tokens, text data is typically high-dimensional and very sparse. Pruning in text-based models may have a more pronounced impact due to the potential loss of critical information associated with specific words or word combinations. Removing connections or neurons can result in a higher degree of information loss compared to image data, which tends to be denser and less sparse.

Sequential and Contextual Nature of Text: Text data relies heavily on sequential and contextual information. The meaning of a word in a sentence depends on the words that precede and follow it. Pruning methods may disrupt the flow of information and context within the network, leading to a greater impact on the model's ability to understand and generate meaningful representations of text. Pruning may have a greater impact on capturing sequential and contextual information in text-based models.

Loss of Linguistic Nuances and Patterns: Text data often contains intricate linguistic nuances and patterns that are crucial for accurate understanding and interpretation. Pruning methods may inadvertently remove connections or neurons responsible for capturing these nuances, resulting in a degradation of the model's ability to capture fine-grained linguistic features. In the contrary, there may be an expansive volume of neurons, and removing a few, may not affect the few and far between important connections.

2.2 Previous results

Our first assignment, I evaluated three advanced neural network pruning methods; SNIP [1], GraSP [2] and SynFlow [3]. I compared these advanced methods with two baseline pruning methods; random pruning and magnitude-based pruning.

For this first analysis, I want to see how changing the compression ratio affects the Top 1 accuracy across the different pruners. For all of these experiment, I used MNIST and Cifar10 data, which are image datasets. This was established in the previous assignment so I will not linger here long.

Here are my results.

Compression	Rand	Mag	SNIP	GraSP	SynFlow
0.05	80.03	84.01	75.67	31.21	77.44
0.1	78.26	83.65	75.67	30.67	76.74
0.2	77.95	83.60	78.08	25.26	79.26
0.5	75.93	84.11	74.06	10.26	79.93
1	10.00	84.21	75.56	10.55	78.65
2	10.00	64.33	55.95	17.13	10.00

From this, it is evident that accuracy will decrease as you increase compression ratios. This is expected as there are fewer active neurons between layers.

3 Pruning Methods

Pruning is a technique used in machine learning and neural networks to reduce the size of a model by removing connections between neurons or entire neurons themselves. The goal of pruning is to simplify the model and reduce its computational complexity, making it easier to train and faster to execute while maintaining its accuracy. Pruning can be done during training or after training, and various methods can be used to identify the least important connections or neurons to remove.

Rather than testing the all of the previous pruning methods, I focus on using one advanced neural network pruning method, GraSP, and one baseline pruning method, random.

3.1 Random

Random pruning is a baseline pruning technique that involves randomly removing connections or neurons from a neural network during or after training. However, it is generally less effective than more sophisticated pruning methods that take into account the importance of connections or neurons based on their contribution to the network's performance.

3.2 GraSP

GraSP (Gradient Signal Preservation) pruning is a pruning technique that identifies the most important connections in a neural network by taking into account their respective gradients. This allows for the core of network structure to be preserved.

4 Experiment One: Mulitclass Classification

4.1 Multiclass Classification

Multiclass text classification is a good task for testing pruning methods because it involves a large number of input features (words or tokens) that are highly correlated with each other. Each input has concrete labels that it corresponds to and accurate predictability can be achieved using small models.

4.2 Data: AG NEWS

PyTorch's AG news [5] dataset is a widely used benchmark dataset for text classification tasks. The dataset contains a collection of news articles from four different categories: World, Sports, Business, and Science/Technology. Each category has 30,000 training samples and 1,900 test samples, making a total of 120,000 training samples and 7,600 test samples. The articles in the dataset are in plain text format, and each sample is labeled with its corresponding category.

Here's an example of a sample from the PyTorch AG news dataset:

```
'category': 'Business',
 'text': 'A Chinese food company has bought an Australian baby formula
    maker for 1.5 billion dollars, in the latest sign of China's
    insatiable appetite for safe, high-quality food products.
```

In this example, the text input is label as "Business" and the input texts discusses a Chinese food company's acquisition of an Australian baby formula maker. The text input contains relevant information that can be used to classify the article into the appropriate category.

4.2.1 Data Processing

I use two pipelines in order to process the data too be fed into the embedding layer. The `text_pipeline` function tokenizes the text using the vocabulary, and the `label_pipeline` function converts the label to an integer.

4.3 Methods

I utilized the torchtext library, namely tokenizers and vocabulary builders. I also built upon the existing code base from assignment one to aid in running single shot experiments, training, evaluating, and pruning. I had to integrate the new model into the code base, and make code changes reflecting the new data and model into the existing protocols.

It is important to know that, in keeping with assignment one, the only layers that were subject to pruning was the single `torch.nn.Linear` Layer. As you'll see, I did not use any batch norm layers.

When running the execution code for each pruning method I used all of the standard parameters, such as `post_epochs = 10`, and only changed the pruner, dataset, and model.

I used `nn.CrossEntropyLoss()` for loss and `optim.Adam` as my optimizer. To obtain results, I trained for 10 epochs

4.4 The Model

TextClassificationModel is a subclass of torch.nn.Module. This class represents a text classification model that is fed news text data and outputs its corresponding category. The model is initialized to embed 95811 words, which is the size of the vocabulary. The model uses a torch.nn.EmbeddingBag layer for embedding the input text, followed by a linear layer for classification. The forward method defines the forward pass of the model, returning the output of the linear layer after passing through the embedding layer.

```
TextClassificationModel(
  (embedding): EmbeddingBag(95811, 64, mode='mean')
  (fc): Linear(in_features=64, out_features=4, bias=True)
)
```

4.5 Results

Table 1: Comparison of Compression Ratio and Accuracy after Pruning

Compression Ratio	Post GraSP Accuracy	Post Random Accuracy
0.0	97.67	97.67
0.1	97.56	97.57
0.5	97.22	97.23
1	96.50	96.57
2	71.85	48.17

Table 2: Comparison of Compression Ratio and FLOP Sparsity

Compression Ratio	GraSP FLOP Sparsisty	Rand FLOP Sparsisty
0.1	0.80	0.79
0.5	0.32	0.32
1	0.11	0.11
2	0.023	0.023

Table 3: Compression Ratio represents the degree of compression applied to the neural network model, indicating the proportion of connections or neurons that were pruned. Post GraSP Accuracy shows the accuracy achieved after pruning using the GraSP method. Post Random Accuracy shows the accuracy achieved after pruning using the random pruning method.

From the table, it is clear that increasing the compression ratio, which more aggressive prunes, negatively impacts the accuracy of the pruned models for both GraSP and random pruning methods. It is expected that GraSP pruning would perform better than random pruning because it identifies the most important connections in a neural network using their gradients. GraSP can selectively preserve the most crucial connections during pruning. By preserving the connections that contribute to the representation of correlated features, GraSP ensures that the model retains the necessary information to perform well on the task, even at higher compression rates.

We also see that both pruning methods achieve similar sparisity's, which emphasize the higher performance of GRaSP compared to random. GraSP's ability to handle correlated features contribute to its better accuracy compared to random pruning.

5 Experiment Two: Language Translation

Seq2seq language translation, or encoder-decoder architecture, is widely used in NLP. The encoder processes the source language sequence, such as an input sentence in Dutch, into a fixed-length representation called the context vector. This encoding step captures the semantic and contextual information of the source sequence. The decoder takes the information and is then able to reconstruct a similar contextualized text structure in the target language.

Seq2seq models are trained using pairs of aligned source and target sequences. The training objective is to minimize the difference between the predicted target sequence and the ground truth target sequence.

5.1 Multi30k

PyTorch's Multi30k dataset is a widely used benchmark dataset for machine translation tasks, specifically for translating sentences from English to other languages.

The Multi30k dataset consists of approximately 30,000 parallel sentences across multiple languages. The sentences are derived from the Flickr30k dataset, which originally contained descriptions of images. The dataset includes English sentences as the target language and their corresponding translations in other languages like German, French, and Czech.

I arbitrarily choose German as my source language. The German language vocabulary included 19214 words, while the English vocabulary included 10837 words, or possible tokens.

Here's an example of a pair of sentences from the PyTorch Multi30k dataset, with English as the source language and German as the target language

```
English sentence: "A man wearing a black jacket and a red hat is standing
    next to a bike."
German translation: "Ein Mann in einer schwarzen Jacke und roten Mutze
    steht neben einem Fahrrad."
```

5.2 Data Processing

I defined source and target languages as German ('de') and English ('en'). A helper function, `yield_tokens`, is defined to tokenize sentences in the data iterator and build the language vocabularly. Special symbols and indices are defined, and vocabularies for both languages are built using `build_vocab_from_iterator`. The code also includes functions to generate masks for source and target sequences, this is used for attention. Additional helper functions are defined for data transformation and collation. Finally, a data loader is created using the Multi30k dataset, and the vocabulary sizes for the source and target languages are obtained..

5.3 Methods

I built upon the existing code base from assignment one to aid in running single shot experiments, training, evaluating and pruning. I had to integrate the new model into the existing code, and make code changes reflecting the new data and model into the existing protocols.

The only layers that were pruned were again `torch.nn.Linear` Layers.

My model parameters included `torch.nn.CrossEntropyLoss(ignore_index=PAD_IDX)` as my loss function and `torch.optim.Adam(model.parameters(), lr=0.0001, betas=(0.9, 0.98), eps=1e-9)` as my optimizer. I used the same execution code as experiment 1, only using a different model and data set.

5.3.1 Metrics

Because this is a translation tasks, using regular accuracy metrics as a tool for correctness is not adequate. Instead, I used a standard metric for this task called Bleu[6].

The BLEU score is computed using the following formula:

$$\text{BLEU} = \text{BP} \cdot \exp\left(\frac{1}{N}\sum_{n=1}^{N}\log(p_n)\right)$$

where BP represents the brevity penalty, N is the maximum n-gram order considered, and p_n is the precision for n-grams. The precision measures the ratio of matching n-grams in the machine-generated translation to the total number of n-grams. In my code I used `N = 4` with each n-gram having equal weight.

I used `from torchmetrics.functional import bleu_score` for my bleu score function. After calculating loss from the model's outputted logits, I used the model, at it's current state in training to translate the source language data, from data loader into the target language.

Here is a snippet of my code to showcase my training methodology using BLEU score:

```
with torch.no_grad():
        for idx, (src_tokens, tgt_tokens, tgt_words, src_words) in
            enumerate(dataloader):

            #external code to generate token attention mask
            src_mask, tgt_mask, src_padding_mask, tgt_padding_mask =
                create_mask(src_tokek, tgt_tokens,)

            #model (Seq2Seq) generates logits
            logits = model(src_tokens, tgt_input, src_mask, tgt_mask,
                src_padding_mask, tgt_padding_mask, src_padding_mask)

            output = logits.reshape(-1, logits.shape[-1])

            blues = 0
            target_words = []
            predicted_words = []

            #iterte over all source ("de") words
            for i in range(src_words):

                #external function that uses the same Seq2Seq model to
                    output and a translated ("en") output
                translated_words = translate(model, src_words[i])

                predicted_words.append(translated_words)
                target_words.append(tgt_words[i])

            bleu_score = bleu_score(predicted_words, target_words)
```

5.4 The Model

Seq2SeqTransformer model uses a transformer-based architecture with encoder-decoder structure for sequence generation tasks. It employs self-attention mechanisms, feed-forward neural networks, layer normalization, and dropout to capture dependencies and generate accurate translations or sequence outputs. The model incorporates token embeddings, positional encoding, and a linear generator to process input and produce output sequences.

```
Seq2SeqTransformer(
  (transformer): Transformer(
    (encoder): TransformerEncoder(
      (layers): ModuleList(
        (0-2): 3 x TransformerEncoderLayer(
          (self_attn): MultiheadAttention(
            (out_proj): NonDynamicallyQuantizableLinear(in_features=512,
                out_features=512, bias=True)
          )
          (linear1): Linear(in_features=512, out_features=512, bias=True)
          (dropout): Dropout(p=0.1, inplace=False)
          (linear2): Linear(in_features=512, out_features=512, bias=True)
          (norm1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
          (norm2): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
```

```
            (dropout1): Dropout(p=0.1, inplace=False)
            (dropout2): Dropout(p=0.1, inplace=False)
          )
        )
        (norm): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
      )
      (decoder): TransformerDecoder(
        (layers): ModuleList(
          (0-2): 3 x TransformerDecoderLayer(
            (self_attn): MultiheadAttention(
              (out_proj): NonDynamicallyQuantizableLinear(in_features=512,
                  out_features=512, bias=True)
            )
            (multihead_attn): MultiheadAttention(
              (out_proj): NonDynamicallyQuantizableLinear(in_features=512,
                  out_features=512, bias=True)
            )
            (linear1): Linear(in_features=512, out_features=512, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
            (linear2): Linear(in_features=512, out_features=512, bias=True)
            (norm1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
            (norm2): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
            (norm3): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
            (dropout1): Dropout(p=0.1, inplace=False)
            (dropout2): Dropout(p=0.1, inplace=False)
            (dropout3): Dropout(p=0.1, inplace=False)
          )
        )
        (norm): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
      )
    )
    (generator): Linear(in_features=512, out_features=10837, bias=True)
    (src_tok_emb): TokenEmbedding(
      (embedding): Embedding(19214, 512)
    )
    (tgt_tok_emb): TokenEmbedding(
      (embedding): Embedding(10837, 512)
    )
    (positional_encoding): PositionalEncoding(
      (dropout): Dropout(p=0.1, inplace=False)
    )
)
```

5.5 Results

Table 4: Comparison of Compression Ratio and Accuracy after Pruning

Compression Ratio	Post GraSP Bleu Score	Post Random Bleu Score
0.0	22.96	22.96
0.1	22.73	22.7
0.5	24.24	24.07
1	22.8	22.93
2	22.85	22.82

Here is an exmaple of a BLEU Score from a trained version of the model after 10 epochs using GRaSP pruning at `compression = 2`:

Table 5: Comparison of Compression Ratio and FLOP Sparsity

Compression Ratio	GraSP FLOP Sparsisty	Rand FLOP Sparsisty
0.1	0.86	0.79
0.5	0.27	0.31
1	0.01	0.10
2	0.023	0.01

```
Predicted Words: [' People are eating a table with a table and some food
    . ', ' A family is taking his head as his friend . ', ' Two kids are
    walking on a wall with a dog and a woman standing in the background]
Target words: ['People eating at a long table full of food and drink.',
    'A food vendor happily serves his hungry customers.', 'Two horses are
    attached to a carriage with a dog and man inside, and a woman standing
     next to it.']
BLEU Score: 0.1496
```

From the tables, we observe that the BLEU score does not change much, or in some cases, even improves slightly. There are several reasons why this might be the case.

Redundancy: Neural networks used for machine translation tasks often have a high degree of redundancy, which means that removing a subset of neurons does not significantly impact the overall performance of the model. As a result, pruning methods can remove redundant neurons without affecting the overall quality of the translation.

Overfitting: Overfitting occurs when a model is too complex and memorizes the training data instead of learning general patterns. This can be applicable to our model due to the large vocabularly sizes used. Pruning can help prevent overfitting by reducing the complexity of the model, which can improve the generalization performance on the test set.

I was surprised to see that GraSP only did marginally better than random. This leads me to be think that perhaps the language models were too big and there were too many noisy connections that there was a high chance of randomly removing a noisy neuron compared to an important one.

Interestingly, it's curious that the Bleu score is hovering around 25%. I wonder if this is related to the uniform weight distribution over the 4 n-grams. The model may be picking up single words at much higher rate compared to the nearly zero matches for 4 n-gram matches. This would be interesting further research to conduct to explore which weight distribution is optimal for the best context between language to be preserved.

6 Conclusion

6.1 NLP TASKS

Dependency on contextual information: Language translation tasks, such as machine translation, heavily rely on contextual information present in the input text. Each word's meaning and translation can depend on the surrounding words and the overall sentence structure. Pruning, which involves removing connections or neurons from the model, can disrupt the flow of information and context within the network. As a result, the translation quality may be more sensitive to the removal of important connections or neurons compared to text classification tasks, where the presence of specific words or features is generally more critical for classification decisions.

Clustering and Locality: Multiclass text classification large number of input features (words or tokens) that are highly correlated with each other because they are assigned to cluster into N groups. Because the inputs are mapped distinctly and local to one another, it may provide more overhead to be available to be prune, which also can lead to it being a less exhaustive and complex model compared to a language translator.

Sensitivity to subtle changes: Language translation is highly sensitive to subtle changes in the input text. Even small perturbations or alterations in the source sentence can lead to different translations. Pruning, which modifies the model's architecture or connectivity, introduces changes that can have a significant impact on the translation output. In text classification, the decision boundaries

are often more robust, and small perturbations in the input text may have a lesser effect on the classification outcome.

In summary, the complex nature of language translation tasks may make it more susceptible to negative effects of pruning, while they may also benefit from dense nuanced linguistic patterns, larger vocabulary sizes, and sensitivity to subtle changes. Pruning can disrupt the flow of information, alter word representations, and affect the model's ability to capture essential linguistic features, resulting in a greater impact on translation quality.

6.2 Text vs. Image Data

As discussed above, text and image data have vastly different ways of being represented in machine learning. Text data contains qualities that make it potentially more susceptible to pruning methods.

Overall, the high-dimensional and sparse nature of text data, the reliance on sequential and contextual information, the presence of intricate linguistic nuances, and the impact on the transferability of pre-trained models make text-based data more susceptible to the effects of pruning. Pruning in text-based models can result in a higher loss of critical information, disrupt the understanding of language, and degrade the model's performance compared to image-based models.

References

[1] Lee, N., Ajanthan, T. and Torr, P.H., 2018. Snip: Single-shot network pruning based on connection sensitivity. arXiv preprint arXiv:1810.02340.

[2] Wang, C., Zhang, G. and Grosse, R., 2020. Picking winning tickets before training by preserving gradient flow. arXiv preprint arXiv:2002.07376.

[3] Tanaka, H., Kunin, D., Yamins, D.L. and Ganguli, S., 2020. Pruning neural networks without any data by iteratively conserving synaptic flow. arXiv preprint arXiv:2006.05467.

[4] Frankle, J., Dziugaite, G.K., Roy, D.M. and Carbin, M., 2020. Pruning Neural Networks at Initialization: Why are We Missing the Mark?. arXiv preprint arXiv:2009.08576.

[5] https://paperswithcode.com/dataset/ag-news

[6] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL '02). Association for Computational Linguistics, USA, 311–318. https://doi.org/10.3115/1073083.1073135