Relabeling Chest X-Rays

A look at consistency between Stanford and NIH Chest X-Ray Labels

Introduction

- Initial Proposal
    - CNN trained to classify 2 GBs Sample of Chest X-Rays from Kaggle
    - National Institute of Health (NIH)

- Literature Survey:
    - Discovery of Luke Oakden-Rayner’s paper regarding incorrect labeling in NIH dataset
    - CheXpert dataset by Stanford University, with verified labelling by radiologists.

Radiologist’s Concerns

- NLP modeling to Label NIH Dataset
  - Concern: High number of images did not match with their labels
- Visually inspected a sample of ~130 images from 6 different classes
- Radiology reports are not meant to give full picture
  - Purpose is to give specific information to doctor, often only answering a specific question
  - Based on clinical context and patient history, not just image
- Suspicious of specific labels (e.g. drained pneumothorax)

Revised Proposal

- Train a model on CheXpert
- Run model on NIH data, and achieve a more accurate labelling of the images

Dataset Comparison

	CheXpert (Stanford)	NIH
Number of patients	65,240	30,805
Number of Images	224,316	112,120
Number of Labels	14	15

Labelling of Datasets

NIH:

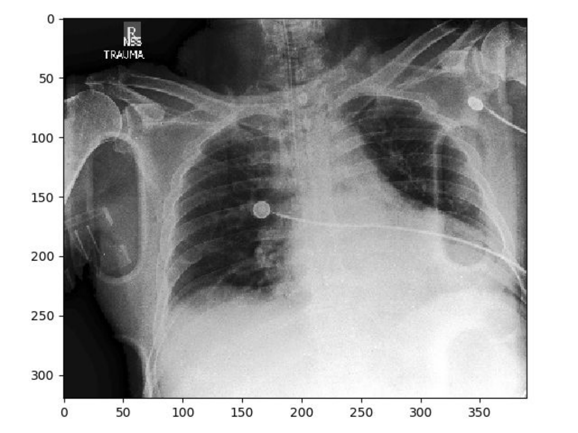
- NLP used to extract labels from reports
- Each condition is either Present (1) or Absent (0)

CheXpert:

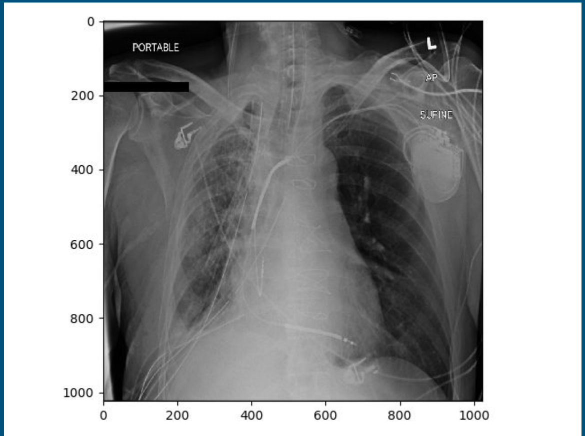
- Improved NLP algorithm
  - Validated by visual inspection by several radiologists.
- Each condition is Positive (1), Negative (0), Uncertain (u), or Blank (no mention in report)

Datasets - CheXpert and NIH

CheXpert - Chest X-ray - Frontal view



NIH - Chest X-Ray - Frontal View



Observations: Cardiomegaly, Lung Opacity, Atelectasis, Pneumothorax, Support Devices

Observations: Emphysema, Infiltration, Pleural_Thickening, Pneumothorax

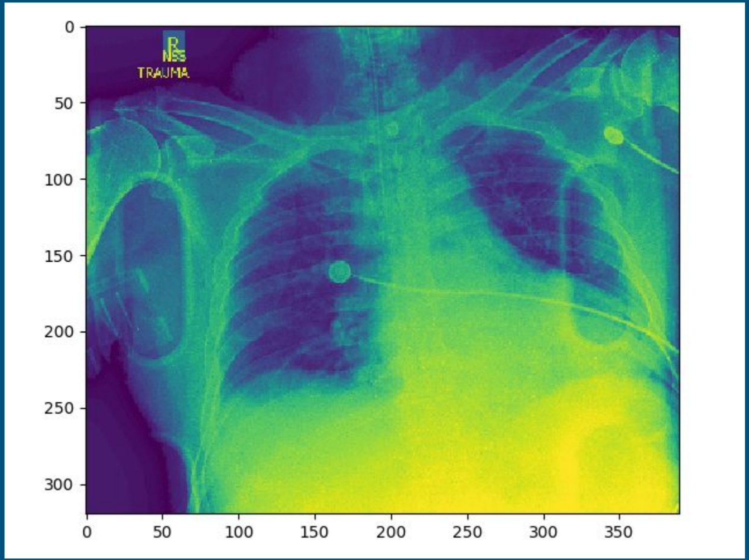
Labels in Common

CheXpert and NIH

- Pneumonia
- Edema
- Cardiomegaly
- Consolidation
- Pneumothorax
- Atelectasis
- No Finding

Image Preprocessing

- Core Preprocessing:
  - Equalize Histogram
  - Grayscale to RGB
  - Gaussian Blur
  - Fixed Ratio Resize: Produces long_side x long_side images filled with pixel_mean
- Data Augmentation (training only)
  - Random Transforms
    - Rotation
    - Translation
    - Shear



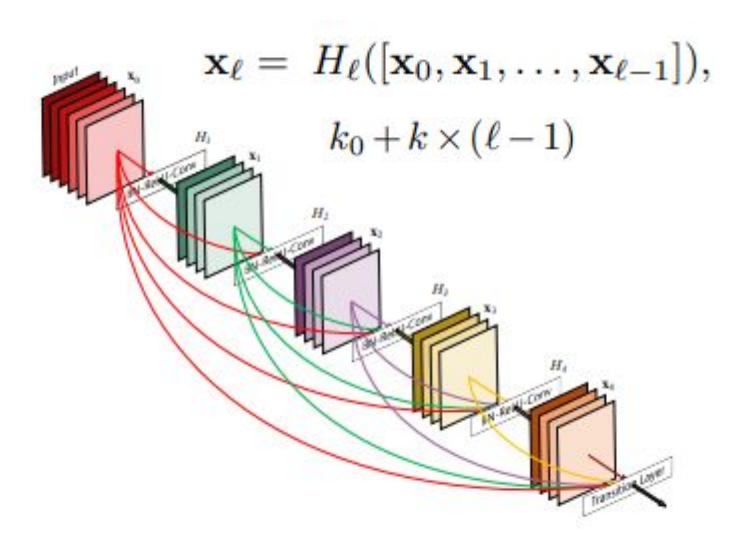
Grayscale to RGB Transformation

Label Preprocessing

- Changed: Uncertain (u), blank
- Strategies - U-Zeros, U-Ones

Label	Description	UZeros	UOnes
1	Positive	1	1
0	Negative	0	0
u	Uncertain	0	1
	No Label	0	0

Our Model - DenseNet-BC



A 5-layer dense block with a growth rate of k = 4. Each layer takes all preceding feature-maps as input.

- **DenseNet -** Dense Convolutional Network
- **DenseBlock -** Densely Connected Layers
- **Transition Layers**
  - Reduces inputs from DenseBlock to DenseBlock
- **Growth Rate (k)**
  - Regulates how much information is added to the network at each layer.
  - K0 is number of channels in input layer
  - L is the number of layers in the DenseBlock
- **Bottleneck -** Computational Improvement
  - Reduces inputs going into Conv Layers
  - Adds 1x1 Conv Layer
- **Compression -** Improves Model Compactness
  - Reduces Number of Output Feature Maps per DenseBlock

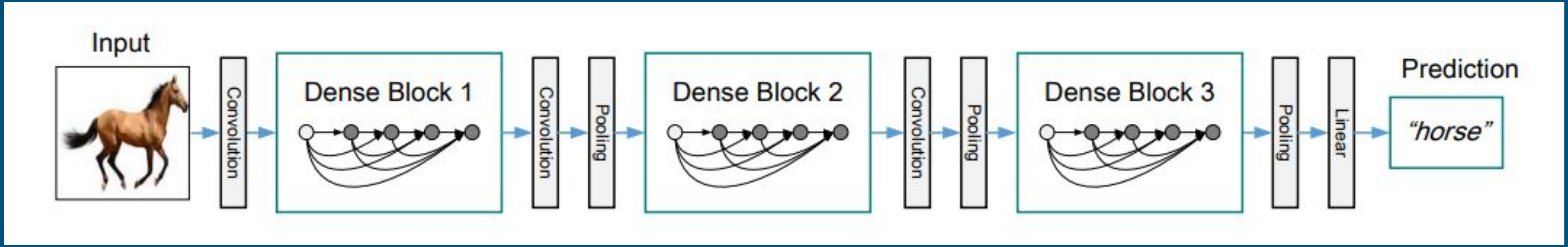
DenseNet 121

DenseNet Advantages:

- Alleviates Vanishing Gradient Problem
- Strengthen Feature Propagation
- Encourage Feature Reuse
- Reduce Number of Parameters

C -> P -> D -> T -> D -> T -> D -> T -> D -> Classification

- **DenseNet121 -** 121 Layer DenseNet
  - C - Convolutional Layer
    - BN -> ReLU -> Conv
  - P - Max Pooling
  - D - Dense Block
    - (1x1 C) -> (3x3 C)
    - Repeats
  - T - Transition Layer
    - (1x1 C) -> (Avg Pooling)



Model Training

- Model Training Phases
  - 15 Epochs per Phase
  - Saved a Model per Phase
- 3 Phases (45 Epochs) Total
  - ~ 21.5 hours of Training Time
- Optimizer = SGD
  - Initial LR of 0.0001
  - Momentum of 0.9
- Loss Criterion
  - BCEWithLogitsLoss
- Custom Pytorch DataLoaders
  - pin_memory = True
- Parellizing the data load (num_workers)

Models	**Training Epochs**	**Batch Size**	**Scheduler**	**RunTime**
v2	0-14	16	LRStep, 1/10 every 2 epochs	~ 6.5 hours
v3	15-29	8	LRStep, 1/10 every 2 epochs	~ 7.5 hours
v4	30-44	8	None (constant LR)	~ 7.5 hours

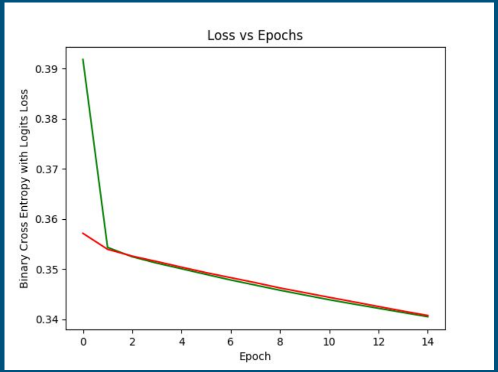
```
model = densenet121(num_classes=14).to(device)

optimizer = torch.optim.SGD(model.parameters(), lr=LR, momentum=MOMENT)

criterion = nn.BCEWithLogitsLoss()
```

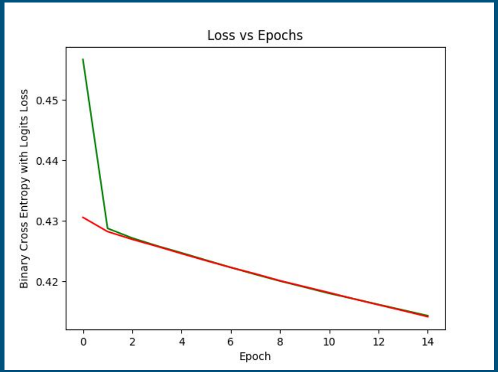
Loss Vs Epochs

UZeros Model

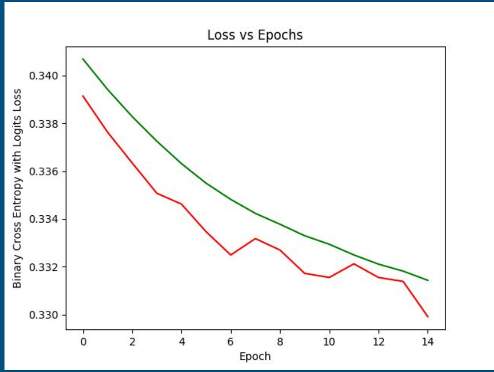
UOnes Model

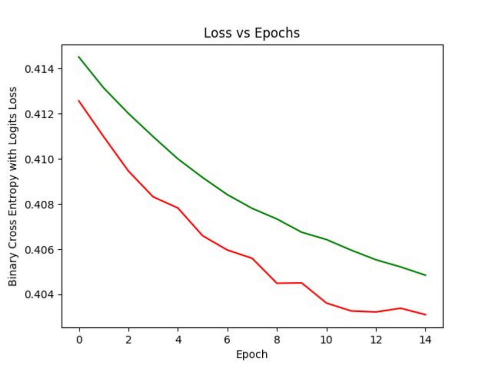
V2





V3





Model Evaluation

AUC, and Precision for CheXpert Dataset

Labels	AUC		Weighted Precision	
	Uzeros	Uones	Uzeros	Uones
Pneumonia	0.552544	0.759956	0.945906	0.786100
Edema	0.758730	0.761552	0.730872	0.715448
Cardiomegaly	0.719437	0.714918	0.774423	0.712859
Consolidation	0.835519	0.861149	0.873153	0.655775
Pneumothorax	0.722345	0.709071	0.832754	0.809125
Atelectasis	0.754545	0.779870	0.723606	0.488606
No Finding	0.853786	0.863856	0.861171	0.863509

Model Evaluation Continued

AUC, and Precision for the NIH Dataset

Labels	AUC		Weighted Precision	
	U-Zeros	U-Ones	U-Zeros	U-Ones
Pneumonia	0.536398	0.686517	0.978003	0.978003
Edema	0.801280	0.811856	0.960727	0.960901
Cardiomegaly	0.630075	0.634492	0.950329	0.950329
Consolidation	0.743061	0.741096	0.920997	0.920997
Pneumothorax	0.661201	0.642607	0.905655	0.905655
Atelectasis	0.671364	0.679177	0.826977	0.826977
No Finding	0.673435	0.674797	0.636713	0.636242

Learning and Limitations

- Image transformations and Gaussian blur led to better learning by the model
- If the criteria is AUC, Uones model performs better than Uzeros
- If Precision is the criteria, Uzeros model has better scores than Uones
- Python Versions Matter (and related packages)
- Training time large due to small batch size; limitations on GPU

Conclusion

- Approach for improving the NIH Dataset’s labels seems valid
- Our high AUC values indicate that our models have some predictive power
- However
  - Our high precision values are driven by relatively high null accuracy rates for some labels
  - Our model mostly predicts zeros
- Suspicious of NIH Results
  - Models share relatively consistent Precision values on NIH dataset

Future Improvements

- Better training/testing subsets
- Ensemble of winning models (using Decision Tree, Logistic Regression)
- Add “Feature Pyramid Attention Network” to current model
- Higher capacity GPU(s) for larger batch sizes, and potential ensemble training.

Thank You