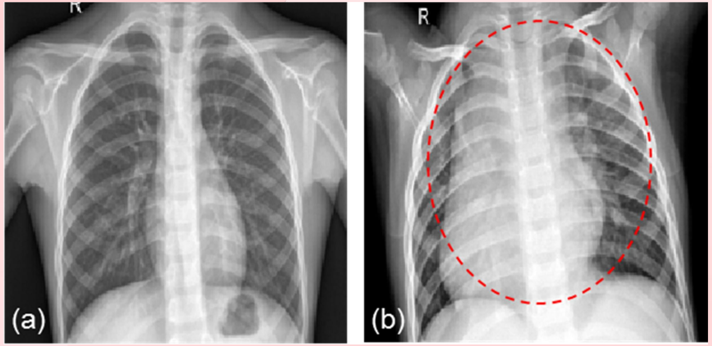
Stanford AIMI 2025 Summer Research Internship



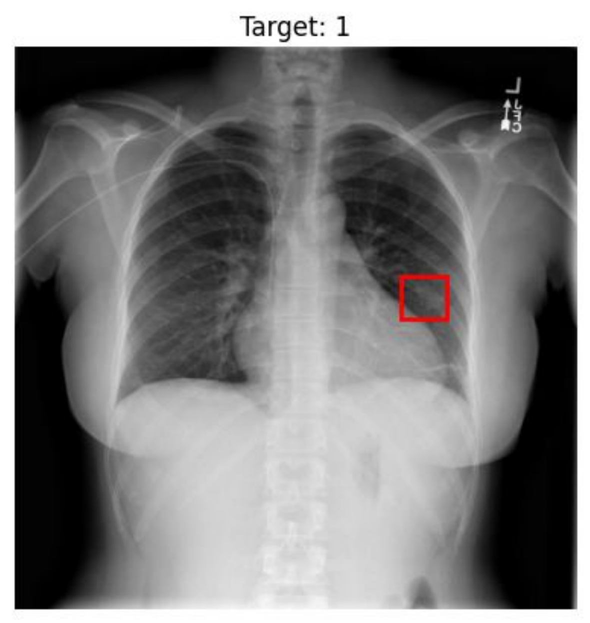
ResNet Rebels

Stanford
AIMI

Problem Statement

Objective: Develop a reliable AI model to detect pneumonia from chest X-rays

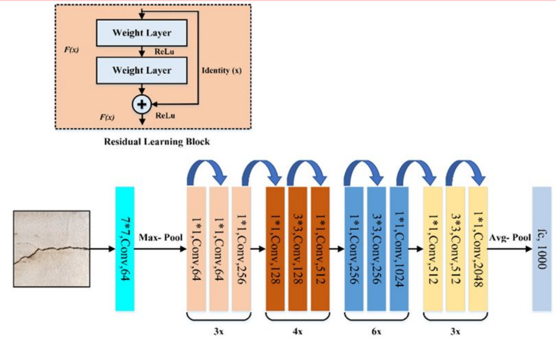
- Train classifier on chest X-rays datasets
- Improve performance using label extraction tools
- Extend to bounding box localization



01

Classification Task

ResNet-18



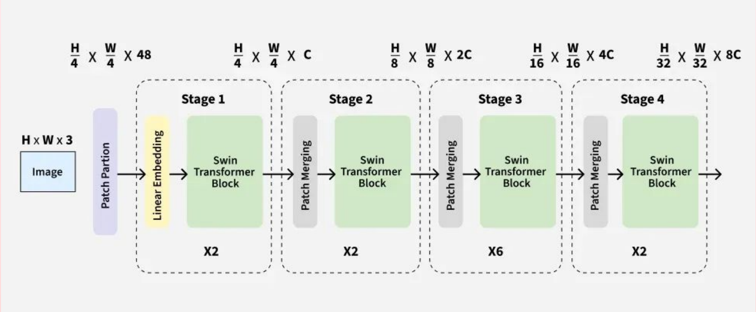
- Shallow variant trained on ImageNet-1K
- Convolutional layers for feature extraction
- Pooling layers for dimensionality reduction
- Fully connected layers for classification across all classes

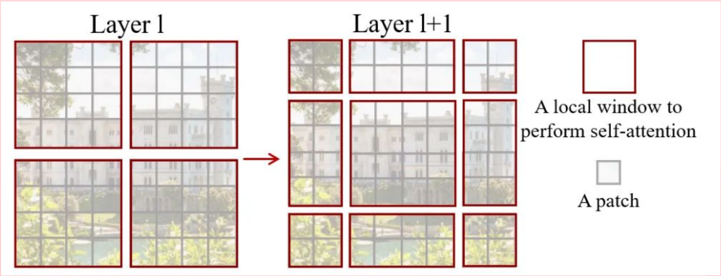
```
Epoch 1 | Loss: 0.4249 | Train Acc: 0.9517 | Train AUC: 0.9871 | Val Acc: 0.7962 | Val AUC: 0.8740
Epoch 2 | Loss: 0.1474 | Train Acc: 0.9965 | Train AUC: 1.0000 | Val Acc: 0.7642 | Val AUC: 0.8549
Epoch 3 | Loss: 0.0460 | Train Acc: 0.9983 | Train AUC: 1.0000 | Val Acc: 0.7832 | Val AUC: 0.8592
Epoch 4 | Loss: 0.0199 | Train Acc: 0.9996 | Train AUC: 1.0000 | Val Acc: 0.7729 | Val AUC: 0.8433
Epoch 5 | Loss: 0.0136 | Train Acc: 0.9987 | Train AUC: 1.0000 | Val Acc: 0.7953 | Val AUC: 0.8562
```

Evaluation metrics encouraged us to search for a better solution

Sliding Windows Transformer (Swin)

- Splits CXR images into fixed-size patches, reducing self-attention complexity from $O(N)$ to $O(N^2)$
- Allows cross-patch interaction across layers
- Four-stage architecture (96 → 192 → 384 → 768 channels) mimics CNN pyramids for downstream tasks
- Captures long-range context well, but demands more GPU memory

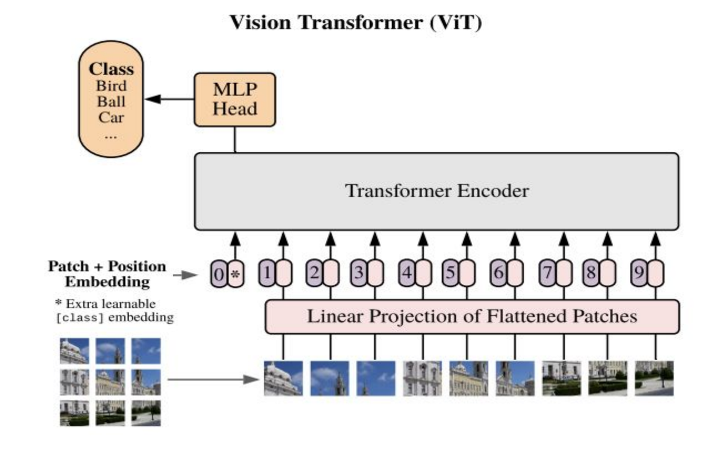


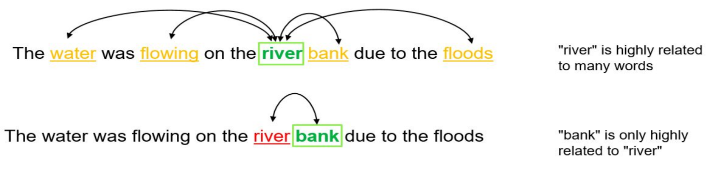


https://arxiv.org/pdf/2103.14030
https://www.geeksforgeeks.org/computer-vision/swin-transformer

Vision Transformer (ViT Base-16)

- Image → flattened 16×16 patches → linear layer → 196 patch tokens
- Add fixed position info + prepend [CLS] token → 197-token sequence
- **Transformer Encoder:** [CLS] attends to all patches via self-attention → builds global image summary
- **Classification Head:** Final [CLS] token passed to MLP → outputs class probabilities





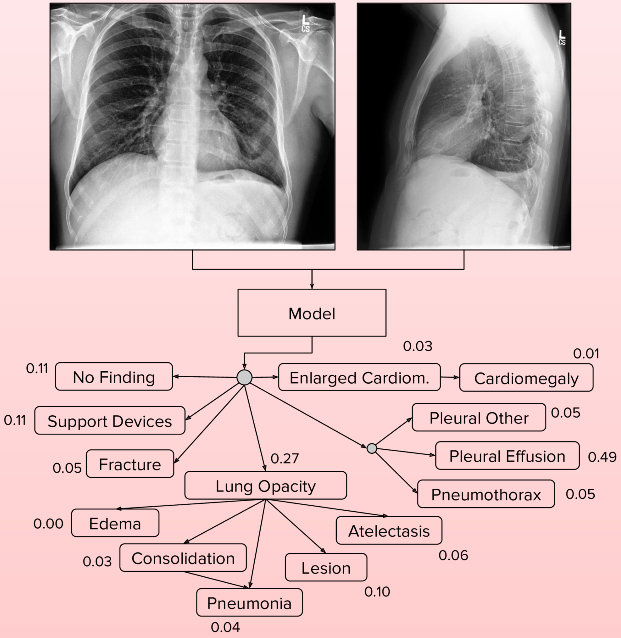
https://arxiv.org/abs/2010.11929
https://medium.com/analytics-vidhya/the-rise-of-attention-in-neural-networks-8c1d57a7b188

Vision Transformer Pretraining

- ViT-Base pretrained on a corpus of ~0.5M CXRs
- Fine-tuned on the CheXpert dataset of 224,316 CXRs from the AIMI Center

Dataset Description

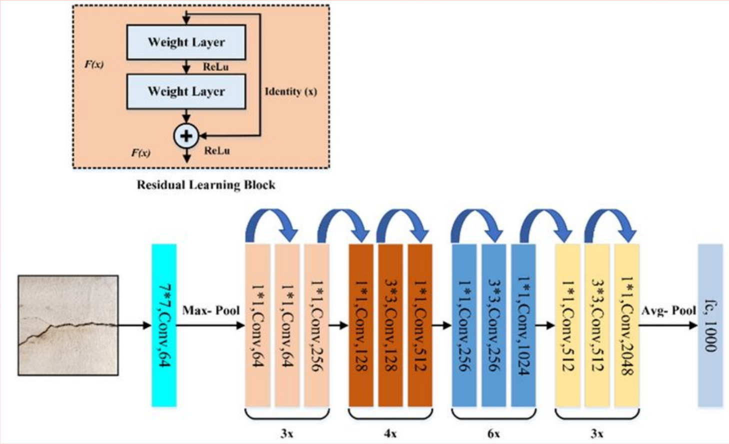
CheXpert is a dataset consisting of 224,316 chest radiographs of 65,240 patients who underwent a radiographic examination from Stanford Health Care between October 2002 and July 2017, in both inpatient and outpatient centers. Included are their associated radiology reports.

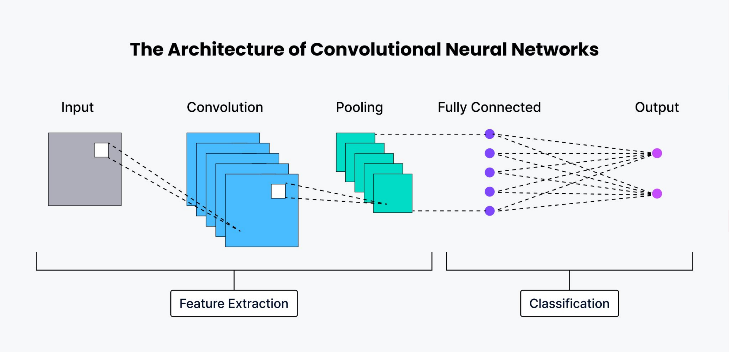


https://aimi.stanford.edu/datasets/chexpert-chest-x-rays
https://stanfordmlgroup.github.io/competitions/chexpert/

ResNet-18 with ImageNET Pretraining

- Shallow variant of the ResNet convolutional neural network trained on ImageNet-1K (2012).
- Convolutional layers extract features from an input image
- Pooling layers reduce the dimensions of data by the outputs of neuron clusters at one layer into a single neuron in the next layer.
- The fully connected layer connect every neuron in the penultimate layer to every neuron in the final layer (which correspond to all possible classes).
- ~11.7M parameters.

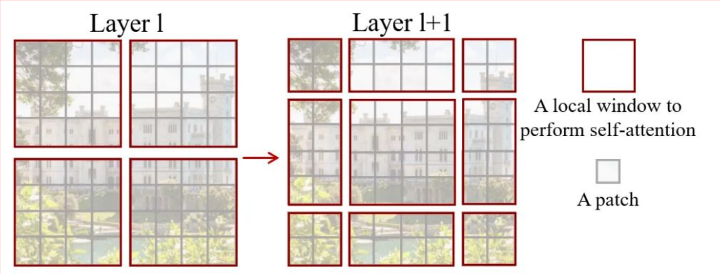


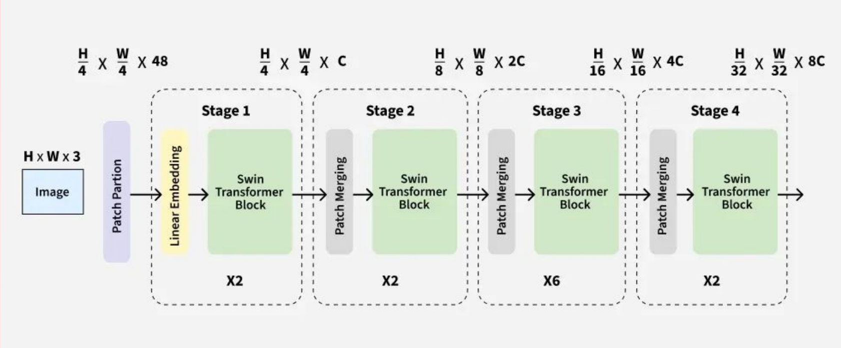


https://roboflow.com/model/resnet-50, https://zilliz.com/glossary/convolutional-neural-network

Sliding Windows (Swin) Transformer

- Each layer cuts the CXR into smaller, fixed-size patches → lower computational cost ($O(N)$ vs. $O(N^2)$) of measuring pixel interactions via self-attention.
- The patches are slid in future layers so that information from multiple discrete windows can be leveraged in classification.
- Four hierarchical stages (sometimes 96 → 192 → 384 → 768 channels) mimic a CNN pyramid for easy use in detectors/segmenters.
- ~88 M parameters; needs more GPU memory but captures long-range context.

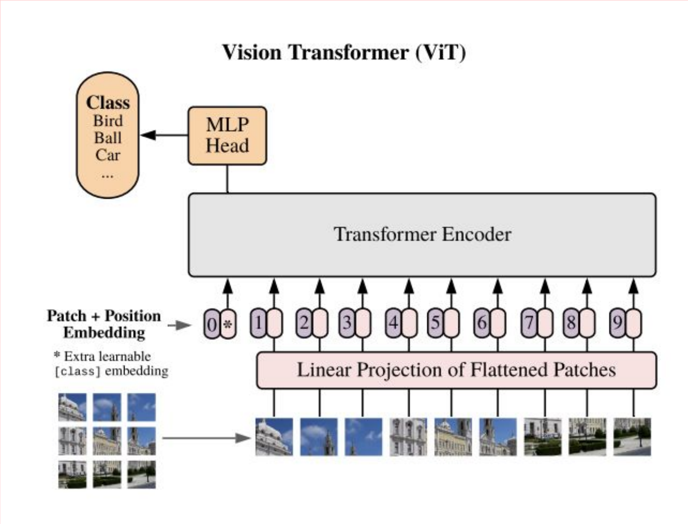


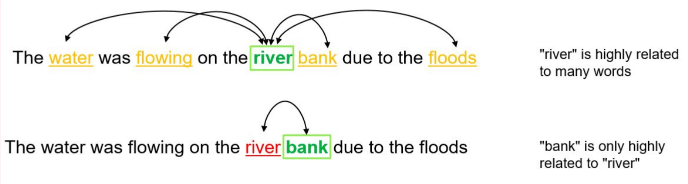


https://arxiv.org/pdf/2103.14030
https://www.geeksforgeeks.org/computer-vision/swin-transformer/

Vision Transformer (ViT-Base/16)

- Patch embedding: image → flattened 16 × 16 patches → linear layer → 196 patch tokens.
- Positional + token addition: prepend the [CLS] vector, add fixed position encodings to every token → 197-token sequence (box "0" in the figure is [CLS]).
- Transformer encoder: multi-head self-attention lets [CLS] attend to—and be attended by—every patch, accumulating a global summary of the input.
- The final [CLS] output feeds the small classifier MLP to produce class probabilities.
- Lowest parameter count (~8M).





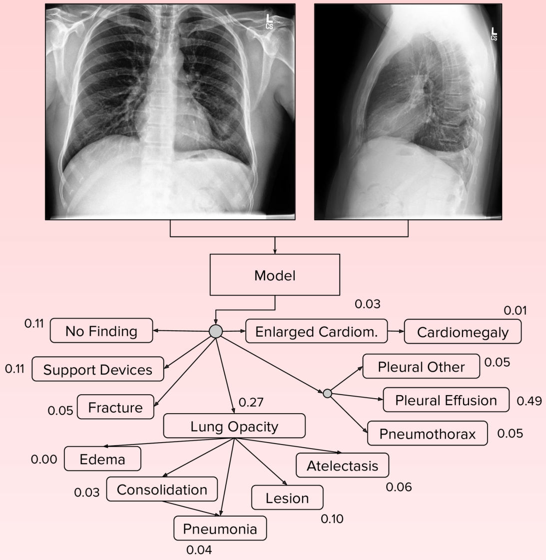
https://arxiv.org/abs/2010.11929
https://medium.com/analytics-vidhya/the-rise-of-attention-in-neural-networks-8c1d57a7b188

Vision Transformer Pretraining

We leverage a ViT-Base pretrained on a corpus of ~0.5M CXRs and fine-tuned on the CheXpert dataset of 224,316 CXRs from the AIMI Center.

Dataset Description

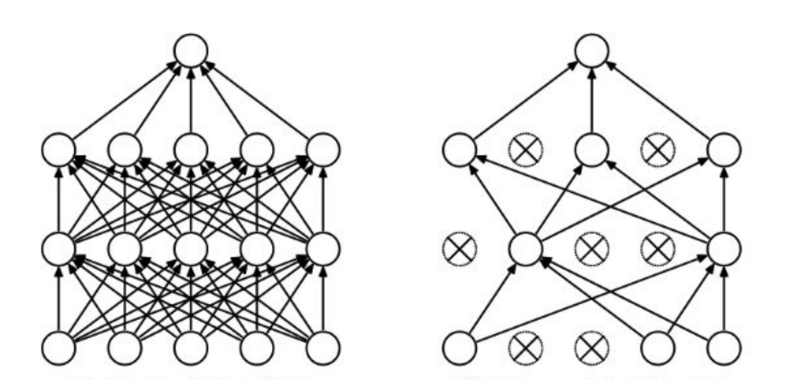
CheXpert is a dataset consisting of 224,316 chest radiographs of 65,240 patients who underwent a radiographic examination from Stanford Health Care between October 2002 and July 2017, in both inpatient and outpatient centers. Included are their associated radiology reports.



https://aimi.stanford.edu/datasets/chexpert-chest-x-rays
https://stanfordmlgroup.github.io/competitions/chexpert/

Further Optimizations–Dropout

- Helps prevent memorizing the training data to improve generalization to unseen cases.
- Disables neurons randomly during training to encourage learning in many sub networks.
- In final configuration, standard dropout was applied with:
  - 10% probability in the main body (backbone) of the ViT.
  - A variable, tuned probability (between 25% and 50%) in the final classification layer (head).
- Stochastic depth dropout (where entire layers are randomly skipped) with 30% probability was applied.



(a) Standard Neural Net (b) After applying dropout.

https://medium.com/data-science/dropout-in-neural-networks-47a162d621d9

Dropout Influence

Validation performance degrades ❌👎

```
Epoch 1 | Loss: 0.4373 | Train Acc: 0.8646 | Train AUC: 0.9260 | Val Acc: 0.8178 | Val AUC: 0.8900
Epoch 2 | Loss: 0.3491 | Train Acc: 0.8899 | Train AUC: 0.9530 | Val Acc: 0.8092 | Val AUC: 0.8784
Epoch 3 | Loss: 0.3022 | Train Acc: 0.9147 | Train AUC: 0.9800 | Val Acc: 0.7824 | Val AUC: 0.8724
Epoch 4 | Loss: 0.2475 | Train Acc: 0.9478 | Train AUC: 0.9822 | Val Acc: 0.8057 | Val AUC: 0.8749
Epoch 5 | Loss: 0.1917 | Train Acc: 0.9682 | Train AUC: 0.9928 | Val Acc: 0.8031 | Val AUC: 0.8867
```

Without dropout

Reduced overfitting to training data ✅

```
Epoch 1 | Loss: 0.6537 | Train Acc: 0.7636 | Train AUC: 0.8086 | Val Acc: 0.7660 | Val AUC: 0.8263
Epoch 2 | Loss: 0.5972 | Train Acc: 0.7775 | Train AUC: 0.8475 | Val Acc: 0.7686 | Val AUC: 0.8539
Epoch 3 | Loss: 0.5488 | Train Acc: 0.7880 | Train AUC: 0.8577 | Val Acc: 0.7876 | Val AUC: 0.8640
Epoch 4 | Loss: 0.5241 | Train Acc: 0.8141 | Train AUC: 0.8880 | Val Acc: 0.8031 | Val AUC: 0.8817
Epoch 5 | Loss: 0.4923 | Train Acc: 0.8254 | Train AUC: 0.8990 | Val Acc: 0.8126 | Val AUC: 0.8915
```

With dropout

Validation performance improves ✅👍

Further Optimizations: Leveraging Additional Unlabeled Data

- **Challenge:** Our initial dataset was limited in size and unbalanced.
- **Solution:** We incorporated a second dataset of chest X-rays labeled using the CheXpert labeler on corresponding radiology reports.

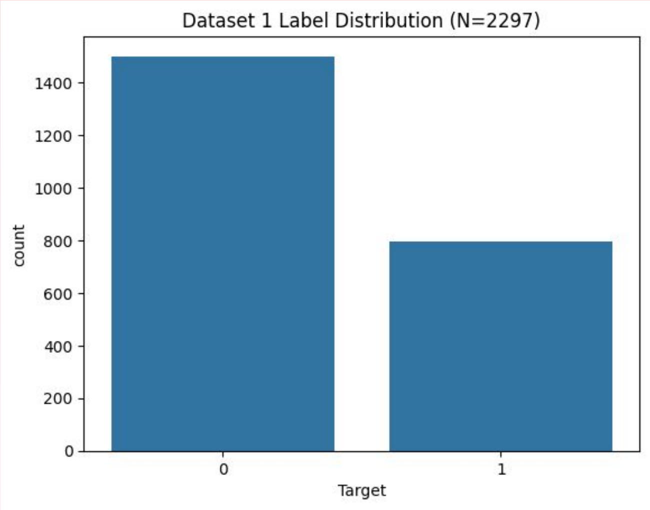
Dataset 1

( , 1 = pneumonia
0 = healthy)

Dataset 2

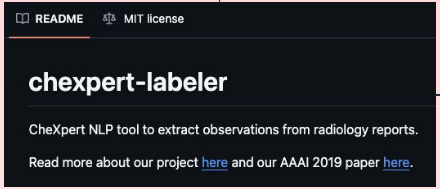
( , Radiology Notes)

Task: convert report to label.



CheXpert Labeling Setup

4233 Radiology Notes 📝



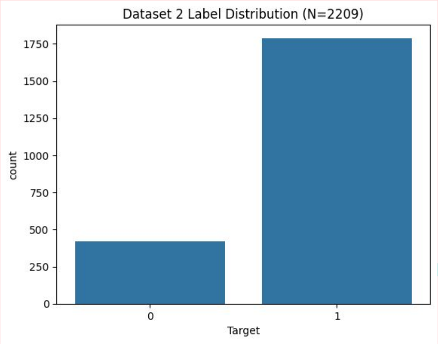
4233 Labels 🏷️
1 = pneumonia
0 = healthy
-1 = uncertain (less confident)

We tried:

- Marking uncertain cases as positive
- Marking uncertain cases as negative
- Removing uncertain cases

Removing uncertain cases yielded the largest, yet negligible (<0.1) AUROC improvement.

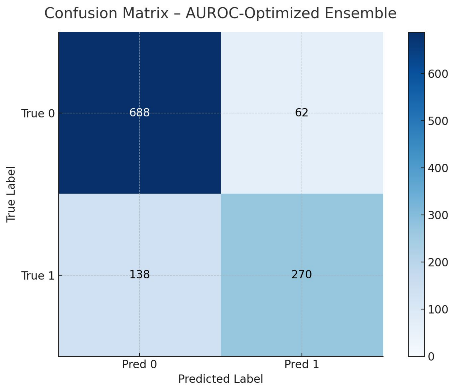
2209 additional (CXR, label) samples used for training

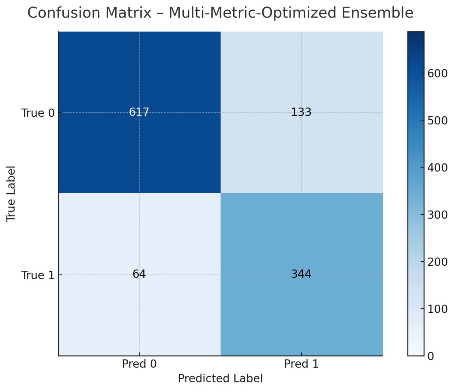


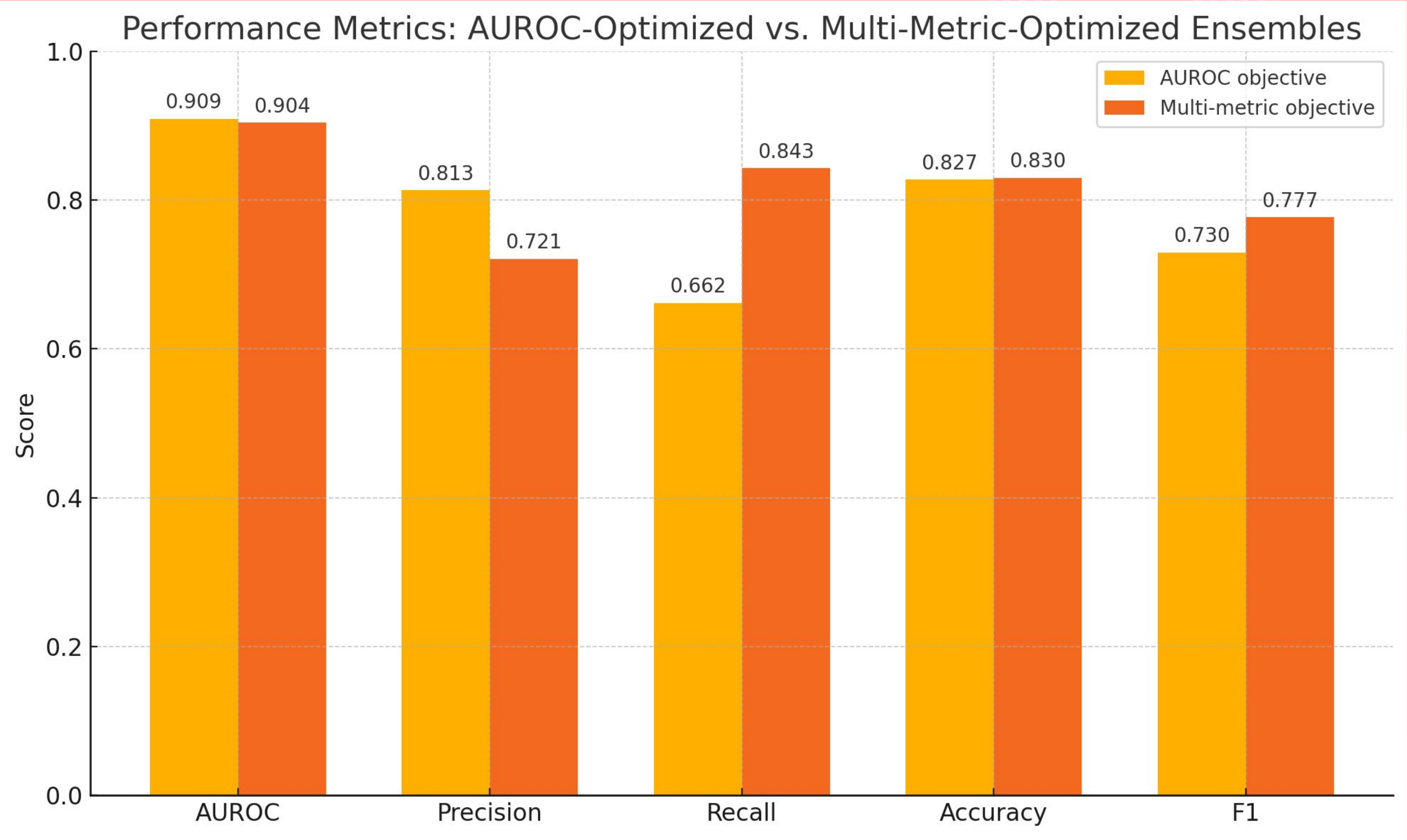
Further Optimizations: Hyperparameter Tuning

We used Optuna to efficiently test hundreds of combinations of hyperparameters.

- Initially tuned batch size, head dropout probability, learning rate, optimizer, and epochs with an objective of maximizing validation AUROC. We observed:
    - → low recall and F1 despite high AUROC.
    - → an alarming false negative count (see confusion matrices at right).
- Reran optimization with a weighted loss function that emphasized positive cases with a tuned weight; used an objective combining recall, f1, and AUROC.






Performance Metrics: AUROC-Optimized vs. Multi-Metric-Optimized Ensembles
AUROC objective
Multi-metric objective
Score
1.0
0.8
0.6
0.4
0.2
0.0
0.909
0.904
0.813
0.721
0.662
0.843
0.827
0.830
0.730
0.777
AUROC
Precision
Recall
Accuracy
F1

Further Optimizations: Ensembling

- Keep 4 best configs from Optuna optimization.
- Train each 3× → 12 models
- Average logits
- Optuna picks best 6-model subset & threshold

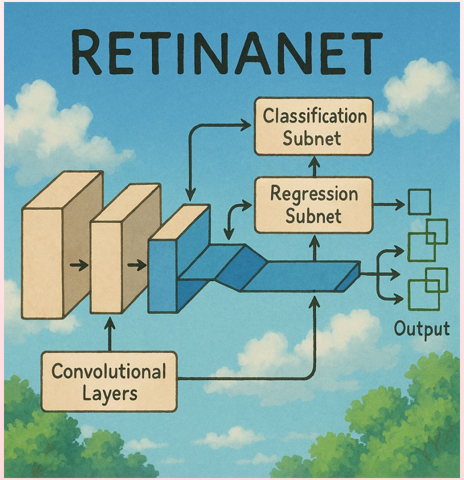
Metric	Best Single ViT Tuned via. Optuna	6-ViT Ensemble	Δ
AUROC	0.898	**0.904**	+0.006
Recall	0.826	**0.843**	+0.017
F1	0.761	**0.777**	+0.016

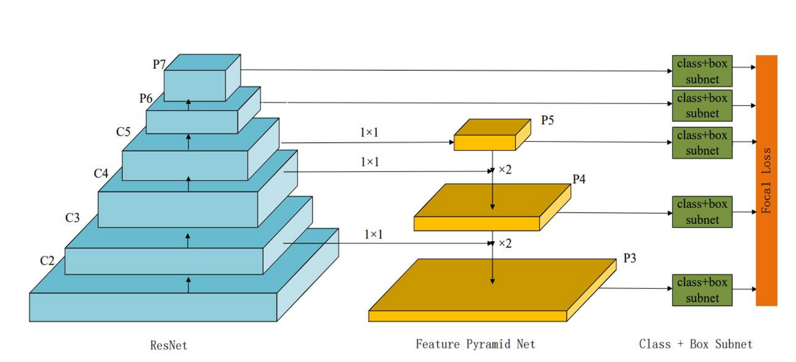
https://www.researchgate.net/figure/Schema-of-the-Pareto-front-the-ideal-point-and-the-final-optimal-solution-for-the_fig1_319080996

02 Localization Task

RetinaNet

- One stage detector with focal loss to address class imbalance
- ResNet50 backbone and FPN neck for multi-scale extraction
- Bounding box regression + classification
- Strong in small object detection

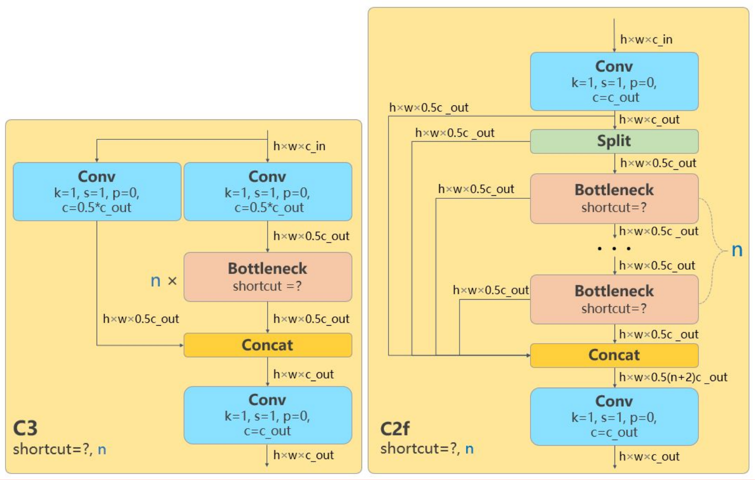


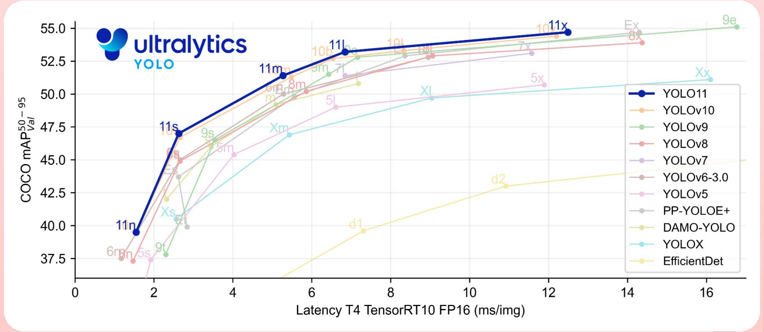


https://paperswithcode.com/method/retinanet
https://blog.stackademic.com/training-a-retinanet-for-custom-object-detection-43fa6e518372

YOLOv8/v11

- Tested model capacity by scaling from yolov8n → yolov8s → yolov8m
- Switched to YOLOv11
  - Higher mAP on COCO dataset with fewer parameters
  - ELAN-L creates more efficient gradient pathways
  - PGI provides clean path for learning signals to flow back





https://docs.ultralytics.com/models/yolo11/
https://mmyolo.readthedocs.io/en/latest/recommended_topics/algorithm_descriptions/yolov8_description.html

Performance Results

Best Run:

Metric	Value
Precision	0.75
Recall	0.69
F1 Score	0.72
mAP	0.47

Alterations:

- Trained on 1024×1024 images
- More epochs
- Heavy augmentations (mosaic, mixup, HSV)
- Optimized with AdamW + lr scheduling
- Early stopping & checkpointing

Final Pipeline

- Ran ViT classifier (pretrained on CheXpert) alongside YOLOv11 object detector
- If YOLO detects nothing but ViT is confident → fallback box with reduced confidence
- If YOLO detects pneumonia but ViT predicts negative → reduce YOLO box confidence
- Lowered confidence score when models disagreed
- Set output to "0" if both models predicted no pneumonia
- Merged final predictions for submission

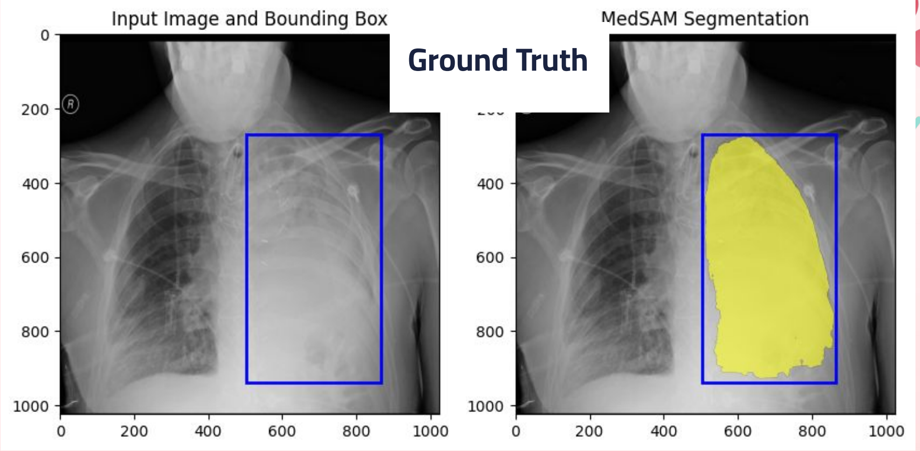
Results

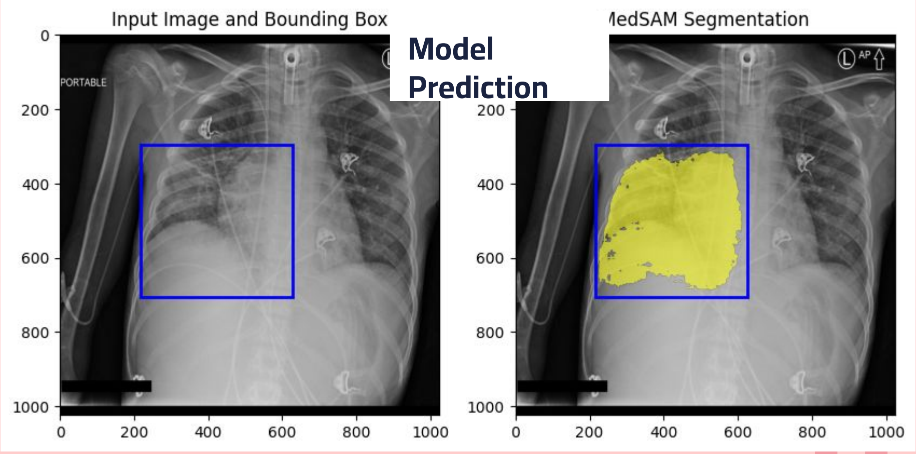
	Metric	Value
0	Final Score	0.429844
1	Classification F1	0.644444
2	Classification AUC	0.753333
3	Localization Precision	0.020000
4	Localization Recall	0.009780
5	Mean IoU	0.539798
6	mAP@[.50:.95]	0.101250

- Combining lowered localization performance → need to explore other methods
- In the future:
  - Further hyperparameter tuning
  - Explore larger datasets
  - Implement further ensembling strategies
- Key limitations included small training set and limited hyperparameter tuning

Segmentation

- MedSAM - ViT Base
- Provide Model with Bounding Box of Ground Truth and Object Detection Prediction
- Transform Image and Convert to Tensors
- Turn Tensor into MedSAM Embedding





https://colab.research.google.com/drive/19WNtRMbpsxeqimBlmJwtd1dzpalvK2FZ?usp=sharing
https://github.com/bowang-lab/MedSAM

Insights & Reflections

- We realized that the lack of data that we had was a big issue
  - With more time + planning, we could have yielded better performance.
- We did have to switch versions of models
  - ex. Switching from YOLO v8 to YOLO v11
- For the final pipeline, we had to determine what to do if the classification model reached a different conclusion than the object detector.
- We were initially focused on the area under the curve, and realized that other evaluation metrics such as recall/sensitivity were extremely important in this setting as well.
- We tried many different approaches, and ended up taking different routes than we initially thought.

Thank you