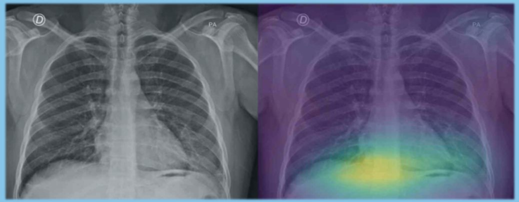
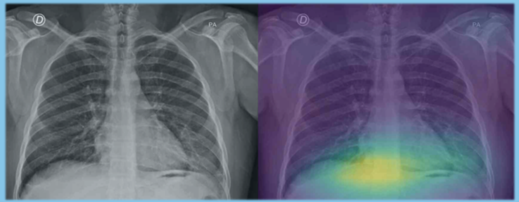


Deep Learning Presentation Slides

Chest X-Ray Classification Using Convolutional Neural Networks

George Pappy - 26 January 2022

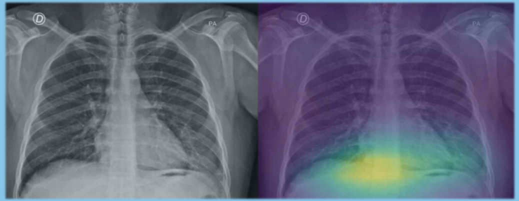
Introduction



<u>Tinytown, Pennsylvania</u>:

- Approximately 5000 residents
- Elderly-skewing Population
- Nearest hospital 100 miles away

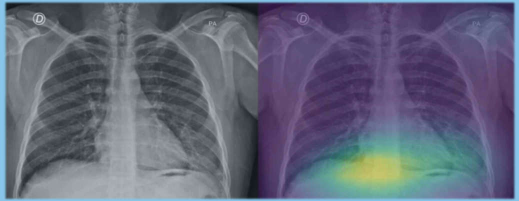
Introduction (con’t.)



Tinytown Health Clinic:

- Run by a single Doctor
- One Registered Nurse

Introduction (con’t.)

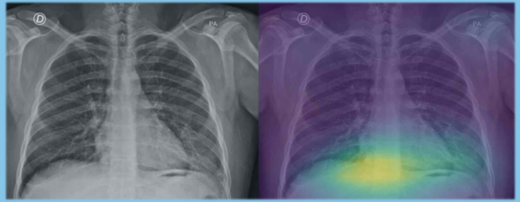


- Tinytown residents suffer inordinately from:
  - Atelectasis
  - Cardiomegaly
  - Edema
  - Pleural Effusion

- Tinytown Health can take chest X-rays:

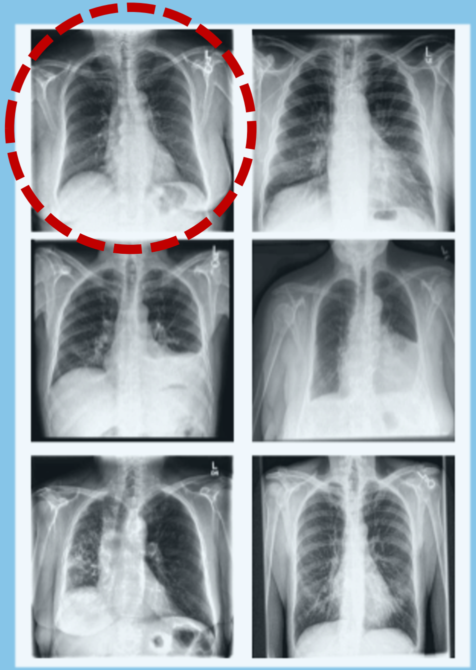
  → **But the staff lacks diagnostic expertise**

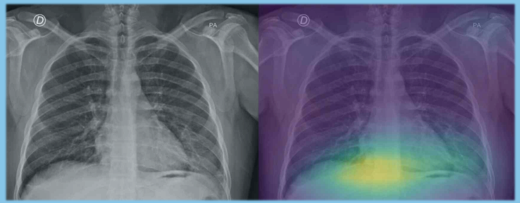
Introduction (con’t.)



Goal: Tinytown Health wants a chest X-ray classifier

- To identify “Potentially Positive” diagnoses
- Would prompt the staff to electronically transmit such X-rays to a Scranton radiologist





Methodology

Dataset: Compiled by Stanford University School of Medicine

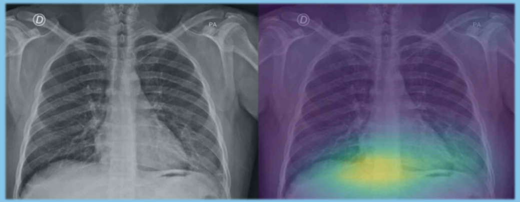
- 53,528 frontal chest X-rays
- Taken October 2002 - July 2017 at inpatient & outpatient centers
- Each X-ray labeled for all 4 diagnoses of interest

  → **4 non-mutually-exclusive targets:**

  - Atelectasis, Cardiomegaly, Edema, Pleural Effusion
  - Various degrees of imbalance (16% : 84% worst; 47% : 53% best)

- 70%/15%/15% Train/Validation/Test Set splits

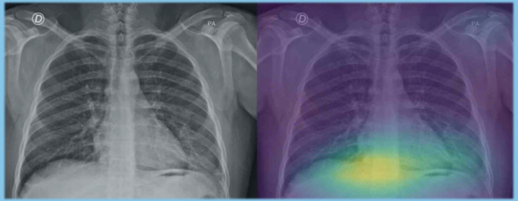
Results



Baseline (Traditional ML) Classifiers:

Model	Target(s)	AUROC (Test Set)
Random Forest (n_trees=500)	All 4	0.494, 0.500, 0.538, 0.636
Logistic Regression (l2=0.0005)	1st (Atelectasis)	0.557
Logistic Regression (l2=0.025)	2nd (Cardiomegaly)	0.576
Logistic Regression (l2=0.000075)	3rd (Edema)	0.573
Logistic Regression (l2=0.00005)	4th (Pleural Effusion)	0.604

Results (con't.)



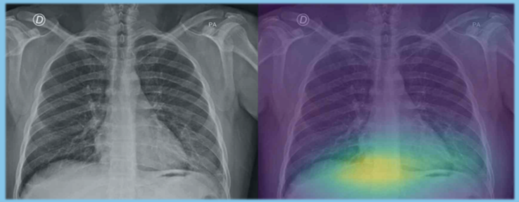
<u>**Multi-Label CNN Model**</u>:

- VGG-16 pre-trained on ImageNet dataset
- Final (top) layers replaced with several Dense layers
- Vast majority of total weights non-trainable

Layer (type)	Output Shape	Param #
input_9 (InputLayer)	[(None, 224, 273, 1)]	0
conv2d_29 (Conv2D)	(None, 224, 273, 3)	30
vgg16 (Functional)	(None, 7, 8, 512)	14714688
flatten_7 (Flatten)	(None, 28672)	0
dense_21 (Dense)	(None, 128)	3670144
dense_22 (Dense)	(None, 64)	8256
dense_23 (Dense)	(None, 4)	260

Total params: 18,393,378
Trainable params: 3,678,690
Non-trainable params: 14,714,688

Results (con't.)

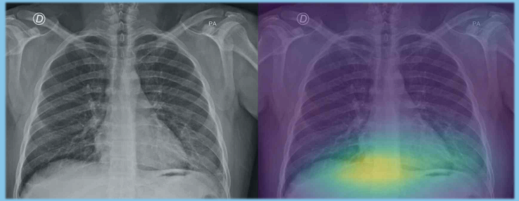


CNN Compared to Baseline Classifiers:

Model	Target(s)	AUROC (Test Set)
Random Forest (n_trees=500)	All 4	0.494, 0.500, 0.538, 0.636
Logistic Regression (l2=0.0005)	1st (Atelectasis)	0.557
Logistic Regression (l2=0.025)	2nd (Cardiomegaly)	0.576
Logistic Regression (l2=0.000075)	3rd (Edema)	0.573
Logistic Regression (l2=0.00005)	4th (Pleural Effusion)	0.604
CNN (no dropout or regularization)	All 4	0.652, 0.725, 0.725, 0.764

No common dropout/regularization parameters could be found to optimize all targets simultaneously

Results (con’t.)



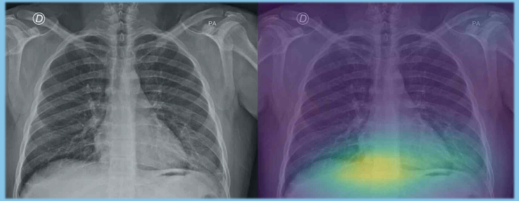
Single-Target CNN Models:

- VGG-16 pre-trained on ImageNet dataset
- Final (5th) block of VGG-16 Convolutional layers not frozen
- Now only ≈41.5% of total weights non-trainable
- Dropout, l1 and l2 regularization

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 273, 1)]	0
conv2d (Conv2D)	(None, 224, 273, 3)	30
vgg16 (Functional)	(None, 7, 8, 512)	14714688
flatten (Flatten)	(None, 28672)	0
dense (Dense)	(None, 128)	3670144
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8256
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 1)	65

Total params: 18,393,183
$\boxed{\text{Trainable params: 10,757,919}}$
Non-trainable params: 7,635,264

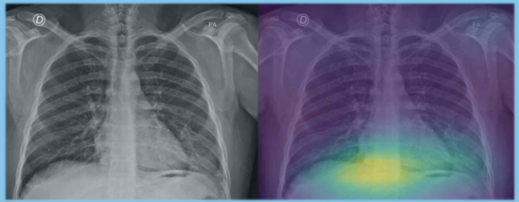
Results (con’t.)



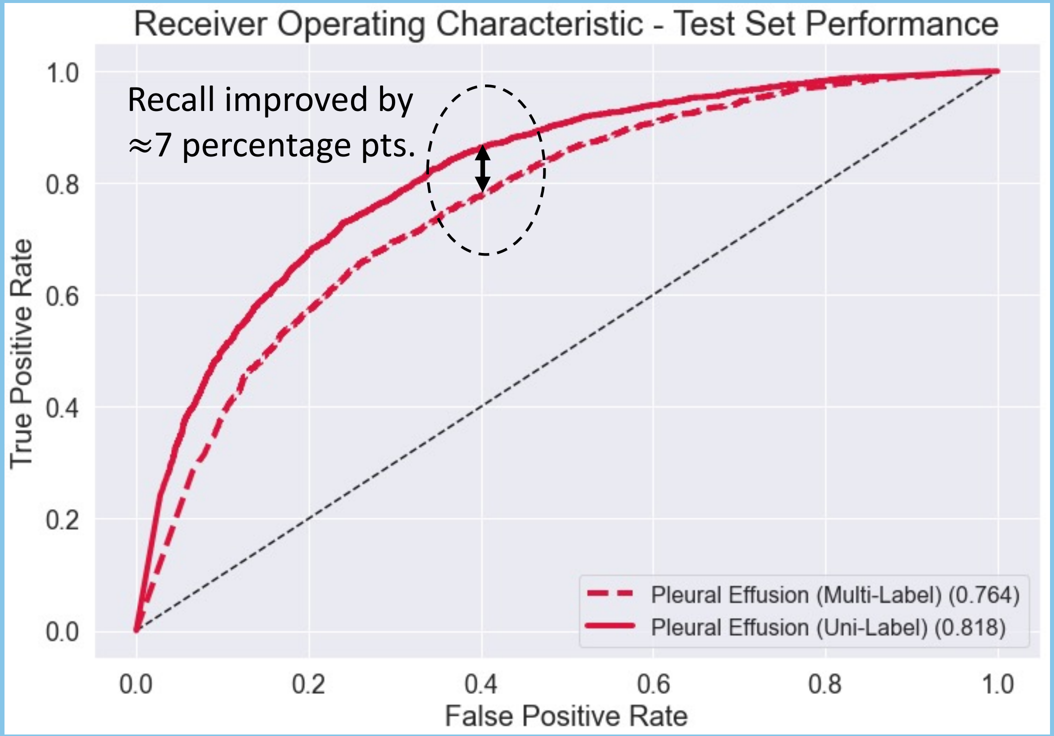
Multi-Label CNN Compared to Single-Target CNNs:

Model	Target(s)	AUROC (Test Set)
CNN (no dropout or regularization)	All 4	0.652, 0.725, 0.725, 0.764
CNN (dropout=0.30, l1=0.00001, l2=0.0001)	1st (Atelectasis)	0.688
CNN (dropout=0.35, l1=0.00001, l2=0.0001)	2nd (Cardiomegaly)	0.788
CNN (dropout=0.35, l1=0.00025, l2=0.0025)	3rd (Edema)	0.753
CNN (dropout=0.40, l1=0.00001, l2=0.0001)	4th (Pleural Effusion)	0.818

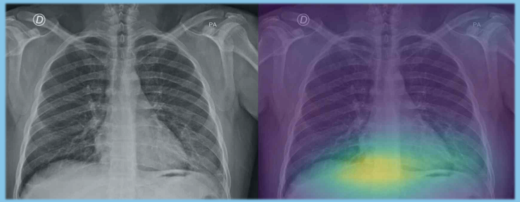
Results (con't.)



<u>Multi-Label CNN Compared to Single-Target CNNs</u>:



Conclusions

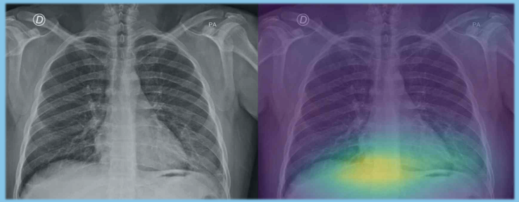


Recommendations

- Clinicians must establish acceptable False Positive Rates for all 4 conditions

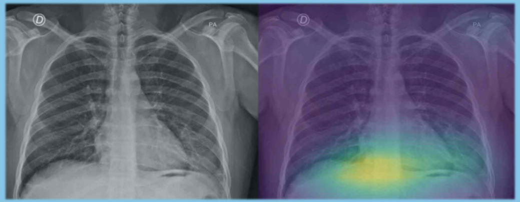
  → Sets thresholds for when to declare "Potentially Positive" for each target

- Deploy the overall model and track its performance metrics over time



Appendix

Future Work



1) Try training the models entirely from scratch (including all VGG-16 blocks)

2) Include additional diagnoses in the overall model

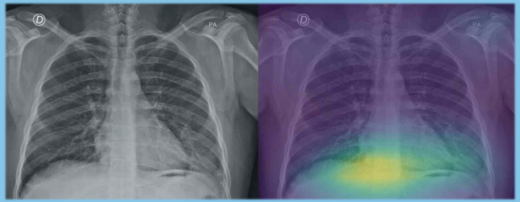
   - Dataset has labels for other conditions

   - Examples: Enlarged Cardiomediastinum, Lung Lesion, Consolidation, Pneumothorax

3) Consider using higher-resolution images (at cost of much longer training time)

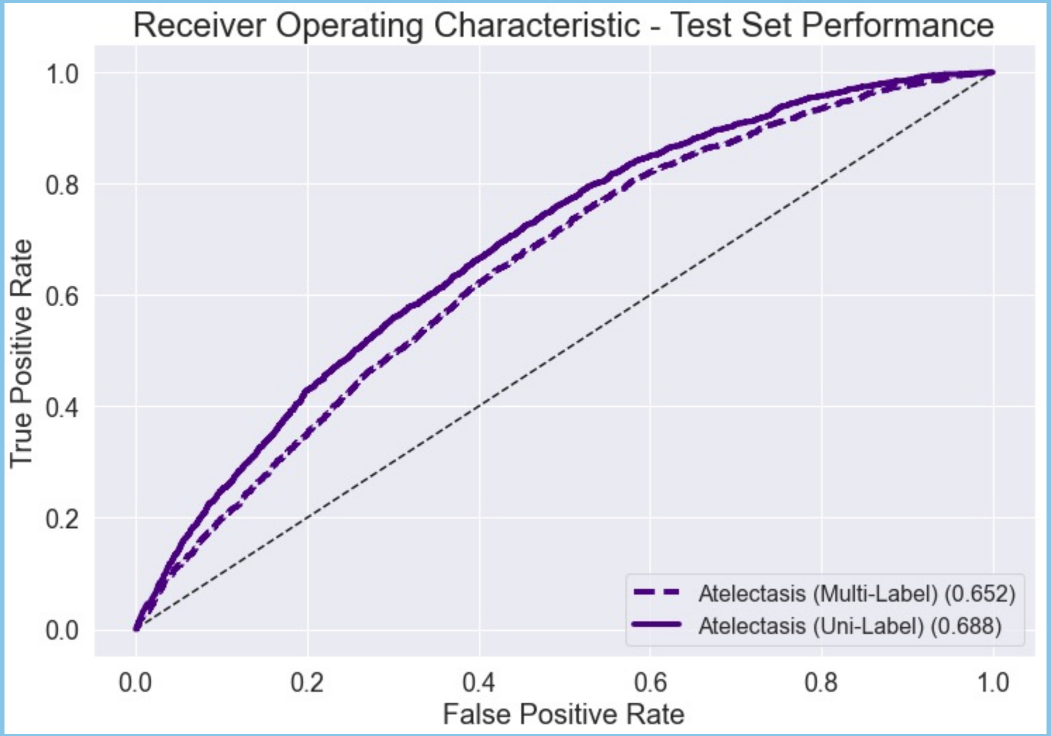
   - Images used for this project were low-resolution: 224 x 273 x 1

   - Many researchers have used 390 x 320 x 1 resolution (and 2457 x 2016 x 1 is available)

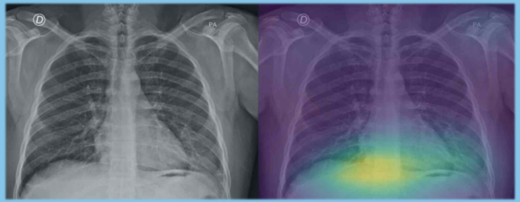


More Results

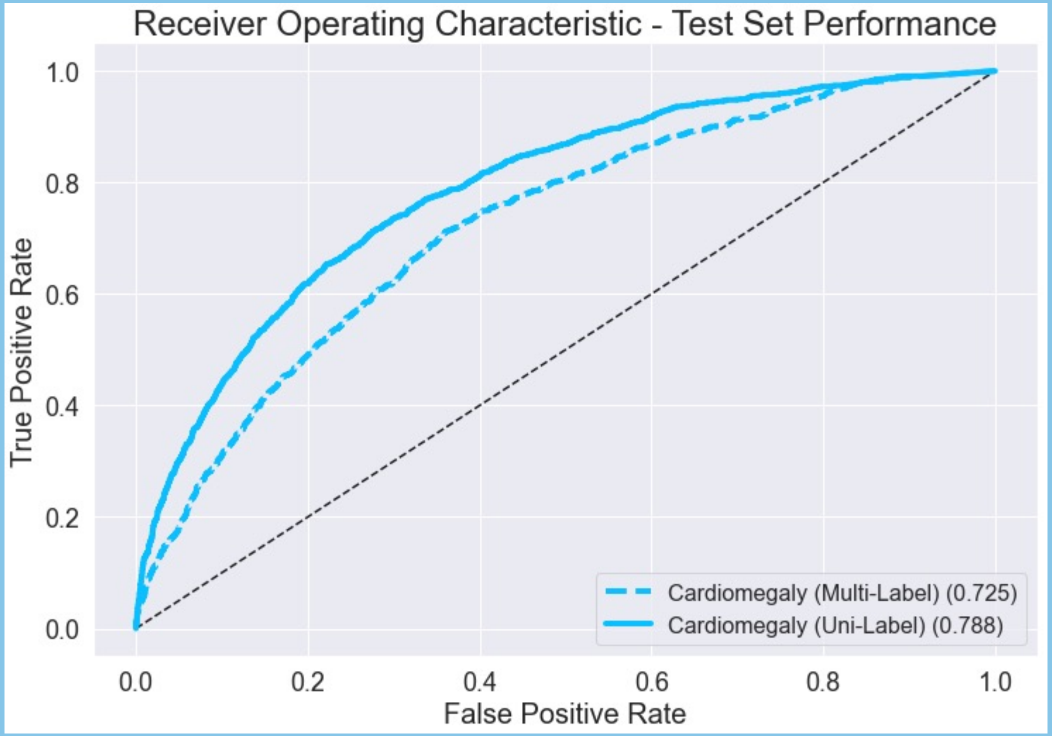
<u>Multi-Label CNN Compared to Single-Target CNNs</u>:



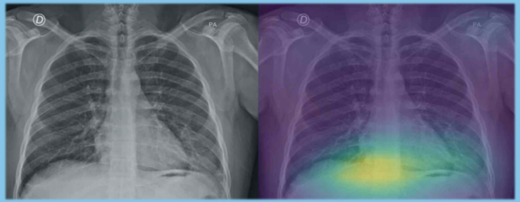
More Results (con't.)



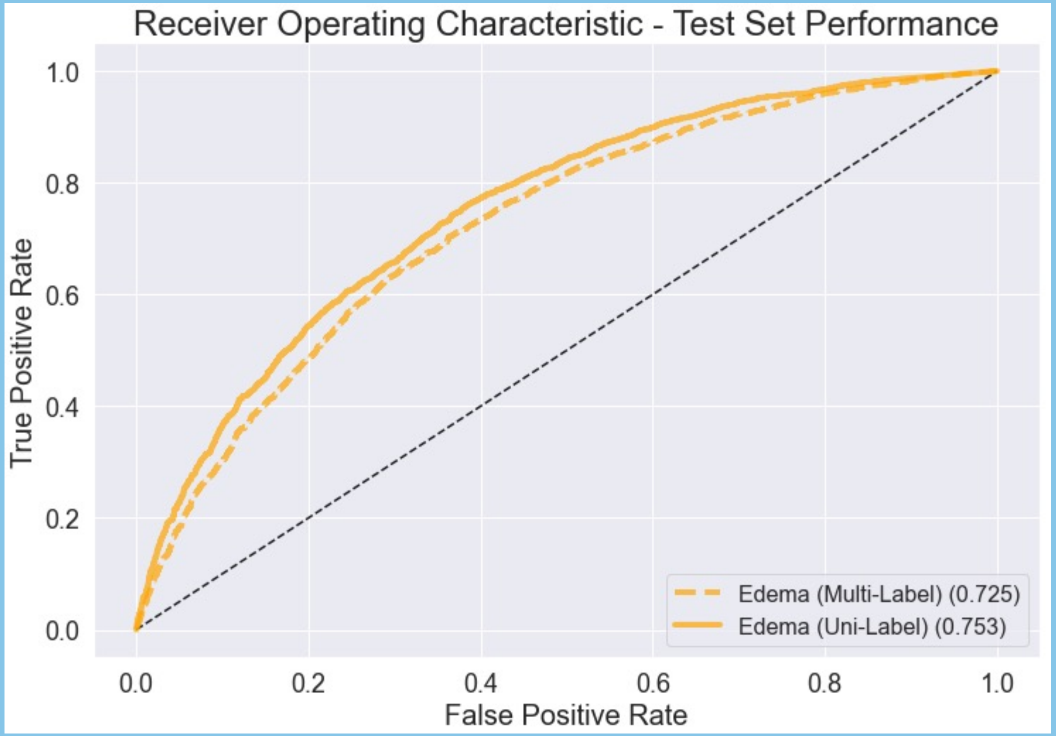
<u>Multi-Label CNN Compared to Single-Target CNNs</u>:



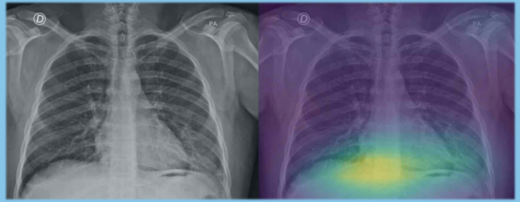
More Results (con’t.)



Multi-Label CNN Compared to Single-Target CNNs:

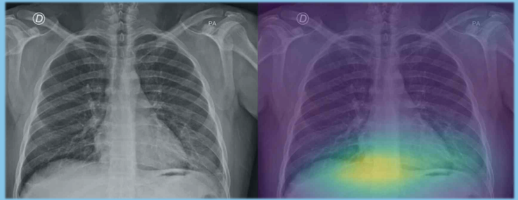


Methods & Tools



- **Pandas**: clean, explore, engineer features and generate final modeling data
- **scikit-learn**: build ML classification models as well as to perform data splitting for model development and measure model performance
- **Keras/TensorFlow 2:** build Deep learning classification models
- **Matplotlib/Seaborn**: visualizing data exploration, modeling and final results
- **Python 3.8**: to run all of the above

CNN Hyperparameter Tuning



- Full training dataset requires ≈2.5 hours per epoch

  → Too time-consuming to perform practical hyperparameter tuning

- Used “baby” training & validation datasets for tuning
  - 2.3% of full dataset
  - Used scikit-learn stratified split to maintain consistent relative class distributions
- Saved best models (w/ hyperparameters tuned using “baby” datasets)
  - Loaded these into final (full dataset) model training pipelines as starting points
  - Converged quickly to optimal, well-fit models (within 2-3 epochs)