CheXPrune: sparse chest X-ray report generation model using multi-attention and one-shot global pruning

Navdeep Kaur1,2 · Ajay Mittal1

Received: 18 November 2021 / Accepted: 5 October 2022 / Published online: 1 November 2022
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2022

Abstract
Automatic radiological report generation (ARRG) smoothenes the clinical workflow by speeding the report generation task. Recently, various deep neural networks (DNNs) have been used for report generation and have achieved promising results. Despite the impressive results, their deployment remains challenging because of their size and complexity. Researchers have proposed several pruning methods to reduce the size of DNNs. Inspired by the one-shot weight pruning methods, we present CheXPrune, a multi-attention based sparse radiology report generation method. It uses encoder-decoder based architecture equipped with a visual and semantic attention mechanism. The model is 70% pruned during the training to achieve 3.33× compression without sacrificing its accuracy. The empirical results evaluated on the OpenI dataset using BLEU, ROUGE, and CIDEr metrics confirm the accuracy of the sparse model vis-à-vis the dense model.

Keywords Radiological reports · Chest radiographs · Deep-learning · Radiological report generation · Textual description · Multi-attention · Pruning · Sparse DNN

1 Introduction
Artificial Intelligence (AI) has played a vital role in the healthcare industry such as disease detection, diagnosis, monitoring, and robotic surgeries. Frost and Sullivan (2015), forecast the AI market to be 6 billion dollars by 2021. The most widely adopted AI-powered solutions aim to facilitate patients, support clinicians, and not replace them. Although AI has achieved high predictive performance in medical image analysis, medical report generation remains a comparatively less explored and challenging task. A successful solution to medical report generation can help the radiologist drive efficiency and reduce human error in interpreting a scan. Chest X-ray (CXR) is the most commonly used imaging modality, with over two billion procedures performed annually. Although easy to acquire, interpreting CXR images is one of the most challenging tasks. Its accurate interpretation requires high-level expertise and years of training. Thus, automating reading the CXR image and describing the observations in words without human involvement enables instant interpretation without delay and human error.

It is always beneficial to have a deeper understanding of the domain in order to recognise and describe the key aspects. As a result, earlier ARRG systems had to rely on manual feature extraction and sentence retrieval. However, recent developments in deep learning have removed the need for human interaction during feature extraction (Jaiswal et al. 2019; Rajpurkar et al. 2017) and paragraph generation (He and Deng 2018; Yang et al. 2011). A successful ARRG system has to identify the abnormalities accurately and capture the semantic relationship between them, specifically its location and the severity involved. Furthermore, these visual observations have to be described in generic language focusing on specific medical concepts.

Based on the recent observations about attention mechanisms, this work investigates the use of a multi-attention mechanism for ARRG. The empirical study of the visual and semantic attention mechanism concludes that attending both the visual and semantic features gives promising results rather than attending both features individually. These ARRG systems usually require high computation power and time,
resulting in high energy consumption and low inference speed. These ARRG systems are generally overparameterized and comprise millions of parameters. Though overparameterization is helpful in successful training, pruning (Blakeney et al. 2020; LeCun et al. 1990; Blalock et al. 2020; Malach et al. 2020; Lee et al. 2018; Zhang and Stadie 2019; Zhu and Gupta 2017) the model can remove redundancy while maintaining good performance. Furthermore, size hinders their deployment on resource-limited devices like embedded systems and mobile devices, thereby restricting their adoption in the real world. As abnormality detection is of foremost importance for the ARRG system, pruning should not affect its accuracy. In this work, we experimented with several methods to determine when and how to prune an ARRG system for model compression without compromising its accuracy.

The main contributions of our research work are as follows.

1. We propose CheXPrune, a sparse multi-attention based deep neural network for CXR report generation. CheXPrune, comprises a base model, which is an encoder-decoder based architecture equipped with the multi-attention mechanism. The base model is trained, pruned, and fine-tuned. Experimental results indicate that the generated sparse model has equivalent accuracy to the base model.
2. To the best of our knowledge, we are the first to apply pruning techniques in radiological report generation.
3. The empirical study of the pruning percentage induced in different model layers signifies that embedding layers present in the model are the most significant layers; thus are least pruned, and the encoder layer has the maximum redundancy; thus is highly pruned.
4. CheXPrune gives consistent and reproducible performance across a wide range of scenarios when evaluated using several benchmark metrics.
5. CheXPrune is significantly lighter (3.33x compressed) than a non-pruned model and thus can be easily deployed.

The rest of the paper is organized as follows. The structure of the CXR report is explained in Sect. 2. Section 3 summarizes the related work done to date in generating the report from CXR images. The materials and method of the proposed model are discussed in detail in Sect. 4. The experimentation details, results, and its comparison with the base model are summarized in Sect. 5. Comparison with state-of-the-art methods are presented in Sect. 6. Conclusion and future scope of the work is presented in Sect. 7.

2 Structure of CXR report

Writing a radiology report is an art; describing the CXR image observations in words needs high-level expertise. Although structural reporting is desirable to improve accuracy, radiologists have used no standard format for writing radiological reports from CXRs (as mentioned by Nobel et al. (2020)). The majority of the reports have the following sections (as shown in Fig. 1):

1. **Patient Information**: It includes patient’s personal information,
2. **Indications**: It includes details the reason for the test,
3. **Findings**: It enumerates radiologists’ observations, and
4. **Impressions**: It includes the diagnosis suggestions.

Thus, automation can be applied for generating the findings and impression sections closely related to each other. To understand the relationship between the words, the dependency graph is shown in Fig. 2.

3 Related work

During the last decade, deep learning has been widely used in healthcare for diagnosis and assistance. In recent years, the automatic generation of medical reports as a key application in this field has increased researchers’ interest. The existing ARRG systems can be broadly classified into four categories:

1. ARRG systems based on encoder-decoder framework
2. ARRG systems based on attention
3. ARRG systems based on reinforcement learning

Fig. 1 Sample radiology report with associated images from OpenI dataset developed by Demner-Fushman et al. (2012)

izados by Springer
4. ARRG systems based on graphs

3.1 ARRG systems based on encoder-decoder framework

In literature, most of the ARRG systems use the encoder-decoder framework. The encoder extracts the features and their relationships from the input image, and the language decoder generates sentences from those features. The general workflow pipeline of an encoder-decoder-based ARRG systems is shown in Fig. 3a.

Shin et al. (2016) were the first to propose an encoder-decoder framework that detects the disease and describes its content, e.g., location, severity, and the affected organ in the chest x-ray images. An encoder takes an image as input and learns its features; and the decoder generates sentences to describe those features. The encoder has to deal with images; hence CNN being good at spatial data becomes an obvious choice to be used as encoders. The decoder has to generate sequential data, hence RNN are used as a decoder in ARRG systems. CNN contains two parts: feature extraction and classification. Features obtained from the last convolution layer and fully connected layers are highly specific for a given class (by its presence or absence). In contrast, features from other convolutional layers are more general. Several CNN models such as VGG19 by Simonyan and Zisserman (2014), Densenet by Huang et al. (2017), Resnet by He et al. (2016) are used to learn the visual features. Transfer learning has also influenced this field whereby pre-trained models over distinct large datasets are used instead of training the model from scratch. Initially, due to the lack of large labeled CXR datasets, the models are pre-trained over large natural image datasets; specifically, Imagenet by Deng et al. (2009). The models pre-trained over the natural image dataset show...
low performance in CXR report generation. The models pre-trained over the same domain dataset show promising results by capturing domain-specific features for decoding (Li et al. 2018, 2019a; Yuan et al. 2019). LSTM, a type of RNN capable of retaining selective information from previous distance steps, is widely used as the decoder. Single LSTM successfully generates a one-sentence caption but could not provide efficient results in paragraph generation. Thus Jing et al. (2017) adopted Hierarchical LSTM to produce long texts and showed that Hierarchical LSTM outperforms the single LSTM. Hierarchical LSTM comprises two layers of LSTM: sentence LSTM and word LSTM. Sentence LSTM will generate a topic, and word LSTM will generate a sentence for each topic. In ARRG systems, the sentences are to be generated for normal and abnormal observations. In the available datasets, the descriptions for normal observations are similar; on the other hand, have distinct descriptions for abnormal observations, leading to biasing. To remove this shortcoming, Harzig et al. (2019) replaced with word LSTM with dual word LSTM. The context of the topics are not considered, so these methods result in repeating sentences which was well resolved by Yin et al. (2019). They used a global label pooling mechanism to match the context and limit the repeating sentences.

3.2 ARRG systems based on attention

The major limitation of encoder-decoder architecture is that it encodes the input sequence to a fixed-length internal representation. So the limit on features learned will make it unfit for very long input sequences. The attention mechanism overcomes this problem by focusing on certain parts of the input sequence or referring back to the input sequence when predicting a specific part of the output sequence, enabling easier learning and higher quality. The workflow pipeline of the ARRG system based on attention is shown in Fig. 3b.

Besides visual features, Jing et al. (2017) predicted Medical Text Indexer (MTI) annotated tags. The semantic features extracted from the embeddings of M most likely tags are combined with visual features to generate the context vector, which is further used to generate the topics and the sentences for each topic. Jing et al. (2017) concluded that the co-attention mechanism that leverages both the visual and semantic features performs better than applying semantic and visual attention individually.

Similarly, Text-Image Embedding network (Tienet) developed by Wang et al. (2018) focused on the mixture of image and text features to provide better salient and meaningful extractions. Tienet integrated the multilevel model into an end-to-end trainable CNN-RNN model used for both CXR image classification and report generation. Xu et al. (2015) used bidirectional LSTM for encoding the semantic information of the last generated sentence to feed it to the attention mechanism to generate a context vector for the current sentence. Focusing on multi-view features, Yuan et al. (2019) discussed different fusion techniques and culminated that combining the visual and text attention outperforms the individual attention models.

3.3 ARRG systems based on reinforcement learning

Reinforcement Learning is used to improve the learning capability of the neural network (Fig. 4). Further influenced by achievements of reinforcement learning in language generation, Li et al. (2018) used CIDEr metrics for computing discounted rewards for retrieval and generation modules. On the other hand, Liu et al. (2019) gave full consideration to clinical efficacy and used both the natural language generation and clinical coherence rewards in reinforcement policy for focused training of the hierarchical language generation model.

3.4 ARRG systems based on graphs

To mimic the radiologist’s behavior, Li et al. (2019a) introduced a graph-based ARRG system comprising knowledge-driven encode, retrieve and paraphrase modules. Using the imbibed medical knowledge, the encoder module transforms the features into an abnormality graph, and corresponding to each abnormality sentence is retrieved from the shelf. Finally, the retrieved sentences are rephrased to generate the desired report. In most ARRG systems, the mutual influence of medical observations over one other is not considered. Zhang et al. (2020) proposed a graph-based model in which a node represents a disease finding. The related findings are closely connected to influence each other during graph propagation and aggregation mutually. Inspired by the success of transformers in various research areas, the variants of the same have also been tried in ARRG systems. Lovelace and Mortazavi (2020) used transformer architecture to generate clinically coherent reports. Chen et al. (2020) incorporated relational memory into a decoder transformer to record previously generated information. Alfarghaly et al. (2021) used a pre-trained GPT2 on both visual and semantic features for ARRG. The pre-trained model eliminated the need for vocabulary. Inspired by curriculum learning (CL) Bengio et al. (2009) and Nooralahzadeh et al. (2021) divided the ARRG process into two steps. The first step generates global concepts from the input image, whereas the second step uses transformer-based architecture to reform them into a finer and coherent text. The comparison of various ARRG systems is summarized in Table 1.
Table 1 Performance comparison of state-of-the-art ARRG systems

| Metrics | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 | CIDEr | ROUGE-L |
|---------|--------|--------|--------|--------|-------|---------|
| ARRG systems based on encoder decoder framework |
| Shin et al. (2016) | 0.319  | 0.198  | 0.133  | 0.094  | 0.268 | 0.291   |
| Krause et al. (2017) | 0.364  | 0.232  | 0.161  | 0.114  | 0.306 | 0.291   |
| Harzig et al. (2019) | 0.357  | 0.233  | 0.165  | 0.118  | 0.313 | 0.340   |
| Yin et al. (2019) | 0.445  | 0.292  | 0.201  | 0.154  | 0.342 | 0.344   |
| ARRG systems based on attention mechanism |
| Jing et al. (2017) | 0.517  | 0.386  | 0.306  | 0.247  | 0.327 | 0.447   |
| Wang et al. (2018) | 0.286  | 0.159  | 0.103  | 0.073  | –     | 0.226   |
| Huang et al. (2019) | 0.476  | 0.340  | 0.238  | 0.169  | 0.297 | 0.347   |
| Yuan et al. (2019) | 0.476  | 0.340  | 0.238  | 0.169  | 0.297 | 0.347   |
| Li et al. (2019b) | 0.482  | 0.325  | 0.226  | 0.162  | 0.280 | 0.339   |
| Li et al. (2019a) | 0.482  | 0.325  | 0.226  | 0.162  | 0.280 | 0.339   |
| ARRG systems based on reinforcement learning |
| Xiong et al. (2019) | 0.350  | 0.234  | 0.143  | 0.096  | 0.323 | –       |
| Liu et al. (2019) | 0.359  | 0.237  | 0.164  | 0.113  | 1.424 | 0.354   |
| Li et al. (2018) | 0.438  | 0.298  | 0.208  | 0.151  | 0.343 | 0.322   |
| Jing et al. (2020) | 0.464  | 0.301  | 0.210  | 0.154  | 0.275 | 0.362   |
| ARRG systems based on graphs |
| Li et al. (2019a) | 0.482  | 0.325  | 0.226  | 0.162  | 0.280 | 0.339   |
| Zhang et al. (2020) | 0.441  | 0.291  | 0.203  | 0.147  | 0.304 | 0.367   |
| Alfarghaly et al. (2021) | 0.387  | 0.245  | 0.166  | 0.111  | 0.257 | 0.289   |
| Chen et al. (2020) | 0.470  | 0.304  | 0.219  | 0.165  | –     | 0.371   |
| Nooralahzadeh et al. (2021) | 0.477  | 0.304  | 0.213  | 0.156  | –     | 0.362   |
| Lovelace and Mortazavi (2020) | 0.415  | 0.272  | 0.193  | 0.146  | 0.316 | 0.318   |

Fig. 4 Illustration of basic ARRG system based on reinforcement learning.
4 Materials and method

4.1 Datasets

In this work, a large-scale OpenI dataset by Demner-Fushman et al. (2012) has been used for training, validation, and testing of the proposed model. This dataset consists of 7,470 lateral and frontal view CXR images with 3955 radiology reports. All the radiographs are 512 × 512 DICOM (Digital Imaging and Communications in Medicine) images. Each report is associated with a pair of frontal and lateral view images containing mainly four sections: impression, findings, indication, and medical subject headings (MeSH). This dataset is maintained by the U.S. National Library of Medicine (USNLM). All the images are collected from various hospitals associated with the Indiana University School of Medicine, United States. In this study, all the data are split into a training, validation, and testing set in the ratio of 7:1:2, respectively, ensuring that the same patient’s frontal and lateral images belong to the same set.

4.2 Evaluation metrics

The problem at hand was to generate a report similar to human-written reports. Thus to evaluate the generated report six evaluation metrics have been used: BLEU by Papineni et al. (2002), ROUGE-L by Lin (2004), CIDEr by Vedantam et al. (2015). BLEU (Bilingual evaluation understudy) is a precision-based metric that determines how many n-gram words in the generated report are present in the original report. Whereas complementing BLEU, ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a recall-based metric that determines how many words in the original report occur in the generated report. In particular, we have used ROUGE-L (Longest Common Subsequence), which uses in-sequence matches that map sentence-level word order as n-gram. CIDEr (Consensus-based Image Description Evaluation) metric is the precision and recall-based metric. It is a measure for consensus encoding how often n-grams in the generated report are present in the original report.

4.3 Proposed method

Inspired by the tremendous growth in the field of computer vision and NLP, we propose a deep neural network (CheXPrune) that takes a CXR image as input and generates its description as output. The proposed model CheXPrune consists of one base model, a variant of encoder-decoder architecture equipped with a multi-attention mechanism. The overview of the base model is shown in Fig. 5. Further, during training, the base model is pruned to get a lighter model (CheXPrune), as shown in Fig. 6.
4.3.1 Base model

Given a CXR image \(I\), the aim of the base model is to generate a coherent report, \(R = \{w_1, w_2, \ldots, w_N\}\), where \(w_i \) presents the index of word in \(V\), the vocabulary of all words contained in the datasets. The base model comprises of four modules:

1. Feature Extractor
2. Multi-Label Classifier
3. Context Vector Generator
4. Language Generator

1. Feature Extractor: The feature extraction network is a convolutional neural network, that takes a set of images \([I_i]_{i=1}^{K}\) to extract visual features \([v_k]_{k=1}^{K}\) for each image. More specifically, we have used VGG19 to extract the features from the CXR image. We resize the input images to \(224 \times 224\) to keep consistency with the VGG19. As each layer of CNN learns filters of increasing complexity, thus the later layers learn more features. Hence the visual features of size \(7 \times 7 \times 512\) are extracted from the last convolutional layer of VGG19. Moreover, we apply the average pooling to obtain the global visual feature. This average of visual features \(V = [v_k]_{k=1}^{K}\) is further passed to a multi-label classifier (MLC) for prediction of the tags and the visual attention network.

2. Multi-Label Classifier (MLC): CXR images have more than one observation; thus, the sub-problem at hand can be considered as multi-label classification. MLC takes visual features as input and predicts the probability distribution over all the tags. For multi-label classification, we have used one fully connected layer that takes visual features \(V\) as input and linearly transforms it into a tensor \(O(V) = W^TV + B\) where \(V\), \(W\), and \(B\) are input, weights, and bias of the fully connected layer. Then, softmax is applied to predict the probability distribution of number of classes of tags, that ranges between 0 and 1. Finally, it is followed by a sparse embedding layer, which is used to store word embeddings and retrieve them using indices. The embeddings of the \(N\) (we took \(N=10\)) most likely predicted tags \([t_n]_{n=1}^{N}\) serve as semantic features that help generate topics further down the line. The training loss of tag prediction \((l_{tag})\) is a mean square error loss between ground truth and predicted distributions.

3. Context Vector Generator: Rather than generating the context vector by considering the image as a whole, we used attention networks to focus on the salient part of the inputs while ignoring others. Several researchers have shown promising results using visual attention in paragraph generation while ignoring important semantic information. Whereas another bunch of researchers completely relied on semantic information without considering visual features. Inspired by Jing et al. (2017), we also studied the effect of attending the visual and semantic features individually and simultaneously as summarized in Table 4. And concludes that attending both the visual and semantic features simultaneously gives better results. Thus, we used both the visual and semantic attention networks to focus on a significant part of the image and semantically important concepts, respectively. We computed the context vector \((ctx)\) by concatenating the visual context vector \((ctx_v)\) and semantic context vector \((ctx_s)\). The visual attention network consists of two fully connected layers. One takes visual features \(V\) from CNN encoder and transforms it into \(O_v(V) = W_vV\). Whereas other fully connected layer takes the hidden state of sentence LSTM at \(t-1\) timestamp \((h_{-1})\) and transforms it into \(O_m(V) = W_mh_{-1}\). Tanh activation function is applied over \(O_v(V) + O_m(V)\) and further fed to fully connected layer to transform it into \(\theta_{v,att} = W_{v,att}\tanh(W_vV + W_mh_{-1})\). Finally the softmax is applied to get visual attention \(\alpha_{v,n}\) and generate the visual context vector \(ctx_v\) as

\[
ctx_v = \sum_{n=1}^{N} \alpha_{v,n} v_n, \quad \alpha_{v,n} \propto e^{\theta_{v,att}}.
\]

The similar model is followed by semantic attention network where it takes semantic features from MLC instead of visual features. Similarly the semantic context vector is generated as

\[
ctx_s = \sum_{m=1}^{M} \alpha_{s,m} t_n.
\]

Concatenation of both visual \(ctx_v\), as well as semantic \(ctx_s\), context vector are fed to a fully connected layer to transform it into joint context vector \((ctx)\),

\[
ctx = W_{att}[ctx_v, ctx_s].
\]

The use of visual and semantic attention enables the language generator to decide where and when to focus while generating the report. Such techniques not only improve network performance, but also aid interpretability.

4. Language Generator: The main aim of the language generator network is to describe the observations in natural language. The language generator network consist of two sections: sentence LSTM and word LSTM. a) Sentence LSTM: Given the context vector as input, sentence LSTM generates the topic vectors and stop control probability over the two states \(\{\text{CONTINUE} = 0, \text{STOP} = 1\}\). The stop control probability determines whether to stop generating the topic vector or not, whereas the topic vector is
further fed to word LSTM. The sentence LSTM is made to continue based on the stop criterion, calculated using the current and the last hidden state producing a distribution. A pre-determined threshold is compared against the value given by the stop criterion, and a value greater than the threshold shall bring the operations of the sentence LSTM and word LSTM to a stop. No other topic vectors will be generated for this run by sentence LSTM. Sentence LSTM network comprises one LSTM Cell (Long short-term memory cell), fully connected layers, and finally, the tanh activation function is applied. The context vector prepared from visual and semantic vectors is fed to LSTMCell and two tensors, one tensor containing the initial hidden state and another having the initial cell state for each element in the batch. Linear layers are applied to the context vector and hidden state, and the result is combined. Further, tanh layer is applied to the combined output. A probability distribution is also calculated from the current and previous hidden state by applying linearity and tanh on the combined result of applying linearity on the two states, respectively. b) Word LSTM: The word LSTM generates one sentence on each topic, which are further combined to generate paragraphs. Taking the topic vector and <start> token as initial input, the subsequent inputs are learned embedding vectors from the ‘embedding layer’. Using the topics vector and the learned embedding vector, the LSTM layer predicts probability distribution over the vocabulary words, and accordingly, the next word is selected. In the next step, the predicted word is fed to next hidden layer of word LSTM to predict the next word, and repeated words are generated until the prediction of <end> token. More precisely, for each timestep, word LSTM computes probability as:

\[
P(\text{word}_{i+1} < \text{word}_{i,0} \ldots \text{word}_{i,j} >, M),
\]

where \(\text{word}_{i,j}\) is \(\text{word}_i\) of \(\text{sentence}_i\) and \(M\) model parameters. These predicted words are transformed onto the vocabulary using the fully connected layer. And the generated sentences are finally concatenated to get the resulting report. The overall training loss that is back-propagated is calculated as

\[
\text{loss} = \lambda_{\text{tag}} l_{\text{tag}} + \lambda_{\text{sent}} \sum_{i=1}^{S} l_{\text{sent}} + \lambda_{\text{word}} \sum_{j=1}^{S} \sum_{cap} l_{\text{word}}
\]

where \(\lambda_{\text{tag}}, \lambda_{\text{sent}}, \lambda_{\text{word}}\) are weights of losses \(l_{\text{tag}}, l_{\text{sent}}, l_{\text{word}}\) respectively.

4.3.2 Pruned model

To reduce the size and computational power of operating, the base model is to be compressed using pruning. The weight parameter details of each layer is summarized in Table 2. To decide the pruning strategy, we took decisions on the basis of the following questions:

1. What should be pruned? Instead of removing weights in groups, we used unstructured pruning that provides huge sparsity while maintaining high performance. Instead of local pruning the layers individually, we have used a more powerful pruning model all at once, i.e., global pruning. The advantage of this global pruning is that it will remove the desired percentage of least significant connections across the model rather than removing the desired percentage from each layer. In this pruning phase, the base model is globally pruned to remove \(p\) percent of connections with the lowest L1-norm; resulting in varying pruning percentages in each layer as per its significance.

2. When to prune? The model is pruned under two scenarios (discussed in Sect. 5): after training (AT) and within training (WT) of the base model, to check its efficacy in radiology report generation.

3. How much to prune? Rigorous experimentation has been performed with varying pruning percentages, as summarized in Table 3. Empirically study of the evaluation results conclude that the base model could be pruned up to 70% during training without sacrificing the model accuracy.

Algorithm 1 CheXPrune: The proposed model

Input: \(T\), the total number of training epochs
\(P\), epoch number when to prune
\(D\), the dataset
\(M\), the base model

Procedure:

\[
\text{M \leftarrow Initialize}
\]
\[
N \leftarrow 0
\]

while \(N \leq T\) do

if \(N == P\) then

\[
\text{M \leftarrow Prune}
\]

end if

\[
\text{M \leftarrow Train}
\]

\[
N \leftarrow N + 1
\]

end while

Stop

5 Experimental results and discussions

5.1 Ablation study and performance evaluation

We start by an attempt to use a base model to generate radiology reports from CXR images. As discussed in Sect. 4, VGG19 is used as feature extractor and HLSTM as language
generator. The results obtained by using visual, semantic and both attention mechanism are listed in Table 4. It can be clearly observed that attending both the visual and semantic features simultaneously gives us better results than attending those individually. Thus we used both attention mechanism in the base model (M1–BM). Further, we implemented several variants of the model to have a sparse model with similar efficacy to the base model. The empirical study of these variants results in the evolution of CheXPrune, the lighter model that can be operated in resource limited environment. We evaluated these variants based on following two scenarios:

**a) Scenario 1:** Pruning after training of Base Model (BM)

**b) Scenario 2:** Pruning the base model during training and fine-tuning it further

Based on above mentioned two scenarios, the following variants have been implemented to have the lightest possible efficient sparse model:

1. **M1 - BM:** This is the base model that uses VGG19 as feature extractor, multi-attention attention mechanism, and hierarchical lstm to generate a radiology report

2. **M2 - AT+20%:** In this variant, the base model is completely trained, and then 20% of weight parameters have been pruned to remove redundancy

| Index | Layer         | Parameter names       | Type        | #Parameters |
|-------|---------------|-----------------------|-------------|-------------|
| 0     | Encoder Layer | conv1_1.weight         | Convolution | 1728        |
| 1     |               | conv1_2.weight         | Convolution | 36864       |
| 2     |               | conv2_1.weight         | Convolution | 73728       |
| 3     |               | conv2_2.weight         | Convolution | 147456      |
| 4     |               | conv3_1.weight         | Convolution | 294912      |
| 5     |               | conv3_2.weight         | Convolution | 589824      |
| 6     |               | conv3_3.weight         | Convolution | 589824      |
| 7     |               | conv3_4.weight         | Convolution | 589824      |
| 8     |               | conv4_1.weight         | Convolution | 1179648     |
| 9     |               | conv4_2.weight         | Convolution | 2359296     |
| 10    |               | conv4_3.weight         | Convolution | 2359296     |
| 11    |               | conv4_4.weight         | Convolution | 2359296     |
| 12    |               | conv5_1.weight         | Convolution | 2359296     |
| 13    |               | conv5_2.weight         | Convolution | 2359296     |
| 14    |               | conv5_3.weight         | Convolution | 2359296     |
| 15    |               | conv5_4.weight         | Convolution | 2359296     |
| 16    | MLC           | classifier.weight      | Linear      | 107520      |
| 17    |               | embed.weight           | Embedding   | 107520      |
| 18    | Sentence LSTM | vis_att.enc_att.weight | Linear      | 32768       |
| 19    |               | vis_att.dec_att.weight | Linear      | 32768       |
| 20    |               | vis_att.full_att.weight| Linear      | 64          |
| 21    |               | sem_att.enc_att.weight | Linear      | 32768       |
| 22    |               | sem_att.dec_att.weight | Linear      | 32768       |
| 23    |               | sem_att.full_att.weight| Linear      | 64          |
| 24    |               | contextLayer.weight    | Linear      | 524288      |
| 25    |               | lstm.weight_ih          | LSTMCell    | 2097152     |
| 26    |               | lstm.weight_hh          | LSTMCell    | 1048576     |
| 27    |               | topic_hid_layer.weight  | Linear      | 262144      |
| 28    |               | topic_context_layer.weight | Linear    | 524288      |
| 29    |               | stop_prev_hid.weight   | Linear      | 32768       |
| 30    |               | stop_cur_hid.weight     | Linear      | 32768       |
| 31    |               | final_stop_layer.weight | Linear      | 128         |
| 32    | Word LSTM     | embedding.weight       | Embedding   | 1055744     |
| 33    |               | lstm.weight_ih_l0       | LSTM        | 1048576     |
| 34    |               | lstm.weight_hh_l0       | LSTM        | 1048576     |
| 35    |               | fc.weight               | Linear      | 1055744     |
M3- AT+50%: In this variant, M1 is trained and then pruned for 50% of weight parameters.

M4- AT+70%: In this variant, M1 is trained and then pruned for 70% of weight parameters.

M5- AT+80%: In this model, 80% of weight parameters are pruned after training of the base model, M1.

M6- AT+90%: This is the 90% pruned model of the trained Base model, M1.

M7- WT+20%: From the model M7 to M11, pruning is applied after 35 epochs during training of the base model, M1. In this variant, 20% weight parameters are pruned after 35 epochs of training of M1 and then fine-tuned to retain the dropped accuracy.

M8- WT+50%: In this variant, 50% weight parameters are pruned after 35 epochs of training of M1 and then fine-tuned to retain the dropped accuracy.

M9- WT+70% (CheXPrune): In this variant 70% weight parameters are pruned after 35 epochs of training of M1, and then fine-tuned to retain the dropped accuracy.

M10- WT+80%: In this variant, the weight parameters are pruned for 80% after 35 epochs of training of M1, and then the sparse model is fine-tuned.

M11- WT+90%: In it, 90% of the weight parameters are pruned after 35 epochs of training of M1.

The performance evaluation results of all the variants mentioned above are shown in Fig. 7, and along with pruning strategy has been discussed below:

**Scenario 1: Pruning after complete training**

In the first scenario, we completely trained the base model, M1, training and validation loss curve is shown in Fig. 7a. Before testing, we pruned the trained model globally with varying pruning percentages and analysed the evaluation metric scores as shown in Fig. 7k. It is clearly observed in the same figure that the trained model can be pruned up to 50% without sacrificing the model accuracy. In contrast, it starts degrading after that and vanishes at 90% pruning. The pruning percentage induced in each layer of the model, as shown in Fig. 7c, are also analysed to determine their importance. More specifically, the encoder layer is proven to be more redundant; thus, maximum sparsity is induced in the encoder layer. We observed that the initial layers of the encoder are less pruned, which enables to retain the model’s accuracy. Also, embedded layers of the model are least pruned, hence are the most significant layers in the model.

Thus, globally pruning the trained model can accurately work up to 50% pruning level.

**Scenario 2: Pruning with fine-tuning during training (CheXPrune)**

During training the model, as shown in Fig. 7a, we observed that model started converging around 30-35 epochs. Hence under Scenario 2 we experimented by first training the model for less than 35 epochs and then applied one-shot pruning with a varying percentage at epoch 35. Post pruning, model performance is reduced; thus, we fine-tune the model till it converges again. The training- validation curve over epochs, Fig. 7b, shows that the model works fine up to 50% one-shot global pruning whereas its performance is reduced after that. The accuracy dropped at 70% pruning level, but after fine-tuning the model, we are able to achieve a similar level of performance. Fine-tuning the model after pruning it above 70% was unable to achieve the desired results. Thus, pruning the model before convergence can accurately work up to 70% pruning level, resulting in 3.33x compression ratio.

Similar to Fig. 7c, pruning percentage induced in different layers of the model while pruning with fine-tuning is shown in Fig. 7d. The scores of several evaluation metrics have been summarized in Fig. 7e–k.
6 Baseline comparisons

We compared CheXPrune with the state-of-the-art methods proposed by Harzig et al. (2019); Huang et al. (2019); Jing et al. (2020, 2017); Li et al. (2018, 2019a, 2019b); Liu et al. (2021); Wang et al. (2018, 2020); Xiong et al. (2019); Yin et al. (2019); Yang et al. (2021) and Zhang et al. (2020). The comparison of evaluation metric scores are listed in Table 5 and it shows that CheXPrune outperforms the state-of-the-art methods in most of the evaluation metric scores.

7 Conclusion and future work

In this paper, we have presented a novel multi-attention based pruned radiology report generation method (CheXPrune). CheXPrune is a deep neural network having two modules: the base model and pruning phase. The base model is a variant of encoder decoder architecture, in which both the visual and semantic features are focused through the multi-attention mechanism for generating context vector, which is further used by hierarchical LSTM for generating chest radiological
report automatically. We have empirically shown that the base model can be pruned up to 70% without sacrificing the model’s performance by using one-shot pruning technique within the training. The structured pruning techniques can be explored to select the least required sub-network to be pruned. The unstructured pruning introduces a huge amount of sparsity, and several software and hardware, e.g., tensorlite, make optimized moves to ignore this sparsity and improve the inference to a high level. Thus, deploying the CheXPrune onto the low computation devices is the future scope of this work.

The proposed ARRG system can be very beneficial in situations when bulk reporting is required such as sudden outbreak of pandemic COVID-19. Such situation pressurizes healthcare workers to deal with enormous amount of medical data and instant reporting can smoothen the clinical workflow. Several researchers have developed interesting techniques (e.g., Shui-Hua et al. 2022; Wang et al. 2021, 2022) that function admirably for identifying and categorising COVID-19 medical images. The feature extractor and language generator are the two modules that make up the basic model given in this study. As a feature extractor, we employed VGG19 convolutional layers. In the future, VGG19 in our base model can be swapped out for more effective feature extraction modules such as used in methods developed by Shui-Hua et al. (2022); Wang et al. (2021); and Wang et al. (2022). The same architecture can then be investigated in order to produce medical reports for additional modalities.

### Acknowledgements
The first author is Visvesvaraya research fellow and her research is funded by Ministry of Electronics and Information Technology (Meity), Government of India, New Delhi, India.

### References
Alfarghaly O, Khaled R, Elkorany A, Helal M, Fahmy A (2021) Automated radiology report generation using conditioned transformers. Inform Med Unlocked 24:100557
Bengio Y, Louradour J, Collobert R, Weston J (2009) Curriculum learning. In: Proceedings of the 26th annual international conference on machine learning, pp 41–48
Blakeney C, Yan Y, Zong Z (2020) Is pruning compression?: Investigating pruning via network layer similarity. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp 914–922
Blalock D, Ortiz JIG, Frankie J, Guttag J (2020) What is the state of neural network pruning? arXiv:2003.03033
Changchang Y, Buyue Q, Jishang W, Xiaoyu L, Xianli Z, Yang L, Qinghua Z (2019) Automatic generation of medical imaging diagnostic report with hierarchical recurrent neural network. In: 2019 IEEE International Conference on Data Mining (ICDM), pp 728–737
Chen Z, Song Y, Chang TH, Wan X (2020) Generating radiology reports via memory-driven transformer. arXiv:2010.16056
Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: a large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition, pp 248–255
Dina D-F, Sameer A, Matthew S, Thoma George R (2012) Design and development of a multimodal biomedical information retrieval system. J Comput Sci Eng 6(2):168–177
Frost S (2015) Cognitive computing and artificial intelligence systems in healthcare. https://store.frost.com/cognitive-computing-and-artificial-intelligence-systems-in-healthcare.html
Harzig P, Chen YY, Chen F, Lienhart R (2019) Addressing data bias problems for chest x-ray image report generation. arXiv:1908.02123
He X, Deng L (2018) Deep learning in natural language generation from images. In: Deep Learning in Natural Language Processing. Springer, New York, pp 289–307
He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778

| Methods                          | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 | ROUGE | CIDEr |
|----------------------------------|--------|--------|--------|--------|-------|-------|
| Harzig et al. (2019)             | 0.373  | 0.246  | 0.175  | 0.126  | 0.359 | 0.315 |
| Huang et al. (2019)              | 0.476  | 0.340  | 0.238  | 0.169  | 0.297 | 0.347 |
| Jing et al. (2020)               | 0.464  | 0.301  | 0.210  | 0.154  | 0.275 | 0.362 |
| Jing et al. (2017)               | 0.517  | 0.386  | 0.306  | 0.247  | 0.447 | 0.327 |
| Li et al. (2018)                 | 0.438  | 0.298  | 0.208  | 0.151  | 0.322 | 0.343 |
| Li et al. (2019b)                | 0.419  | 0.280  | 0.201  | 0.150  | 0.553 | 0.371 |
| Li et al. (2019a)                | 0.482  | 0.325  | 0.226  | 0.162  | 0.280 | 0.339 |
| Liu et al. (2021)                | 0.512  | 0.327  | 0.240  | 0.179  | 0.383 |       |
| Wang et al. (2018)               | 0.286  | 0.159  | 0.103  | 0.073  | -     | 0.226 |
| Wang et al. (2020)               | 0.503  | 0.333  | 0.236  | 0.175  | 0.360 | 0.331 |
| Xiong et al. (2019)              | 0.350  | 0.234  | 0.143  | 0.096  |       | 0.323 |
| Yin et al. (2019)                | 0.445  | 0.292  | 0.201  | 0.154  | 0.344 | 0.342 |
| Yang et al. (2021)               | 0.471  | 0.336  | 0.238  | 0.166  | 0.382 | 0.345 |
| Zhang et al. (2020)              | 0.441  | 0.291  | 0.203  | 0.147  | 0.304 | 0.367 |
| CheXPrune (M9-WT+70%)            | 0.5428 | 0.4451 | 0.3737 | 0.3197 | 0.5976| 0.3215|

Best scores are highlighted in bold

Table 5 Performance comparison with state-of-the-art ARRG systems
Huang X, Yan F, Wei X, Li M (2019) Multi-attention and incorporating background information model for chest X-ray image report generation. IEEE Access 7:154808–154817
Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4700–4708
Jaiswal AK, Tiwari P, Kumar S, Gupta D, Khanna A, Rodrigues JPC (2019) Identifying pneumonia in chest X-rays: a deep learning approach. Measurement
Jing B, Wang Z, Xing E (2020) Show, describe and conclude: on exploiting the structure information of chest X-ray reports. arXiv:2004.12274
Jing B, Xie P, Xing E (2017) On the automatic generation of medical imaging reports. arXiv:1711.08195
Krause J, Johnson J, Krishna R, Fei-Fei L (2017) A hierarchical approach for generating descriptive image paragraphs. In: Computer Vision and Pattern Recognition (CVPR)
LeCun Y, Denker JS, Solla SA (1990) Optimal brain damage. In: Krause J, Johnson J, Krishna R, Fei-Fei L (2017) A hierarchical approach for generating descriptive image paragraphs. In: Computer Vision and Pattern Recognition (CVPR)
LeCun Y, Denker JS, Solla SA (1990) Optimal brain damage. In: Krause J, Johnson J, Krishna R, Fei-Fei L (2017) A hierarchical approach for generating descriptive image paragraphs. In: Computer Vision and Pattern Recognition (CVPR)
LeCun Y, Denker JS, Solla SA (1990) Optimal brain damage. In: Krause J, Johnson J, Krishna R, Fei-Fei L (2017) A hierarchical approach for generating descriptive image paragraphs. In: Computer Vision and Pattern Recognition (CVPR)
Lin CY (2004) Rouge: a package for automatic evaluation of summaries. In: Text summarization branches out, pp 74–81
Liu F, You C, Xian W, Shen GX et al (2021) Auto-encoding knowledge graph for unsupervised medical report generation. Adv Neural Inform Process Syst 34:16266–16279
Liu G, Hsu TMH, McDermott M, Boag W, Weng WH, Szolovits P, Ghassemi M (2019) Clinically accurate chest X-ray report generation. arXiv:1904.02633
Lovelace J, Mortazavi B (2020) Learning to generate clinically coherent chest X-ray reports. In: Proceedings of the 2020 Conference on empirical methods in natural language processing: findings, pp 1235–1243
Malach E, Yehudai G, Shalev-Schwartz S, Shamir O (2020) Proving the lottery ticket hypothesis: pruning is all you need. In: International Conference on Machine Learning, pp 6682–6691
Martijn Nobel J, Kok EM, Robben SGP (2020) Redefining the structure of structured reporting in radiology. Insights Imaging 11(1):1–5
Noorahalzadeh F, Gonzalez NP, Frauenfelder T, Fujimoto K, Krauthammer M (2021) Progressive transformer-based generation of radiology reports. arXiv:2102.09777
Papineni K, Roukos S, Ward T, Zhu WJ (2002) Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics, pp 311–318
Rajpurkar P, Irvin J, Zhu K, Yang B, Mehta H, Duan T, Ding D, Bagul A, Langlotz C, Shpanskaya K et al (2017) CheXnet: radiologist-level pneumonia detection on chest X-rays with deep learning. arXiv:1711.05225
Shin HC, Roberts K, Lu L, Demner-Fushman D, Yao J, Summers RM (2016) Learning to read chest X-rays: recurrent neural cascade model for automated image annotation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2497–2506
Shui-Hua W, Khan MA, Govindaraj V, Fernandes SL, Zhu Z, Yu-Dong Z (2022) Deep rank-based average pooling network for Covid-19 recognition. Comput Mater Continua 2797–2813
Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556
Vedantam R, Zitnick CL, Parikh D (2015) Cider: consensus-based image description evaluation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4566–4575
Wang S-H, Zhang X, Zhang Y-D (2021) Dssae: deep stacked sparse autoencoder analytical model for Covid-19 diagnosis by fractional fourier entropy. ACM Trans Manage Inform Syst (TMIIS) 13(1):1–20
Wang W, Zhang X, Wang S-H, Zhang Y-D (2022) Covid-19 diagnosis by we-saj, Syst Sci Control Eng 10(1):325–335
Wang F, Liang X, Xu L, Lin L (2020) Unifying relational sentence generation and retrieval for medical image report composition. IEEE Trans Cybern
Wang X, Peng Y, Lu L, Lu Z, Summers RM (2018) Tienet: text-image embedding network for common thorax disease classification and reporting in chest X-rays. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 9049–9058
Xiong Y, Du B, Yan P (2019) Reinforced transformer for medical image captioning. In: International Workshop on Machine Learning in Medical Imaging. Springer, New York, pp 673–680
Xu K, Ba J, Kiros R, Cho K, Courville A, Salakhudinov R, Zemel R, Bengio Y (2015) Show, attend and tell: Neural image caption generation with visual attention. In: International conference on machine learning, pp 2048–2057
Yang Y, Teo CL, Daumé III H, Aloimonos Y (2011) Corpus-guided sentence generation of natural images. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, pp 444–454
Yang X, Ye M, You Q, Ma F (2021) Writing by memorizing: hierarchical retrieval-based medical report generation. arXiv:2106.06471
Yuan J, Liao H, Luo R, Luo J (2019) Automatic radiology report generation based on multi-view image fusion and medical concept enrichment. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, New York, pp 721–729
Zhang MS, Stadie B (2019) One-shot pruning of recurrent neural networks by Jacobian spectrum evaluation. arXiv:1912.00120
Zhang Y, Zhang Y, Qi P, Manning CD, Langlotz CP (2021) Biomedical and clinical English model packages for the Stanza Python NLP library. J Am Med Inform Assoc 28(9):1892–1899. https://doi.org/10.1093/jamia/ocab090
Zhang Y, Wang X, Xu Ziyue, Yu Q, Yuille A, Xu D (2020) When radiology report generation meets knowledge graph. arXiv:2002.08277
Zhu M, Gupta S (2017) To prune, or not to prune: exploring the efficacy of pruning for model compression. arXiv:1710.01878

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.