Reproducing and Analyzing Baselines for Chest X-Ray Report Generation: Insights and Challenges
Jimeng Sun

Written by Hong Wu, Yinzhe Luo
University of Illinois at Urbana Champaign
201 North Goodwin Avenue
Urbana, Illinois 61801
admin@cs.illinois.edu

Abstract

Link to presentation video —
https://drive.google.com/file/d/1KIeM1Y0xCpH-_UOKOd6PHbLlOYts6fEK/view?usp=sharing
Link to Github Repo —
https://github.com/hwu5542/cxr-baselines

Introduction

The writing of radiology reports, especially for chest X-rays (CXRs), is a routine but time-consuming part of clinical practice. Since CXR imaging is widely used, the number of reports generated daily is substantial. These reports require domain-specific expertise, and automating the process could help ease the burden on radiologists while speeding up diagnosis.

The paper "Baselines for Chest X-Ray Report Generation" by Boag et al. (2020) addresses several knowledge gaps in previous research in clinical radiology reports from chest X-ray images. In particular, there are no standard baselines for the generation of chest radiographs. Standard NLG metrics prefer grammatically correct over clinical accuracy. Good grammar reports score higher than clinical n-gram reports in BLEU or CIDEr. CheXpert prefers clinical accuracy without readability. Previous research was focusing on complex neural models. Public datasets before MIMIC-CXR lack of paired X-ray images with reports in English. Under this background, the author conducted the study on MIMIC-CXR to obtain a baseline model that meets both the standard of NLG metrics and clinical precision. The study establishes robust baselines for future research, advocating for the inclusion of simple yet effective methods like nearest neighbor retrieval alongside advanced neural models.

In our project, we explored a small extension by including a BERT-based decoder in the original CNN-RNN pipeline. The idea was to see if adding a modern pre-trained language model would improve how natural the reports sounded or whether it helped with capturing key medical points. We were curious to see whether a pre-trained language model could improve either the readability or the diagnostic usefulness of the generated text. Throughout the project, we paid close attention to the details of implementation. We used observable pattern in the program to keep track of where things worked as expected and where they didn't. In the purpose to understand the reproducibility of the original work with help of AI and to efficiently tune the code to serve our analysis.

Citation to the original paper

Boag, W., Hsu, T. M., McDermott, M., Berner, G., Alsentzer, E., & Szolovits, P. (2020). Baselines for Chest X-Ray Report Generation. Proceedings of Machine Learning Research, 116, 126-140. ML4H at NeurIPS 2019.

Scope of Reproducibility

In this reproducibility study, we replicated core components of the original work, including data loading and preprocessing, image and report parsing, implementation of baseline models (random retrieval, first nearest neighbor, n-gram, and CNN-RNN with BERT enhancements), and evaluation using both CheXpert and NLG metrics. Our results were fairly close to what the original paper reported, drawing similar conclusion. We had a consistent setup with the author. And through the project, it gave us a better understanding of the challenges involved.

- **Dataset processing:** We processed the first two partitions of MIMIC-CXR dataset. Tested on 476 GB of data, a total of 26,505 radiographic studies out of the 227,835 records provided by MIMIC-CXR. One tenth of the original tested dataset. Splitting it into 21,070 training and 5,435 test samples. we read chest X-ray images in DICOM format and resized them to 224×224 pixels. The images were normalized to match DenseNet input expectations. For the reports, we applied a text-cleaning process that removed sensitive identifiers and standardized formatting. We isolated the "Findings" section and extracted the labels using a parser. Each image was paired with its corresponding cleaned report labels, and the dataset was split into training and test sets.
- **Models:** 4 models, random, n-gram(1-, 2-, and 3-grams), 1-NN, and CNN-RNN. All models from the original paper were reconstructed, with random retrival model being the baseline model. Their performance were evaluated and validated against the original findings.

- **Evaluation pipeline:** For evaluation, we used standard NLG evaluation (BLEU, CIDEr) for grammar score and CheXpert labeler for their clinical accuracy.

Methodology

Environment

Python: 3.9.18
TensorFlow with GPU: 2.10.1
Cuda: 12.8
cuDNN: 9.9.0
Docker: 28.0.4

Dependencies:

- **numpy**, **pandas** – for numerical operations and data manipulation.
- **pydicom**, **pillow (PIL)** – for loading and preprocessing DICOM images.
- **torch**, **torchvision** – for implementing and training deep learning models.
- **scikit-learn** – for splitting the dataset and computing evaluation metrics.
- **tqdm** – for displaying training progress bars.
- **matplotlib** – for plotting results and data inspection.

Data

Dataset Access and Download:

We used the MIMIC-CXR v2.1.0 dataset, which contains chest radiographs in DICOM format and associated radiology reports. The dataset is hosted on PhysioNet and requires credentialed access through a data use agreement process. To use the dataset, user must complete the CITI "Data or Specimens Only Research" course and agree to the terms of use before downloading via https://physionet.org/content/mimic-cxr/2.1.0/. Upon gaining the access to the dataset, simply following the instruction on GitHub to make use of the downloaded data.

For evaluation, we also used the CheXpert labeler released by Stanford, which is available at CheXpert GitHub repository via https://github.com/stanfordmlgroup/chexpert-labeler. Our project includes a local version of the labeler under the `evaluate/` folder. It uses NegBio-based pattern matching to extract 14 clinical labels (e.g., Cardiomegaly, Edema, Pleural Effusion) from free-text reports.

Dataset Description:

The MIMIC-CXR dataset contains 377,110 chest X-ray images from 227,835 radiographic studies. The studies were conducted at the Beth Israel Deaconess Medical Center in Boston, MA. Each study is accompanied by a radiology report. Each has a findings section that contains the diagnostic information. Over the two month duration, we have downloaded 476 GB of data from PhysioNet. Through data preprocessing, we filtered anteroposterior(AP) views, removed brightness/contracts adjusted duplicate radiographs and select only studies with valid image and the findings section in the reports to further compress the data size. This will yield a final size of 26,505 (train: 21,070, test: 5,435) with no patient overlapping between splits. Also employing the parquet storage format, we further pruned the size to 3.65 GB. Reducing more than 99.23% of the data size. We utilized the pretrained DenseNet121 encoder from ChestX-ray14 to extract 1024-dimensional features. Also adding rule-based labeler for 14 clinical categories with around 3 sec/report CPU overhead in data loading and parsing.

Image	Patient ID	Study ID	Pooled Features	Findings
[1.24, 0.25...]	10718588	55138110	[3.6e-05, -4.9e-03...]	Leftward tracheal deviation, pulmonary edema, atelectasis...
[2.08, 2.08...]	10718588	55782674	[1.8e-04, -2.1e-03...]	Improved pulmonary congestion, severe cardiomegaly...
[0.46, 0.33...]	10718588	55782674	[-5.0e-06, -1.7e-03...]	Improved pulmonary congestion, no effusion...
[2.12, 2.11...]	10718588	59509348	[-1.3e-04, -4.1e-03...]	Low lung volumes, pulmonary edema improved...
[0.90, 0.37...]	10718588	59509348	[-3.7e-05, 1.9e-03...]	Low lung volumes, stable cardiomegaly...

Table 1: Sample parsed MIMIC CXR in DataFrame

Model

Link to Github Repo of Original Paper —
https://github.com/wboag/cxr-baselines

In the paper *"Baselines for Chest X-Ray Report Generation"*, there are four baseline models proposed, random, n-gram, 1-nearest neighbor(retrieval) and CNN-RNN. Therefore in our project, we same Included these models.

1. Random : The Random model selects a report $\hat{R}$ uniformly at random from the training set:

$$\hat{R} \sim \mathcal{U}(R_{\text{train}})$$

This Model ignores the input and it serves as naive lower bound baseline for our comparison.

2. n-gram : The n-gram model tokenizes each report into n-grams (1-gram, 2-gram, or 3-gram) and retrieves the report from the training set that shares the most overlapping n-grams. Equation used:

$$\hat{R} = \arg \max_{R_i \in R_{\text{train}}} \text{sim}_{\text{n-gram}}(R_i, R_I)$$

Here, $\text{sim}_{\text{n-gram}}$ is a function that counts shared n-gram tokens. We use Unigram, Bigram and Trigram to generate reports based on the successive word occurrence, using the closest 100 training images in DenseNet121.

3. 1-Nearest Neighbor (1-NN): This model converts each report into a vector using method TF-IDF, which captures how often certain words appear. Then, when a new test report comes in, the model compares it to every other reports in the training set using cosine similarity (a measure of how close the vectors are in direction). After that, the report that from the training set is most similar (has the highest cosine score) is returned as the prediction as the final. Both grammatical and clinical accuracy should increase

$$\hat{R} = \arg \max_{R_i \in R_{\text{train}}} \frac{f(R_I) \cdot f(R_i)}{\|f(R_I)\| \|f(R_i)\|}$$

Here, $f(R)$ is the vector form of a report. This method is often effective in capturing the medical content of reports, as it try to matches based on shared wording and terminology.

4. CNN-RNN Encoder-Decoder: Implementation in file `cnn_rnn.py`, this model consists of two main parts:

CNN Encoder: project the output from 1024 dimensions to 256 dimensions:

$$\mathbf{v}_{\text{img}} = \text{GAP}(\text{DenseNet121}(I)) \in R^{1024}$$

This vector is then projected into a 256-dimensional latent space:

$$\mathbf{z}_{\text{img}} = W_{\text{proj}} \cdot \mathbf{v}_{\text{img}} + b_{\text{proj}}, \quad \mathbf{z}_{\text{img}} \in R^{256}$$

RNN Decoder: The decoder is a unidirectional LSTM that receives token embeddings and the image vector. At each time step t:

$$\mathbf{h}_t, \mathbf{c}_t = \text{LSTM}(\mathbf{e}_{t-1}, \mathbf{h}_{t-1}, \mathbf{c}_{t-1})$$

The decoder generates vocabulary probabilities using:

$$\mathbf{y}_t = \text{softmax}(W_o \cdot \mathbf{h}_t + b_o)$$

Using cross-entropy loss per token as Optimizer. Epoch: 64. Learning Rate: 1×10^{-3}, decaying by half every 16 epochs. Teacher Forcing: feeding ground-truth tokens to decoder initially, replace 5% to feeding predictions per 16 epochs.

5. CNN-RNN-BERT Hybrid: Implementation in file `cnn_rnn_bert.py`, this is a extended model that replaces the original LSTM decoder in the CNN-RNN architecture with a transformer-based decoder using a pretrained BERT model from HuggingFace. This model's goal is to improve linguistic fluency and contextual coherence.

Processed through DenseNet121 (pretrained on ChestX-ray14) and mapped into a vector $\mathbf{z}_{\text{img}} \in R^{256}$. Then fused into the input sequence of the BERT model. The BERT decoder takes in the image context along with token embeddings and produces contextualized representations at each step.

$$\mathbf{H} = \text{BERT}([\mathbf{z}_{\text{img}}, \mathbf{e}_1, \ldots, \mathbf{e}_n])$$
$$\hat{y}_t = \text{softmax}(W \cdot \mathbf{h}_t + b)$$

Retrieval Models (Random, n-gram, 1-NN)

- **Input:**
  - A test image I (text or features optional)
- **Output:**
  - A full report $\hat{R}$ retrieved from the training set
- **Techniques:**
  - Uniform sampling (Random)
  - Token-level n-gram matching (n-Gram)
  - Cosine similarity in sparse vectors (1-NN)

CNN-RNN Model

- **Input:**
  - DICOM image $I \in R^{224 \times 224}$
  - Token sequence $T = (t_1, \ldots, t_n)$
- **Output:**
  - Generated token sequence $\hat{T} = (\hat{t}_1, \ldots, \hat{t}_m)$
- **Techniques:**
  - Image encoding via pretrained DenseNet121
  - Projection to compact latent space
  - LSTM-based decoder with teacher forcing
  - Greedy decoding at inference

CNN-RNN-BERT Model

- **Input:**
  - DICOM image $I \in R^{224 \times 224}$
  - Partial token sequence $T = (t_1, \ldots, t_n)$
- **Output:**
  - Predicted sequence $\hat{T} = (\hat{t}_1, \ldots, \hat{t}_m)$
- **Techniques:**
  - Pretrained DenseNet121 image encoder
  - Tokenization/decoding with `TFBertModel`
  - Image feature fusion with token embeddings
  - Teacher forcing training; greedy decoding

Pretrained Model: The DenseNet121 encoder was initialized with ImageNet weights. During training, all convolutional layers were frozen to preserve general visual features.

Training Configuration

Hyperparameters

- **Learning Rate:** 1×10^{-3} (halved every 16 epochs)
- **Batch Size:** 32 samples
- **Model Architecture:**
  - 256 LSTM units
  - 256-dimensional word embeddings
- **Training Duration:** 64 epochs
- **Sequence Generation:**
  - Max sequence length: 100 tokens
  - Beam search width: 4 candidates
  - Length normalization: score/(len(seq)$^{0.7}$)
- **Teacher Forcing:**
  - Initial rate: 100%
  - Reduction: 5% every 16 epochs

Computational Resources

- **Hardware:** NVIDIA GeForce RTX 3080 Ti (10GB VRAM)

- **Datasets:**

	Training Samples	Test Samples
Small	6,566	1,620
Medium	9,767	2,419
Large	21,070	5,435

- **Training Times:**
  - Small: 3–4 minutes/epoch
  - Medium: 5–7 minutes/epoch
  - Large: 12–15 minutes/epoch
- **Total GPU Time:** ∼24 hours (all trials)

Training Methodology

- **Loss Function:** Sparse categorical crossentropy
- **Optimizer:** Adam with exponential learning rate decay
- **Training Procedure:**
  - Initial phase: 100% teacher forcing
  - Scheduled reduction: 5% fewer ground-truth tokens every 16 epochs
  - Gradual transition to prediction-based decoding

Evaluation

Metrics Used:

In this project, we included three metrics for evaluation for the assessment of models. In linguistic and clinical :

- **BLEU-4:** A precision-based metric that compares n-gram overlap (up to 4-grams) between the generated report and the reference report. The Higher BLEU scores, the better linguistic similarity.
- **CIDEr:** Originally designed for image captioning tasks, CIDEr weights n-gram matches using TF-IDF scores and compares a candidate sentence to a set of reference sentences. Contents that words are more informative are more favor and generic language in contrast.
- **CheXpert F1:** CheXpert labeler is used to both generated and ground-truth reports to extract 14 clinical labels. Then it is used to compute the macro-averaged F1 score. This metric focuses on the medical correctness the generated output. It is independent from grammar.

Results

Overview of Reproduced Performance

We reproduced and evaluated 4 baseline models: random, n-gram (1–3), 1-Nearest Neighbor (1-NN), and CNN-RNN with a transformer-based decoder extension (CNN-RNN-BERT). Evaluation metrics included BLEU (1–4), CIDEr, and CheXpert clinical metrics: Accuracy, Precision, and F1. See Table 2 in appendix for detail.

CNN-RNN-BERT models performed the best in BLEU and CIDEr, but it's CheXpert Precision and F1 score are nearly zero. We suspect a disconnect between fluency and clinical correctness.

Label-Level Clinical Accuracy (CheXpert F1)

Table 3 in appendix shows F1 scores across all 13 CheXpert labels. 1-NN outperformed other models on nearly all labels. CNN-RNN-BERT performed poorly in almost all medical classes.

Comparison with Original Paper

Our reproduced results show a similar trend with some noticeable difference in variations of metric values.

In our reproduction, 1-NN achieved the highest F1 score (0.689), same as the original paper but with much higher score. However, for our CNN-RNN-BERT models, while achieving competitive BLEU scores, consistently underperformed on CheXpert Precision and CheXpert F1.

For n-gram models, particularly 2-gram and 3-gram, performed better than the original paper But random and 1-gram models not so well in comparison.

Despite some perofrmance differences in differnt model, overall we think our reproduction have similar trend as the original paper: simple retrieval-based models such as 1-NN is a strong baselines and often outperform more complex generative models in clinical correctness. Our reproduction reinforce the author's idea of the importance of considering both language and clinical evaluation metrics when evaluating the effectiveness of the automatic report generation system.

Extension: Transformer Decoder

We extended the CNN-RNN model by replacing the RNN decoder with a pretrained BERT model. It improved performance on BLEU and CIDEr , but failed to capture any meaningful clinical signal. Future work is needed:

- BioBERT or ClinicalBERT initialization
- Label-aware training (e.g., auxiliary CheXpert loss)
- Constraining decoding with clinical priors or structured prompts

Discussion

Implications of Experimental Results

In our reproduction, we are able to reproduce some of the core ideas of the original paper. Simpler robust retrieval-based models such as 1-NN can outperform complex generative models in clinical utility, even if it generates less fluent text. Our best-performing model, 1-NN, scored a CheXpert F1 of 0.689, much higher than original report's 1-NN scores. we think this strongly reinforces the idea of lexical fidelity often correlates more strongly with clinical accuracy than fluency alone.

However, our extended CNN-RNN-BERT models, despite incorporating pretrained language models and achieving decent BLEU scores, failed to reproduce clinically interpretable results under CheXpert evaluation. These findings highlight a potential tradeoff in medical NLG between language modeling and domain-specific correctness: Strong fluency does not guarantee clinical relevance.

Reproducibility Assessment

Overall, we think the original paper is largely reproducible. We were able to reconstruct the all the baseline models, follow the data processing pipeline, and evaluate results with the same metrics on a smaller dataset. The authors' GitHub repository was instrumental in bridging undocumented implementation details. However, due to the source code being

the initial build but not a completed version, there were a significant amount of knowledge gap in preprocessing, parsing, model training, and evaluating. Our reconstructed version of the parsed dataset and models were based closer to the description of the paper instead of the source code. This leads to diverging performance in our CNN-RNN-BERT models.

For the part irreproducible, we are uncertain about how CheXpert-compatible phrasing was handled in the decoder output. Without aligning formatting with the CheXpert labeler's expectations, even linguistically plausible sentences is what we end up with, which we think is what contributed to the result of zero F1 scores our CNN-RNN-BERT models.

What Was Easy?

For us, re-implement the random, n-gram, and 1-NN baseline models was straightforward. Althought, we were given the access to the original code from author's github, it only provided a vague framework of orchestration, giving us utmost freedom in detailed implementation. Theses models are conceptually simple and the essential parts were well documented and described in the original paper. Also, Evaluating BLEU and CIDEr scores was also easy to replicate using standard libraries.

What Was Difficult?

For us, a challenging aspect was aligning our model outputs with the expectations of the CheXpert labeler. The labeler is sensitive to formatting, punctuation, and phrase structure, even small discrepancies in report generation (e.g., missing periods or word order changes) could result in zero-valued F1 scores. Additionally, some hyperparameters and preprocessing steps (e.g., exact regex patterns used to extract "Findings") were either undocumented or required guesswork. The biggest challenge however, is reusing outdated libraries that does not support GPU bulk processing. The bllipparser library contains .C files that are required to be run in a linux system. By creating a container we were able to run the CheXpert evaluation, but with 30% CPU usage and gigantic amount of RAM usage. For evaluation of 5,435 records, it took more than one week to process all 13 model varients. Each variant were taking more than five hours to evaluate. And the WSL 2 needed to be manually restarted in between each varient to free up the memory from loaded models in the C code. Many trials and failures were also taken to fight the ginormous memory usage problem. To prevent the container constainly exceeding the momory limitation, we broke the evaluation into partitions and run it one by one.

Recommendations for Improving Reproducibility

To support more reproducible research in medical report generation, we suggest the following:

- GPU Usage: Updating the source code and dependencies to fully embrace GPU processing of model training as well as evaluations.
- Version metrics and labelers: Evaluation tools needs a overhaul, specifying the exact commit or version of external tools (like CheXpert, bllipparser) helps improve process efficiency and ensure stable benchmarking across reproductions.
- Provide a validated report formatter: Clinical evaluation depends strictly on text structure. Including a post-processor or formatter to ensure compatibility with evaluation labelers like CheXpert would prevent model outputs from being unfairly penalized.
- Document full preprocessing pipelines: Dataset splits, cleaning scripts, regex filters, and specific text fields used. A full data-to-model script without ambiguities would be great.

In conclusion, the original paper study's findings are largely reproducible. But to achieve consistent evaluation results remains heavily dependent on implementation and subtleties in formatting, processing. Standardization in preprocessing and evaluation is key to robust medical NLG research.

Author Contributions

Hong Wu: Project proposal, Project Paper and presentation Revising, Data Loading and Preprocessing, data parser, Baseline Models(random retrieval, First Nearest Neighbor, ngram, CNN-RNN Bert), chexpert evaluation, NLG evaluation, pyHealth PR

Yinzhe Luo: Project final paper, presentation, analyze implementation logic, data evaluation and provide assistance in project as needed

References

Boag, W.; Hsu, T. M.; McDermott, M.; Berner, G.; Alsentzer, E.; and Szolovits, P. 2020. Baselines for Chest X-Ray Report Generation. In *Proceedings of Machine Learning Research*, volume 116, 126–140. ML4H at NeurIPS 2019.

Johnson, A. E. W.; Pollard, T. J.; Berkowitz, S.; Greenbaum, N. R.; Lungren, M. P.; Deng, C.-y.; Mark, R. G.; and Horng, S. 2019. MIMIC-CXR: A large publicly available database of labeled chest radiographs. *arXiv:1901.07042*.

Rajpurkar, P.; Irvin, J.; Zhu, K.; Yang, B.; Mehta, H.; Duan, T.; Ding, D.; Bagul, A.; Langlotz, C.; Shpanskaya, K.; Lungren, M. P.; and Ng, A. Y. 2017. CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning. *arXiv:1711.05225*.

Appendix: Tables and Figures

Model (Variant)	BLEU-1	BLEU-2	BLEU-3	BLEU-4	CIDEr	Accuracy	Precision	F1
Random	0.253	0.128	0.074	0.045	0.047	0.536	0.622	0.632
1-gram	0.202	0.033	0.004	–	0.013	0.563	0.620	0.684
2-gram	0.210	0.109	0.056	0.030	0.019	0.565	0.631	0.672
3-gram	0.220	0.114	0.066	0.041	0.030	0.568	**0.653**	0.651
1-NN	0.260	0.138	0.083	0.053	0.059	**0.584**	0.642	**0.689**
CNN-RNN-BERT (epoch8)	**0.264**	**0.167**	**0.116**	**0.085**	0.214	0.380	0.000	0.000
CNN-RNN-BERT (epoch16–64 avg)	0.242	0.150	0.102	0.075	**0.216**	0.380	0.000	0.000

Table 2: Aggregated performance metrics across all baseline models and CNN-RNN-BERT extension.

Label	Random	3-Gram	1-NN	CNN-RNN-BERT (e16)	CNN-RNN-BERT (e40)
No Finding	0.568	0.286	0.455	0.413	0.413
Cardiomegaly	0.113	0.390	**0.445**	¡0.001	¡0.001
Atelectasis	0.241	0.271	**0.375**	¡0.001	¡0.001
Pleural Effusion	0.222	0.364	**0.532**	¡0.001	¡0.001
Edema	0.111	0.192	**0.286**	¡0.001	¡0.001
Pneumonia	0.063	0.036	**0.080**	¡0.001	¡0.001
Fracture	0.041	0.022	**0.060**	¡0.001	¡0.001
Lung Lesion	0.277	0.062	0.062	¡0.001	¡0.001
Consolidation	0.043	0.037	**0.085**	¡0.001	¡0.001
Pleural Other	0.020	¡0.001	**0.039**	¡0.001	¡0.001
Enlarged Cardiomediastinum	0.234	0.135	**0.142**	¡0.001	¡0.001
Pneumothorax	0.043	0.082	**0.111**	¡0.001	¡0.001
Support Devices	0.316	0.388	0.527	¡0.001	¡0.001
Micro-Average	0.215	0.294	**0.397**	0.209	0.209
Macro-Average	0.148	0.185	**0.258**	0.030	0.030

Table 3: Per-label CheXpert F1 performance across selected models. CNN-RNN-BERT models underperform clinically.