Medical Image Processing with Deep Learning

---- Mammograms Classification and Automatic Tumor detection

Members: Yijie Jin (CityU), Yifan Zheng (CUHK)
Mentors: Dr. Kwai Wong, Dr. Raymond Chan
Background

- Breast cancer is the second leading cause of cancer-related death among women in the United States.
- Roughly one eighth of women in the United States will develop breast cancer during their lifetimes.
- Five-year relative survival rates can be up to 3-4 times higher for cancers detected at an early stage versus at a later stage.
Mammography

- Mammography is the most common breast screening technology. It is the process of using low-energy X-rays to examine the human breast for diagnosis and screening. It is the most reliable method for screening breast abnormalities before they become clinically palpable.
- Reading mammograms is a tedious and error-prone process, and not all radiologists achieve uniformly high levels of accuracy.
Goals

- Classify mammograms into three classes, normal, benign and malignant (CNNI-BCC and VGG16)
- Automatically detect the tumor without prior information of the presence of a cancerous lesion (IDBLL)
Challenges

● Hard to find a database due to privacy reasons -> a public database (MIAS)
● Mammograms usually have a low contrast -> Remove black background
● “needle in a haystack” nature of mammogram classification -> Cut the images into small patches
Dataset and Data Preprocessing

Dataset: mini-MIAS database of mammograms

- 322 images in total

Data Augmentation:

- Rotation (by 90, 180, 270 degrees respectively)
- Flip (vertically)
- Equally Sampling (1024×1024 -> 128×128 and 1024×1024 -> 256×256)
- Sample with overlap

Data Cleansing:

- Remove the image patches with black background
Dataset - mini-MIAS

- Labelled
- Have information about the coordinates of tumor center
- Have information about the radius of the tumor
Procedure

Data Preprocessing
- 1024 x 1024 images
- Rotation
- Flip
- Crop
- Tumor centered: 128 x 128
- Not tumor-centered: 128 x 128

Classification
- 6-layer CNN: BCC
- Normal
- Benign
- Malignant

Object Detection
- VGG16
- Automatic Lesion Locator
Data Augmentation

---- Flip/Rotation

Rotation clockwisely by 90 degrees

Flip vertically
Data Augmentation

Equally Sampling

1024×1024

256×256
Data Augmentation
---- Sample with Overlap

- Sample the image patches every 128 pixels.
- 1 -> 49
Data Cleansing

---- Remove images with black background
Relabel the Image Patches

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
</tbody>
</table>

Before

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

After
Problems

- Tumors are separated
- Hard to put the small patches together
Data Augmentation

- Tumor-centered Sampling

- 1024×1024 → 256×256
- Tumor center at the image center
Convolutional Neural Network Improvement for Breast Cancer Classification (CNNI-BCC)

- 30 layers (28 convolutional layers + 1 pooling layer + 1 fully connected layer)
- Implemented in Keras with TensorFlow backend
Experiments

<table>
<thead>
<tr>
<th>Layer (type)</th>
<th>Output Shape</th>
<th>Param #</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv2d_1 (Conv2D)</td>
<td>(None, 64, 64, 32)</td>
<td>320</td>
</tr>
<tr>
<td>depthwise_conv2d_1 (DepthwiseConv2D)</td>
<td>(None, 64, 64, 32)</td>
<td>32800</td>
</tr>
<tr>
<td>conv2d_2 (Conv2D)</td>
<td>(None, 64, 64, 64)</td>
<td>2112</td>
</tr>
<tr>
<td>depthwise_conv2d_2 (DepthwiseConv2D)</td>
<td>(None, 32, 32, 64)</td>
<td>262208</td>
</tr>
<tr>
<td>average_pooling2d_1 (AveragePooling2D)</td>
<td>(None, 8, 8, 64)</td>
<td>0</td>
</tr>
<tr>
<td>flatten_1 (Flatten)</td>
<td>(None, 4096)</td>
<td>0</td>
</tr>
<tr>
<td>dense_1 (Dense)</td>
<td>(None, 3)</td>
<td>12291</td>
</tr>
</tbody>
</table>
Results

The loss on the test set is: 0.14544324301610326

The accuracy on the test set is: 0.7324840809888901
VGG16
VGG16 Architecture

- 5 blocks
- Convolutional Layer + Pooling Layer in each block
- 3 fully connected layer and softmax function
Data Shuffling

<table>
<thead>
<tr>
<th>Normal</th>
<th>Benign</th>
<th>Malignant</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9</td>
<td>0.1</td>
<td></td>
</tr>
</tbody>
</table>

Load the data in sequence

Split training and validation set

BMBMNMBNMBNNMNBNMNNBMN…NMNMBNBNBMNMBMNMBNMBNMB

After shuffling

Problems:
- The training data is not general.
- The neural network cannot enough features of malignant cases.
Cross-Validation

- 10-Fold Cross-validation
- Train 10 different models with different validation sets.
Experiments
--- Classify mammograms into 2 categories

- Training set: 2520 image patches (128×128)
- Epoch number: 80
- Batch size: 32
Experiments

-- Classify mammograms into 3 categories

- Training set: 11340 image patches
- Epoch number: 60
- Batch size: 32

Model Accuracy

Model Loss
Normal Convolution and Separable Convolution

**Normal Convolution**

- Filter number: 4
- Filter size: $3 \times 3$
- Channel number: 3
- Total number of parameters: $4 \times 3 \times 3 \times 3 = 108$
Normal Convolution and Separable Convolution

**Depthwise Convolution**
- Filter number: 3
- Filter size: 3×3
- Channel number: 3
- Number of parameters: $3 \times 3 \times 3 = 27$

**Pointwise Convolution**
- Filter number: 4
- Filter size: 1×1
- Channel number: 3
- Number of parameters: $1 \times 1 \times 3 \times 4 = 12$

Total number of parameters: $27 + 12 = 39$
Comparison between Normal Convolution and Separable Convolution

<table>
<thead>
<tr>
<th></th>
<th>Number of Parameters in 1 Layer</th>
<th>Total Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv2D</td>
<td>1,792</td>
<td>23,715,730</td>
</tr>
<tr>
<td>SeparableConv2D</td>
<td>283</td>
<td>4,820,525</td>
</tr>
</tbody>
</table>

- Keras implementation: SeparableConv2D
Interactive Detection Based Lesion Locator (IDBLL)
Structure of Single Shot MultiBox Detector (SSD)

Base Network for classification
+ Multi-scale feature maps for detection
+ Convolutional predictors for detection: multiple classes confidences
+ Default boxes and aspect ratios: localization
Prior Box & Objective Loss Function

\[
L(x, c, l, g) = \frac{1}{N} (L_{conf}(x, c) + \alpha L_{loc}(x, l, g))
\]
Training Process

- **Input images**
- Generate **feature maps** of different scales
- Generate **anchor boxes** for each feature map
- Determine **positive or negative** for each anchor box
- Train the parameters of the layer for **classification**
- Train the parameters of the layer for **localization**
CPU & GPU We Used

**CPU**
- 128GB
- 28 cores
- 8TB on-node storage

**GPU**
- 2 NVIDIA Tesla K80
  - Kepler architecture
- 12GB/GPU
- 48GB total/node

**GPU**
- 8 NVIDIA Volta V100
- 16GB/GPU
- 128GB total/node
**Experiment A**

--- Best Result

- **Best Result**
  - 50 epochs: 5.06561
  - 110 epochs: 3.14813

- **Converges slow**
  - Time consuming
  - 760s/epoch

**Training Setting**
- 110 epochs
- Batch size 100
- Steps per epoch 60
Experiment A

---- Best Result

<table>
<thead>
<tr>
<th></th>
<th>Benign AP</th>
<th>Malignant AP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.839</td>
<td>0.846</td>
</tr>
<tr>
<td>mAP</td>
<td>0.842</td>
<td></td>
</tr>
</tbody>
</table>
Precision-Recall Curve & mAP calculation

\[
\text{Precision} = \frac{TP}{TP + FP}
\]

\[
\text{Recall} = \frac{TP}{TP + FN}
\]

where TP = True Positive, FP = False Positive, FN = False Negative.

In our case of testing for cancer:

\[
\text{Precision} = \frac{TP}{\text{Total Positive Results Shown by Model}}
\]

\[
\text{Recall} = \frac{TP}{\text{Total Cancer Cases in Ground Truth}}
\]

<table>
<thead>
<tr>
<th>Rank</th>
<th>Correct?</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>True</td>
<td>1.0</td>
<td>0.2</td>
</tr>
<tr>
<td>2</td>
<td>True</td>
<td>1.0</td>
<td>0.4</td>
</tr>
<tr>
<td>3</td>
<td>False</td>
<td>0.07</td>
<td>0.4</td>
</tr>
<tr>
<td>4</td>
<td>False</td>
<td>0.5</td>
<td>0.4</td>
</tr>
<tr>
<td>5</td>
<td>False</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>6</td>
<td>True</td>
<td>0.5</td>
<td>0.6</td>
</tr>
<tr>
<td>7</td>
<td>True</td>
<td>0.57</td>
<td>0.8</td>
</tr>
<tr>
<td>8</td>
<td>False</td>
<td>0.5</td>
<td>0.8</td>
</tr>
<tr>
<td>9</td>
<td>False</td>
<td>0.44</td>
<td>0.8</td>
</tr>
<tr>
<td>10</td>
<td>True</td>
<td>0.5</td>
<td>1.0</td>
</tr>
</tbody>
</table>

\[
\text{AP} = \frac{1}{20} \sum_{r \in \{0.0, ..., 1.0\}} p_{\text{interp}}(r)
\]

\[
p_{\text{interp}}(r) = \max_{\hat{r} \geq r} p(\hat{r})
\]
Experiment B

---- Increasing Learning Rate

def lr_schedule(epoch):
    if epoch < 80:
        return 0.001
    elif epoch < 90:
        return 0.0001
    else:
        return 0.00001

def lr_schedule(epoch):
    if epoch < 80:
        return 0.002
    elif epoch < 90:
        return 0.0002
    else:
        return 0.00002
Experiment C
---- Enlarging Batch Size

Usually, enlarging batch size:
- accelerate processing speed
- determine the direction of descent more accurately

However, our test shows:

<table>
<thead>
<tr>
<th>Batch Size</th>
<th>Steps per Epoch</th>
<th>Epoch</th>
<th>Validation Loss</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>2</td>
<td>300</td>
<td>9.24585</td>
<td>87s/epoch</td>
</tr>
<tr>
<td>200</td>
<td>3</td>
<td>300</td>
<td>8.64310</td>
<td>80s/epoch</td>
</tr>
<tr>
<td>100</td>
<td>60</td>
<td>30</td>
<td>7.024</td>
<td>760s/epoch</td>
</tr>
</tbody>
</table>
### Experiment D
---- Using Separable Convolution

<table>
<thead>
<tr>
<th>Batch Size</th>
<th>Steps per Epoch</th>
<th>Epoch</th>
<th>Validation Loss</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>6</td>
<td>150</td>
<td>4.36554</td>
<td>245s/epoch</td>
</tr>
<tr>
<td>200</td>
<td>3</td>
<td>150</td>
<td>4.80492</td>
<td>245s/epoch</td>
</tr>
<tr>
<td>300</td>
<td>2</td>
<td>150</td>
<td>5.03458</td>
<td>245s/epoch</td>
</tr>
</tbody>
</table>

- Take more time to train per epoch
- Converge faster than Conv2D-model in terms of epochs
- Converge slower than Conv2D-model in terms of real time
Experiment E

--- Testing on More General Cases

Test on uniformly cut set
Experiment E (Cont.)

Retraining on More General Cases

Uniformly cut data set
- 3500 in training set
- 500 in validation set
- 329 in test set

Training Setting
- 20 epochs
- batch size 100
- steps per epoch 35

![Training Records — SSD_256_retrained](image)
Experiment E (Cont.)

----Retraining on More General Cases

<table>
<thead>
<tr>
<th>Tumorcenter test set</th>
<th>Uniformly cut test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benign AP</td>
<td>0.742</td>
</tr>
<tr>
<td></td>
<td>0.394</td>
</tr>
<tr>
<td>Malignant AP</td>
<td>0.54</td>
</tr>
<tr>
<td></td>
<td>0.332</td>
</tr>
<tr>
<td>mAP</td>
<td>0.641</td>
</tr>
<tr>
<td></td>
<td>0.363</td>
</tr>
</tbody>
</table>

Before retraining:

<table>
<thead>
<tr>
<th>Tumorcenter test set</th>
<th>Uniformly cut test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benign AP</td>
<td>0.839</td>
</tr>
<tr>
<td></td>
<td>0.310</td>
</tr>
<tr>
<td>Malignant AP</td>
<td>0.846</td>
</tr>
<tr>
<td></td>
<td>0.257</td>
</tr>
<tr>
<td>mAP</td>
<td>0.842</td>
</tr>
<tr>
<td></td>
<td>0.284</td>
</tr>
</tbody>
</table>
Related Work

- P Xi. et.al. IEEE MeMeA 2018: VGGNet for classification and ResNet for localizing abnormalities
- TG Debelee. LNICST, volume 244: PCA for feature dimensionality reduction, KNN for classification, Particle Swarm Optimized Wavelet Neural Network (PSOWNN)
Future Works

- Enhance the contrast of mammograms
- Use ResNet to reduce the training time.
- Use more traditional machine learning method, such as k-nearest neighbors algorithm (KNN), support vector machine for classification
- Use other public database, e.g.: Digital Database for Screening Mammography (DDSM)
References


Thank you for listening!