Three Experiments on CNNs for Object Detection in Chest X-rays

Terry Griffin

COMP.5300.205 Deep Learning

Spring 2020

Overview

This project consists of three experiments on Convolutional Neural Networks (CNNs) used for the detection of manifestations related to Tuberculosis in X-ray images.

1. Reducing the number of parameters for transfer learning with grayscale images.
2. Comparing the performance of R-CNN, Mask R-CNN, Cascade R-CNN, and Cascade Mask R-CNN.
3. Using manifestation specific location information to reduce false positives.

1. Transfer Learning

- Uses the weights from a previously trained network as a starting point.
- Many projects use weights from the ImageNet challenge.
  - Image classification task.
  - 1.2 million images.
  - 1,000 classes.
- Fine-tune the full network or only part for the new task.
  - Deep learning frameworks allow enabling or disabling of training each layer.

Transfer Learning for X-ray Image

- Many projects use ImageNet pretrained weights.
- ImageNet contains color (3 channel) images, X-rays are grayscale (1 channel).
- Can we modify a model to use 1 channel input and still use the pre-trained weights?
- Is the reduction in the number of parameters significant?

Converting a Model to Single Channel Input

Instead of expanding the grayscale image from one channel to three, we can modify the kernels to work on a single channel image:

Replace

$$S(i,j)=\sum_{M}\sum_{N}\sum_{D} I_3(i+m,j+n,d)\,K_3(m,n,d)$$

with

$$S(i,j)=\sum_{M}\sum_{N} I_1(i+m,j+n)\,K_2(m,n)$$

where

$$\sum_{D} I_3(i,j,d)=3\,I_1(i,j)$$

$$K_2(m,n)=3\sum_{D} K_3(m,n,d)$$

Pytorch Code Model Conversion

Converting a PyTorch model from three channels to single channel input requires two changes to the first convolutional layer:

1. Set the number of input channels to 1
2. Update the parameter weights (kernels) by summing across the depth dimension and multiplying by 3.

```
def resnet_grayscale_model(model):
    conv = model.conv1
    if conv.in_channels > 1:
        with torch.no_grad():
            conv.in_channels = 1
            conv.weight = nn.Parameter(conv.weight.sum(dim=1, keepdim=True) *
3)

def densenet_grayscale_model(model):
    conv = model.features.conv0
    if conv.in_channels > 1:
        with torch.no_grad():
            conv.in_channels = 1
            conv.weight = nn.Parameter(conv.weight.sum(dim=1, keepdim=True)* 3)
```

Notes: Different models require custom implementation dependent code

Parameter Reduction

Comparison of the number of parameters for three channel and single channel models.
Some selected results:

Model	Number of Parameters for Three Channel Input	Number of Parameters for One Channel Input	Difference
ResNet18	11,689,512	11,683,240	0.054%
ResNet50	25,557,032	25,550,760	0.025%
ResNet101	44,549,160	44,542,888	0.014%
DenseNet121	7,978,856	7,972,584	0.079%
DenseNet169	14,149,480	1,4143,208	0.044%

Conclusions

- The reduction in the number of parameters is insignificant.
- The model implementation dependent conversion code is a weak point.

2. Comparing Models for Object Detection

- As part of previous work a Mask R-CNN model for object detection of four TB manifestations was created.
- This experiment compares the performance of four models on each of the four manifestations:
  - Mask R-CNN (the previously developed model)
  - R-CNN
  - Cascade Mask R-CNN
  - Cascade R-CNN

Implementation Details

- Models were created using the PyTorch framework and the Detectron2 library.
- The dataset for each manifestation was divided into: 80% training, 10% validation, 10% test
- Each model architecture was tuned separately.
- The COCO mean average precision (mAP) and average precision at a threshold of 0.50 (AP50) are used evaluation.

Example Results

Results for the Airspace Consolidation manifestation.
Other manifestations show similar trends between models.

Network	Backbone	mAP	AP50
R-CNN	ResNet-50	0.3625	0.7184
Mask R-CNN	ResNet-50	0.3313	0.6705
Cascade R-CNN	ResNet-50	0.3718	0.7094
Cascade Mask R-CNN	ResNet-50	0.3299	0.6785
R-CNN	ResNet-101	0.4002	0.7990
Mask R-CNN	ResNet-101	0.3583	0.7871
Cascade R-CNN	ResNet-101	0.3992	0.7882
Cascade Mask R-CNN	ResNet-101	0.3602	0.7891

Conclusions

- R-CNN outperforms Mask R-CNN by a small amount.
  - Possibly due to the easier task
- Cascade versions did not show any improvement over base models.
  - May be due to the limited size of the dataset

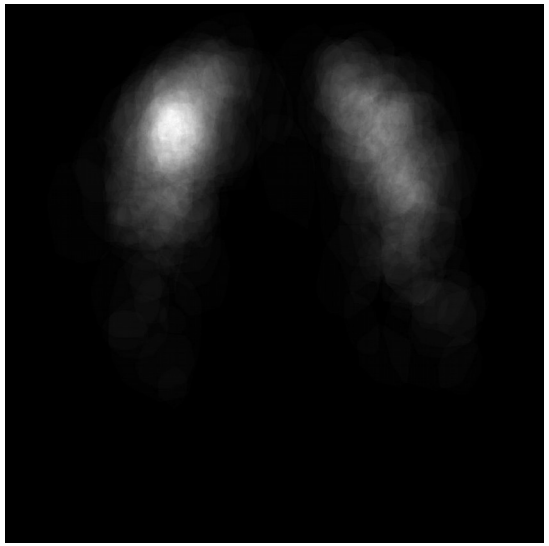
3. Using Location Information to Reduce False Positives

- CNNs are inherently translation independent.
- Some TB manifestations occur only in certain locations in the lungs.
- Create a heatmap for each manifestation based on the locations in the training and validation set.
- Use the heatmap as a filter in a post inference step.

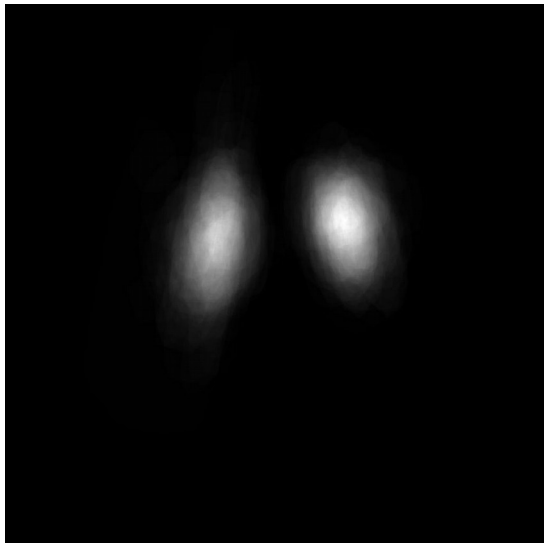
Manifestation Heatmaps

Airspace Consolidation

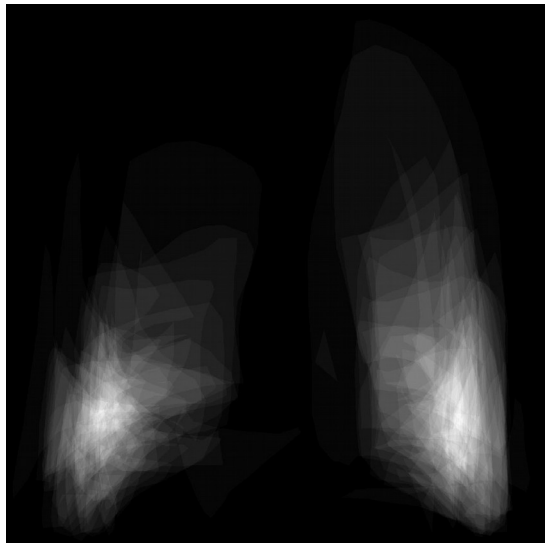
Cavitation



Lymphadenopathy



Pleural Effusion



Airspace consolidation and Cavitation do not show a strong relationship with location, while Lymphadenopathy and Plueral Effusion do

Implementation

- The heatmaps were incorporated into the processing of the evaluation function.
- Any instance whose bounding box on the heatmap did not contain any value above a given threshold was removed.
- The evaluation function is used on the validation and test sets. The data was not fed back into the network during training (a good next step).

Results

- No difference in performance for Airspace Consolidation and Cavitation.
- No additional false negatives were created for Lymphadenopathy or Pleural Effusion.
- For Lymphadenopathy 8 out of 40 false positives were removed.
- For Pleural Effusion 3 out of 8 false positives were removed.

Questions

?