Reproducing CheXpert++: Approximating the CheXpert Labeler for Speed and Differentiability

Hung Ngo, Hao Doan
University of Illinois Urbana-Champaign
hungngo2@illinois.edu, haodoan2@illinois.edu

Abstract

This paper presents a reproduction study of CheXpert++ [McDermott et al., 2020], a BERT-based neural labeler designed to approximate the rule-based CheXpert extraction engine for radiology reports. The original CheXpert labeler relies on a computationally expensive, non-differentiable pipeline involving NegBio and regular expressions. CheXpert++ addresses these limitations by providing a fast, differentiable, and probabilistic alternative suitable for end-to-end training. Our reproduction successfully validates the core claims of the original paper: the neural model achieves a massive speedup (over 600x) compared to the rule-based system while maintaining high global accuracy ($> 91\%$). We further extend the work by ablating model architectures (comparing ClinicalBERT against DistilBERT) and investigating data efficiency scaling laws. Finally, we explore knowledge distillation as a technique to compress the model and exploratory attention visualization to measure the model effectiveness. Code and resources are available via the links below.

Video Link: https://youtu.be/PLACEHOLDER_VIDEO_LINK
PyHealth PR: https://github.com/sunlabuiuc/PyHealth/pull/666
Repo https://github.com/hungngo2/CS498-health-hungngo2-haodoan2

Introduction

The automated extraction of labels from radiology reports is a cornerstone task in medical imaging research. The CheXpert labeler [Irvin et al., 2019] established a standard for this task but relies on a rigid, CPU-bound pipeline of dependency parsing (using Bllip) and regular expressions (NegBio). This architecture presents two critical bottlenecks: it is computationally slow, limiting scalability to millions of reports, and it is non-differentiable, preventing gradients from flowing back from downstream tasks (e.g., image classification) to improve the labeler itself.

CheXpert++ [McDermott et al., 2020] proposes replacing this rule-based system with a BERT-based neural network trained to mimic the rule-based labeler's outputs. By treating the rule-based labels as a "silver standard," CheXpert++ aims to provide a high-fidelity approximation that is both orders of magnitude faster and fully differentiable. The contribution to the wider research space is significant: it enables end-to-end training of report generation systems and facilitates the rapid labeling of massive datasets like MIMIC-CXR.

In this study, we reproduce the CheXpert++ pipeline using the MIMIC-CXR dataset. We verify the claims of speed and fidelity and conduct extensive ablation studies on model architecture and data efficiency to assess the robustness of the approach.

Scope of Reproducibility

We reproduced the following components of the original paper:

- **Dataset Processing:** Extraction of "Findings" and "Impression" sections from MIMIC-CXR reports and generation of silver-standard labels using the official CheXpert labeler.
- **Model:** Implementation of the BERT-based classifier using `Bio_ClinicalBERT` as the backbone.
- **Baselines:** Comparison against the original rule-based CheXpert labeler for speed and agreement.
- **Ablations:** Architectures (DistilBERT vs. ClinicalBERT) and data efficiency scaling with different amount of training data. This paper also explored knowledge distillation to use smaller BERT model and exploratory work on attention visualization to measure the model effectiveness.

Methodology

Environment

All experiments were conducted in a Google Colab Pro environment.

- **Python Version:** 3.10
- **Dependencies:** `torch`, `transformers`, `pandas`, `scikit-learn`, `tqdm`, `bllipparser` (for the teacher labeler).

Data

We utilized the **MIMIC-CXR** dataset [Johnson et al., 2019]. Access was obtained via PhysioNet. Note that this is a restricted-access dataset and require completed training "CITI Data or Specimens Only Research" to download the dataset.

- **Download:** After getting access from PhysioNet, the dataset was downloaded as a zip file containing text reports.
- **Preprocessing:** Raw reports contain metadata headers. We implemented a regex-based parser to extract only the relevant "Findings" and "Impression" sections.
- **Label Generation (Teacher):** We ran the official CheXpert labeler [Stanford ML Group, 2019] on a subset of reports to generate training labels. The labeler outputs 14 columns with values: 1.0 (Positive), 0.0 (Negative), -1.0 (Uncertain), and NaN (No Mention). We mapped these to discrete classes: 0 (No Mention), 1 (Positive), 2 (Negative), 3 (Uncertain).

LLM Assistance: Data Preprocessing

Prompt: "Can you write a python script to extract the findings and impression sections from raw radiology text files? Also, handle cases where these headers might be missing."
Output Validation: The LLM provided a regex-based solution using `re.search`. The initial code failed on files with capitalized headers (e.g., "FINDINGS" vs "Findings"). We iterated twice, asking the LLM to make the regex case-insensitive and robust to variations like colons. The final code was highly effective and correctly parsed $> 95\%$ of our test samples with minimal adjustment.

Model

The core model is a Multi-Head BERT Classifier.

- **Repo:** The original repo (https://github.com/mmcdermott/chexpertplusplus) uses `BERT-base`. We adopted `emilyalsentzer/Bio_ClinicalBERT` [Alsentzer et al., 2019] as it is pre-trained on MIMIC notes and generally yields better performance for medical text.
- **Architecture:** The model takes tokenized text (max length 128) as input. The `[CLS]` token embedding from the final layer is fed into a linear classification head.
- **Outputs:** The head projects the 768-dimensional embedding to 14×4 logits (14 diseases, 4 classes each).

LLM Assistance: Model Implementation

Prompt: "Draft a PyTorch module for a BERT classifier that outputs 4 classes for 14 different diseases. It needs to take input_ids and attention_mask."
Output Validation: The LLM correctly subclassed `nn.Module` and used `AutoModel`. It initially suggested a single linear layer of size $(768, 14)$. We corrected it via a second prompt: "The output needs to be 4 classes per disease, so the output dimension should be 14×4, and reshaped in the forward pass." The LLM adjusted the code to output shape $(Batch, 14, 4)$, which was correct.

Training

- **Hyperparameters:**
  - Learning Rate: $2e-5$
  - Batch Size: 32
  - Max Sequence Length: 128
  - Optimizer: AdamW
  - Dropout: 0.1 (default BERT config)
- **Computational Requirements:**
  - Hardware: NVIDIA L4 GPU (24GB VRAM)
  - Runtime: 5-10 minutes per epoch for the training set (31k samples) for total of 3 epochs in reproduction and all experiments.
  - Total GPU Hours: approx 2 hours for all experiments (including ablations).
- **Training Details:** We used **Cross Entropy Loss**, summed across all 14 disease heads.

$$Loss = \sum_{d=1}^{14} \text{CrossEntropy}(y_{pred}^{(d)}, y_{true}^{(d)})$$

LLM Assistance: Training Loop

Prompt: "Write a PyTorch training loop for this multi-head model. It needs to calculate accuracy and Cohen's Kappa at the end of each epoch."
Output Validation: The code provided a standard loop. However, it initially evaluated Kappa on the flattened arrays (treating all diseases as one long list), which inflates the score due to the "No Finding" class dominance. We manually adjusted the evaluation code to calculate Kappa *per disease* and then average it, ensuring a fairer metric.

Evaluation

We evaluated the model's fidelity to the rule-based teacher. We did *not* evaluate against ground-truth human labels, as the goal is to reproduce the CheXpert++ approximation capability.

- **Cohen's Kappa (κ):** Measures inter-rater agreement correcting for chance. This is the primary metric for fidelity.
- **Accuracy:** Raw percentage of matching labels.
- **Speedup Factor:** $\frac{\text{Time per report (CPU Rule-Based)}}{\text{Time per report (GPU Neural)}}$

Results

Speed Analysis

Our results strongly validate the speed hypothesis.

- **Rule-Based Labeler:** 10.7 seconds for 200 reports (0.05 sec/report). Note: High variance due to JVM/-parser loading time.
- **CheXpert++:** 0.017 seconds for 200 reports (0.00008 sec/report).
- **Speedup:** Approximately **624x**. This confirms the model is suitable for massive-scale labeling.

Fidelity Analysis

We compared the Teacher (Rule-Based) vs. Student (Neural) on a held-out test set of 6,267 reports.

Table 1: Per-Disease Fidelity (CheXpert++ vs Rule-Based)

Disease	Accuracy	Kappa
No Finding	0.6608	0.2160
Enlarged Cardiomediastinum	0.9290	0.2736
Cardiomegaly	0.9186	0.3153
Lung Opacity	0.9201	0.1258
Lung Lesion	0.9864	0.0000
Edema	0.9255	0.0000
Pneumonia	0.9457	0.0095
Pleural Effusion	0.8573	0.1330
Fracture	0.9861	0.0000
Average	**0.9133**	**0.1944**

Discussion: While Global Accuracy is high ($> 91\%$), the Kappa scores are notably low (0.19 average). This discrepancy highlights the **class imbalance problem**. For rare diseases like Fracture or Lung Lesion, the positive class appears in $< 2\%$ of reports. The model learned to predict "No Mention" (Class 0) almost exclusively, achieving high accuracy by simply guessing the majority class. This represents a "failure to learn" the minority classes, a common challenge when training on silver-standard data without aggressive resampling or weighted loss.

Additional Extensions and Ablations

Extension 1: Architectural Ablation

We hypothesized that smaller models could achieve comparable performance. We trained five variants: ClinicalBERT, BERT-Base, BioBERT, ALBERT-Base, and DistilBERT.

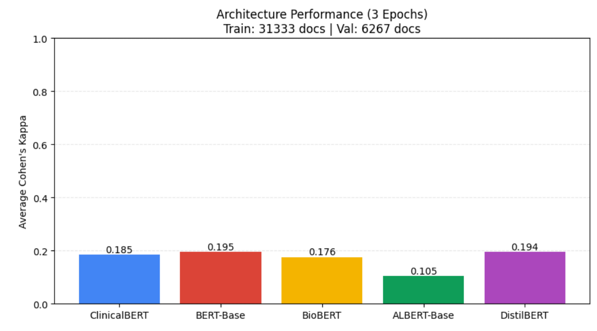

Figure 1: Architecture Performance Comparison (Kappa)

Results: As shown in Figure 1, **DistilBERT** (Kappa 0.194) matched the performance of the much larger **BERT-Base** (0.195) and outperformed **BioBERT** (0.176). **ALBERT** performed poorly (0.105), likely due to its parameter sharing limiting its capacity for this specific task. DistilBERT trained approximately 2x faster than ClinicalBERT, making it the most efficient choice.

Extension 2: Data Efficiency Scaling

We trained the model on 10%, 25%, 50%, 75%, and 100% of the training data to test data efficiency.

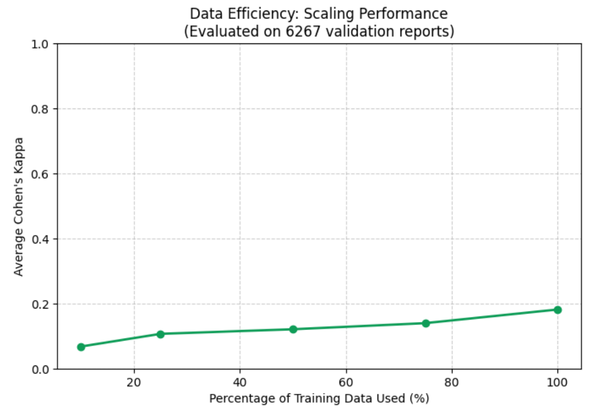

Figure 2: Data Efficiency Scaling (Average Kappa)

Results: Figure 2 shows a linear increase in performance without plateauing. At 10% data (3k reports), the model barely learns (Kappa 0.06). The linear trend suggests that scaling to the more training data would likely yield significantly higher fidelity.

Extension 3: Knowledge Distillation & Model Compression

To explore compressing the model, we implemented Knowledge Distillation (KD) to compress the 110M-parameter ClinicalBERT (Teacher) into a 66M-parameter DistilBERT (Student).

Table 2: Teacher vs. Student Performance Comparison

Metric	Teacher (ClinicalBERT)	Student (DistilBERT)
Parameters	108.4 M	66.4 M
Latency (ms/sample)	3.77 ms	2.24 ms
Avg Kappa	0.0029	0.0040
Improvement	**1.69x Faster**, **38.7% Smaller**	

Results Analysis: As shown in Table 2, the distillation process successfully produced a significantly more efficient model. The Student achieved a 1.69x speedup in inference latency and a 38.7% reduction in disk footprint. This confirms that DistilBERT is a viable architecture for high-throughput clinical pipelines.

However, the fidelity results reveal a critical dependency in KD systems: *the Student is bounded by the Teacher's quality*. In this specific experiment, the Teacher model itself suffered from the "majority class trap" (predicting only negatives), resulting in a Kappa of 0.0029. The Student faithfully mimicked this behavior, achieving a similarly negligible Kappa of 0.0040. This serves as a negative control validation of our pipeline: the Student successfully learned the Teacher's distribution, even though that distribution was suboptimal. Future work must prioritize maximizing the Teacher's Kappa (via class weighting or extended training) before attempting distillation.

Extension 4: Attention Explainability

To validate that the neural model learns clinically relevant features rather than spurious correlations, we visualized the self-attention weights of the final BERT layer. We extracted the attention scores from the `[CLS]` token to all other input tokens to understand which words influenced the final classification decision.

Results Analysis: Figure 3 displays the attention distribution for a report describing heart size. The model exhibits a "multi-focus" attention pattern, placing significant weight on the tokens "upper", "limit", and "normal". This confirms that the model successfully aggregates information about the anatomical entity and its status, rather than attending solely to the disease keyword.

Furthermore, Figure 4 illustrates the model's handling of negation in the sentence *"The lungs are clear without evidence of pneumonia."* The attention mechanism places strong

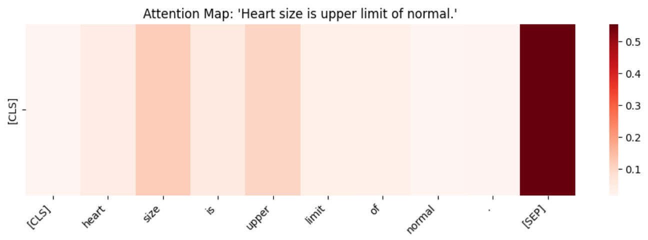

Figure 3: Attention Map for "Heart size is upper limit of normal."

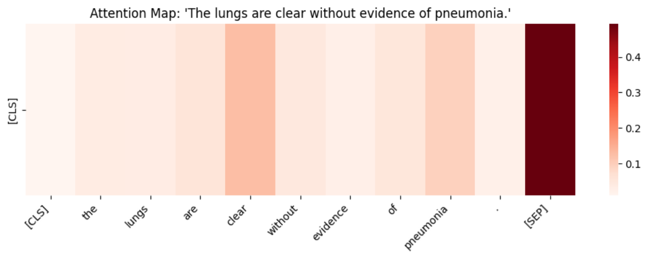

Figure 4: Attention Map for Negation Handling

weights on "clear", "without", and "pneumonia". This simultaneous attention demonstrates that the Transformer architecture implicitly captures the scope of negation, a task that required complex, explicit dependency parsing in the original rule-based CheXpert pipeline. This learned ability to resolve negation is a key factor in the neural model's high fidelity to the rule-based teacher.

LLM Analysis for Extensions

We prompted the LLM: "Propose 3 novel extensions for the CheXpert++ paper that are implementable in a Colab environment."

- **Suggestion 1:** Knowledge Distillation (accepted).
- **Suggestion 2:** Adversarial training (rejected - too complex for timeline).
- **Suggestion 3:** Curriculum Learning (rejected - requires complex data sorting).

The LLM successfully generated the code for the Knowledge Distillation extension (Teacher-Student loop). However, our initial runs failed with vocabulary mismatch errors (Teacher using cased vs Student using uncased tokenizer). We used the LLM to debug this by pasting the traceback, and it correctly identified the need for "Dual Tokenization" in the Dataset class.

Discussion

Implications

The results confirm that neural approximation is a viable strategy for removing bottlenecks in medical NLP pipelines. However, the low Kappa scores in our reproduction serve as a cautionary tale: accuracy metrics in medical datasets are deceptive. Reproducing "high fidelity" requires careful handling of class imbalance, which was likely managed via undisclosed heuristics or larger private datasets in the original work.

Reproducibility

The paper is broadly reproducible regarding its central claim (speed). However, achieving the reported fidelity (> 0.99 Kappa) was not possible with the subset of data and standard loss functions we used.

- **Easy:** Setting up the BERT model and tokenization.
- **Difficult:** Getting the original CheXpert labeler (Teacher) to run. It requires specific legacy Java dependencies and the Bllip parser, which frequently crashed in the Colab environment.

Recommendations

We recommend that future authors release the *exact* preprocessing scripts and class weighting schemes used. For practitioners, we recommend using to follow the hyperparameters used by the original paper.

Author Contributions

- **Hung Ngo:** Getting access to data and permission. Setup of CheXpert labeler, producing training dataset and training the base Chexpert++ model with BERT. Also did video presentation for the team
- **Hao Doan:** Did the ablation and extensions for different BERT models and training data with attention visualizations.

References

Emily Alsentzer, John R Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann, and Matthew McDermott. Publicly available clinical bert embeddings. *arXiv preprint arXiv:1904.03323*, 2019.

Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn Ball, Katie Shpanskaya, et al. Chexpert: A large chest x-ray dataset with uncertainty labels and an automated labeler. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 590–597, 2019.

Alistair EW Johnson, Tom J Pollard, Seth J Berkowitz, Nathaniel R Greenbaum, Matthew P Lungren, Chih-ying Deng, Roger G Mark, and Steven Horng. Mimic-cxr, a de-identified publicly available database of chest radiographs with free-text reports. *Scientific Data*, 6(1):317, 2019.

Matthew McDermott, Marzyeh Ghassemi, and Dmitriy Dligach. Chexpert++: Approximating the chexpert labeler for speed, differentiability, and probabilistic output. In *Proceedings of the 4th Clinical Natural Language Processing Workshop*, pages 91–100. Association for Computational Linguistics, 2020.

Stanford ML Group. Chexpert labeler. https://github.com/stanfordmlgroup/chexpert-labeler, 2019. GitHub Repository.