Deep Learning

Columbia University
Iddo Drori, Summer 2019
Lectures

- **First Day of Classes (Tuesday, January 22)**
- Lecture 1 (Monday, July 8): Introduction
- Lecture 2 (Tuesday, July 9): Backpropagation
- Lecture 3 (Wednesday, July 10): Optimization
- **Lecture 4 (Thursday, July 11): CNN’s**
- Lecture 5 (Monday, July 15): CNN’s
- Lecture 6 (Tuesday, July 16): Regularization
- Lecture 7 (Wednesday, July 17): Sequence Models
- Lecture 8 (Thursday, July 18): Sequence Models
- Lecture 9 (Monday, July 22): Generative Models
- Lecture 10 (Tuesday, July 23): Generative Models
- Lecture 11 (Wednesday, July 24): Generative Models
Convolutional Neural Networks
Number of weights between two layers of fully connected neural network is multiplication of layer sizes.

Locally sharing weights reduces number of weights to a constant.
1D Convolution

\[(f \ast g)(i) = \sum_{j=-k}^{k} g(j) f(i - j)\]
1D Convolution

Input:
\[
x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_8 \quad x_9 \quad x_{10}
\]

Filter:
\[
k_1 \quad k_2 \quad k_3 \quad k_1 \quad k_2 \quad k_3
\]

Output:
\[
y_2 \quad \quad \quad \quad \quad \quad \quad \quad y_9
\]
1D Convolution

Input:

Padding (zero)

| 0 | x₁ | x₂ | x₃ | x₄ | x₅ | x₆ | x₇ | x₈ | x₉ | x₁₀ | 0 |

Filter:

| k₁ | k₂ | k₃ |

Output:

| y₁ | y₂ |

Padding
1D Convolution: Identity

Input: $x_1 \ x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7 \ x_8 \ x_9 \ x_{10}$

Filter: $0 \ 1 \ 0$

Output: $x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7 \ x_8 \ x_9 \ x_{10}$

Padding (reflection)
1D Convolution: Average

Padding (reflection)
Input:

\[
\begin{array}{cccccccccc}
& & & & & & & & & \\
x_1 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{10} \\
\end{array}
\]

Filter:

\[
\frac{1}{3} \cdot \begin{array}{ccc}
1 & 1 & 1 \\
\end{array}
\]

Output:

\[
\begin{array}{cccccccccccc}
& & & & & & & & & & & & & & & & \\
& & & & & & & & & & & & & & & & \\
\end{array}
\]

Padding
1D Convolution: Sharpen

Padding (reflection)

Input:

\[
\begin{array}{cccccccccc}
X_1 & X_1 & X_2 & X_3 & X_4 & X_5 & X_6 & X_7 & X_8 & X_9 & X_{10} & X_{10} \\
\end{array}
\]

Filter:

\[
\begin{array}{c}
-1 & 2 & -1 \\
\end{array}
\]

Output:

\[-x_1 + 2x_2 - x_3\]
1D Convolution: Matrix Multiplication

\[ K_X^T = Y^T \]

8x10 10x1 8x1

\[
\begin{array}{cccccccccccc}
  k_1 & k_2 & k_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  0 & k_1 & k_2 & k_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & k_1 & k_2 & k_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & k_1 & k_2 & k_3 & 0 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & k_1 & k_2 & k_3 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & 0 & k_1 & k_2 & k_3 & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & 0 & 0 & k_1 & k_2 & k_3 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & 0 & 0 & 0 & k_1 & k_2 & k_3 & 0 & 0 \\
  0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_1 & k_2 & k_3 & 0 \\
\end{array}
\]

\[ K \]

\[
\begin{array}{cccc}
  x_1 & x_2 & x_3 & x_4 \\
  y_1 & y_2 & y_3 & y_4 \\
  x_5 & x_6 & x_7 & x_8 \\
  y_5 & y_6 & y_7 & y_8 \\
  x_9 & x_{10} \\
  y_9 \\
\end{array}
\]
Sharing Weights: 1D Convolution

\[ y = k \ast x = Kx = k_1 Sx + k_2 Sx + k_3 Sx \]

filter

\[ k = [k_1 \ k_2 \ k_3] \]

\[
\begin{bmatrix}
  y_1 \\
  y_2 \\
  y_3 \\
  y_4 \\
  y_5 \\
  y_6 \\
  y_7 \\
\end{bmatrix} =
\begin{bmatrix}
  k_1 & k_2 & k_3 & 0 \\
  0 & k_1 & k_2 & k_3 & 0 \\
  0 & 0 & k_1 & k_2 & k_3 & 0 \\
  0 & 0 & 0 & k_1 & k_2 & k_3 & 0 \\
  0 & 0 & 0 & 0 & k_1 & k_2 & k_3 \\
  0 & 0 & 0 & 0 & 0 & k_1 & k_2 \\
\end{bmatrix}
\begin{bmatrix}
  x_1 \\
  x_2 \\
  x_3 \\
  x_4 \\
  x_5 \\
  x_6 \\
  x_7 \\
\end{bmatrix}
\]

\[ S =
\begin{bmatrix}
  1 & 0 & 0 & 0 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 1 & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 1 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\end{bmatrix}
\]

\( K \) is a band diagonal Toeplitz matrix

elements repeated along diagonal

\[ \frac{\partial y}{\partial k_i} = Sx \]

simple partial derivatives with respect to each weight

1's diagonal shift matrix
2D Convolution

\[(f \ast g)(i, j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} g(i, j) f(i - u, j - v)\]
2D Convolution
Convolution

Commutative \[ f \star g = g \star f \].

Associative \[ f \star (g \star h) = (f \star g) \star h \].

Distributive \[ f \star (g + h) = f \star g + f \star h \].

Differentiation \[ \frac{d}{dx} (f \star g) = \frac{df}{dx} \star g = f \star \frac{dg}{dx} \].
2D Convolution

Input:

\[
\begin{array}{cccccccc}
  x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} & x_{17} \\
  x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & x_{26} & x_{27} \\
  x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & x_{36} & x_{37} \\
  x_{41} & x_{42} & x_{43} & x_{44} & x_{45} & x_{46} & x_{47} \\
  x_{51} & x_{52} & x_{53} & x_{54} & x_{55} & x_{56} & x_{57} \\
  x_{61} & x_{62} & x_{63} & x_{64} & x_{65} & x_{66} & x_{67} \\
  x_{71} & x_{72} & x_{73} & x_{74} & x_{75} & x_{76} & x_{77} \\
\end{array}
\]

Filter:

\[
\begin{array}{ccc}
  k_{11} & k_{12} & k_{13} \\
  k_{21} & k_{22} & k_{23} \\
  k_{31} & k_{32} & k_{33} \\
\end{array}
\]

\[ y_{22} = k_{11}x_{11} + k_{12}x_{12} + k_{13}x_{13} + ... + k_{33}x_{33} \]
2D Convolution

Convolution with 5x5 kernel
2D Convolution

Convolution with 7x7 kernel
2D Convolution

Convolution with 3x3 kernel

Convolution with 3x3 kernel

Convolution with 3x3 kernel
2D Convolution: Separable Kernels

\[
\begin{array}{c}
\frac{1}{3} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \text{Box blur}
\end{array}
\]

\[
\frac{1}{4} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad \approx \text{Gaussian blur}
\]

\[
\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad \text{Sobel edge detection}
\]

\[
\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \quad \text{Prewitt edge detection}
\]
Sharing Weights: 2D Convolution

$k = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix}$

$\begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} \\ x_{31} & x_{32} & x_{33} & x_{34} & x_{35} \\ x_{41} & x_{42} & x_{43} & x_{44} & x_{45} \\ x_{51} & x_{52} & x_{53} & x_{54} & x_{55} \end{bmatrix}$

$K$ is a block Toeplitz matrix

Toeplitz matrices repeated along diagonals
Convolution: Color Image
Convolution: Color Image

\[
\begin{array}{cccccc}
X_{11} & X_{12} & X_{13} & X_{14} & X_{15} & X_{16} \\
X_{21} & X_{22} & X_{23} & X_{24} & X_{25} & X_{26} \\
X_{31} & X_{32} & X_{33} & X_{34} & X_{35} & X_{36} \\
X_{41} & X_{42} & X_{43} & X_{44} & X_{45} & X_{46} \\
X_{51} & X_{52} & X_{53} & X_{54} & X_{55} & X_{56} \\
X_{61} & X_{62} & X_{63} & X_{64} & X_{65} & X_{66} \\
X_{71} & X_{72} & X_{73} & X_{74} & X_{75} & X_{76} \\
\end{array}
\]

- **Red**
- **Green**
- **Blue**

\[
\begin{array}{cccc}
Y_{22} \\
Y_{60} \\
\end{array}
\]
Convolution: Color Image
## 2D Convolution

### Input:

Padding (reflection)

<table>
<thead>
<tr>
<th>x_{11}</th>
<th>x_{12}</th>
<th>x_{13}</th>
<th>x_{14}</th>
<th>x_{15}</th>
<th>x_{16}</th>
<th>x_{17}</th>
<th>x_{17}</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_{11}</td>
<td>x_{11}</td>
<td>x_{12}</td>
<td>x_{13}</td>
<td>x_{14}</td>
<td>x_{15}</td>
<td>x_{16}</td>
<td>x_{17}</td>
</tr>
<tr>
<td>x_{21}</td>
<td>x_{21}</td>
<td>x_{22}</td>
<td>x_{23}</td>
<td>x_{24}</td>
<td>x_{25}</td>
<td>x_{26}</td>
<td>x_{27}</td>
</tr>
<tr>
<td>x_{31}</td>
<td>x_{31}</td>
<td>x_{32}</td>
<td>x_{33}</td>
<td>x_{34}</td>
<td>x_{35}</td>
<td>x_{36}</td>
<td>x_{37}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>x_{51}</th>
<th>x_{52}</th>
<th>x_{53}</th>
<th>x_{54}</th>
<th>x_{55}</th>
<th>x_{56}</th>
<th>x_{57}</th>
<th>x_{57}</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_{61}</td>
<td>x_{62}</td>
<td>x_{63}</td>
<td>x_{64}</td>
<td>x_{65}</td>
<td>x_{66}</td>
<td>x_{67}</td>
<td>x_{67}</td>
</tr>
<tr>
<td>x_{71}</td>
<td>x_{72}</td>
<td>x_{73}</td>
<td>x_{74}</td>
<td>x_{75}</td>
<td>x_{76}</td>
<td>x_{77}</td>
<td>x_{77}</td>
</tr>
</tbody>
</table>

### Filter:

<table>
<thead>
<tr>
<th>k_{11}</th>
<th>k_{12}</th>
<th>k_{13}</th>
</tr>
</thead>
<tbody>
<tr>
<td>k_{21}</td>
<td>k_{22}</td>
<td>k_{23}</td>
</tr>
<tr>
<td>k_{31}</td>
<td>k_{32}</td>
<td>k_{33}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>y_{11}</th>
<th>y_{12}</th>
<th>y_{13}</th>
<th>y_{14}</th>
<th>y_{15}</th>
<th>y_{16}</th>
<th>y_{17}</th>
</tr>
</thead>
<tbody>
<tr>
<td>y_{21}</td>
<td>y_{22}</td>
<td>y_{23}</td>
<td>y_{24}</td>
<td>y_{25}</td>
<td>y_{26}</td>
<td>y_{27}</td>
</tr>
<tr>
<td>y_{31}</td>
<td>y_{32}</td>
<td>y_{33}</td>
<td>y_{34}</td>
<td>y_{35}</td>
<td>y_{36}</td>
<td>y_{37}</td>
</tr>
<tr>
<td>y_{51}</td>
<td>y_{52}</td>
<td>y_{53}</td>
<td>y_{54}</td>
<td>y_{55}</td>
<td>y_{56}</td>
<td>y_{57}</td>
</tr>
<tr>
<td>y_{61}</td>
<td>y_{62}</td>
<td>y_{63}</td>
<td>y_{64}</td>
<td>y_{65}</td>
<td>y_{66}</td>
<td>y_{67}</td>
</tr>
<tr>
<td>y_{71}</td>
<td>y_{72}</td>
<td>y_{73}</td>
<td>y_{74}</td>
<td>y_{75}</td>
<td>y_{76}</td>
<td>y_{77}</td>
</tr>
</tbody>
</table>
Convolution Layer

- **Image**: \(n \times n \times 3\)
- **Kernel 1**: \(k \times k \times 3\)
- **Output**: \(n \times n\)
Convolution Layer

image
\( n \times n \times 3 \)

output
\( n \times n \)

kernel 1
\( k \times k \times 3 \)

kernel 2
\( k \times k \times 3 \)

kernel 3
\( k \times k \times 3 \)

kernel 4
\( k \times k \times 3 \)
Convolution Layer

input $n \times n \times 3$  \rightarrow  activations $n \times n \times f$
Max Pooling

\[ m = \max(X_1, X_2, X_3, X_4) \]
Convolutional Neural Network

28x28 input

28x28, 32 conv

14x14, 32 pool

14x14, 64 conv

14x14, 64 pool

7x7, 64 = 3136

fc 1024

fc 10
1. Representations sharing weights

2. Backpropagation by reverse differentiation, $O(n)$ speedup

3. Differentiable programming

Deep Learning Pros

- Image (CNN/ResNet/ODENet) Across Space
- Sequence (RNN/LSTM/GRU) Across Time
- Graph (GNN) Across Neighborhoods
CNN Training on CIFAR-10 Data

Source: cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html
Convolutional Neural Network (CNN)
ImageNet

CNN architecture

Source: ImageNet Classification with Deep Convolutional Neural Networks, Krizhevsky et al, NIPS 2012
ImageNet Filters

convolutional kernels of first layer

Source: ImageNet Classification with Deep Convolutional Neural Networks, Krizhevsky et al, NIPS 2012
ImageNet Results

Source: ImageNet Classification with Deep Convolutional Neural Networks, Krizhevsky et al, NIPS 2012
ImageNet Results

Source: ImageNet Classification with Deep Convolutional Neural Networks, Krizhevsky et al, NIPS 2012
Which training image patches do specific activation units in layer 1 respond to?

Visualizing and understanding convolutional networks, Zeiler and Fergus, ECCV 2014.
Understanding ImageNet

Which training image patches do specific activation units in layer 2 respond to?

Visualizing and understanding convolutional networks, Zeiler and Fergus, ECCV 2014.
Which training image patches do specific activation units in layer 3 respond to?

Visualizing and understanding convolutional networks, Zeiler and Fergus, ECCV 2014.
Which training image patches do specific activation units in layer 4 respond to?

Visualizing and understanding convolutional networks, Zeiler and Fergus, ECCV 2014.
Which training image patches do specific activation units in layer 5 respond to?

Visualizing and understanding convolutional networks, Zeiler and Fergus, ECCV 2014.
Input Maximizing Activation

\[
\arg\max_x a^l_i(W, x)
\]

given trained network with weights \(W\)
find input \(x\) which maximizes activation of unit \(i\) at layer \(l\)
starting from \(x\) as random noise perform gradient ascent on \(x\)

given trained network with weights $W$
find input $x$ which maximizes activation
starting from $x$ as random noise perform gradient ascent on $x$
Input Maximizing Different Objectives

\[ \text{given trained network with weights } W \]

\[ \text{find input } x \text{ which maximizes different objectives} \]

\[ \text{starting from } x \text{ as random noise perform gradient ascent on } x \]

Feature Visualization, Olah et al, Distill, 2017.
Training Patches vs. Optimization

Feature Visualization, Olah et al, Distill, 2017.
Maximization and Minimization

negative optimized  maximum negative patches  slightly negative patches  slightly positive patches  maximum positive patches  positive optimized

Feature Visualization, Olah et al, Distill, 2017.
Interactions Between Activations

Joint optimization
linear interpolation between objectives

Feature Visualization, Olah et al, Distill, 2017.
Visualizing Every Network Activation

Feature Visualization, Olah et al, Distill, 2017
https://distill.pub/2017/feature-visualization/appendix
Transfer Learning

- Task 1: learn to recognize animals given many (10M) examples which are not horses
- Keep layers from task 1, re-train on last layer
- Task 2: learn to recognize horses given a few (100) examples
Network Depth

What is the effect of depth on performance?
CNN Applications

- Image classification
- Object detection
- Image segmentation
- Image compression
- Image denoising
- Pose estimation
- Image synthesis and style transfer
- Image completion
- Reinforcement learning
CNN’s for Object Detection

Source: Ren et al, 2015
CNN’s for Image Segmentation

Source: Long et al, 2015
CNN’s for Image Segmentation

Source: He et al, 2017
CNN’s for Pose Estimation

Source: He et al, 2017

Source: Guler et al, 2018

https://www.youtube.com/watch?v=Dhkd_bAwwMc
CNN’s for Style Transfer

Source: Huang et al, 2017  https://www.youtube.com/watch?v=BcflKNzO31A
CNN’s for Image Synthesis

Source: Wang et al, 2018  https://www.youtube.com/watch?v=S1OwOd-war8
CNN’s for Image Completion

CNN’s for Chess and Go

**ResNet architecture:** for volume representing possible Chess moves
CNN’s for Face Recognition

Problem: single example for each person.
Solution: learn similarity rather than identity.

Reduce to verification: are $x_i$ and $x_j$ the same person?

Encode $x$ as $f(x)$ using CNN

Compare $f(x_i)$ with $f(x_j)$ by $d(f(x_i), f(x_j))$
CNN’s for Face Recognition

Train on input pairs \((x_i, x_j)\)

Label each pair \(y=1\) if \(x_i\) and \(x_j\) are same person, \(y=0\) otherwise

Use CNN encoding of pair \(f(x_i), f(x_j)\)

Loss function \(\mathcal{L}(x_i, x_j) = \mathcal{L}(y, \hat{y})\) \(\hat{y} = g\left(d\left(f(x_i), f(x_j)\right)\right)\)

Taigman et al, 2014
CNN’s for Medical Imaging

- New models
- Data augmentation
- Ensemble of models
- Known posterior distribution

CNN’s for X-Ray Interpretation

Input: 224,316 chest radiographs of 65,240 patients, labeler automatically detects 14 observations in radiology reports with uncertainties inherent in radiograph interpretation

Output: probability of 14 observations given available frontal and lateral radiographs

Features: Gender, Age, Frontal/Lateral, AP/PA, No Finding, Enlarged Cardiomegaly, Cardiomegaly, Lung Opacity, Lung Lesion, Edema, Consolidation, Pneumonia, Atelectasis, Pneumothorax, Pleural Effusion, Pleural Other, Fracture, Support Devices

CNN architectures: comparison of ResNet with DenseNet.

Source: Irvin et al, CheXpert dataset, 2019.
CNN’s for Anatomical Landmark Detection

3D CNN’s in deep reinforcement learning

Source: Alansary et al, MIDL 2018.
Assessing the Ability of Convolutional Neural Networks to Detect Glaucoma from OCT Probability Maps

Thakoor, Kaveri A.1; Zheng, Qian2; Nan, Linyong3; Li, Xinhui1; Tsamis, Emmanouil1; Rajeshkar, Rashmi1; Dweidi, Isht4; Dori, Iddo5; Sadja, Paul3,4,6; Hood, Donald C.5,6

Columbia University Department of 1Biomedical Engineering, 2Computer Science, 3Electrical Engineering, 4Radiology, 5Psychology, 6Ophthalmology, New York, NY

INTRODUCTION & PURPOSE

- Glaucoma is among the leading causes of irreversible blindness in the world
- Over 50% of glaucoma cases go undiagnosed due to lack of access to new specialized ophthalmology for timely screening

- We have developed a multi-Convolutional Neural Network (CNN) architecture to:
  - Automate detection of glaucoma from retinal nerve fiber layer (RNFL) probability maps derived from OCT cube/volume scan
  - Evaluate eyes of glaucoma patients (G), controls (NG-H), and healthy controls (NG-4)
- Results are shown in model performance (left), right)

METHODS

- The report above was generated for 322 eyes of 162 patients and 415 eyes of 415 healthy controls (NG-4) from wide-field OCT cube scans (Topcon).
- Patients were early glaucoma or glaucoma suspects (mean deviation on 24-2 visual field better than -6 dB).
- Senior author (OG) rated each patient eye on a scale between 0 (non-glaucomatous, NG-4) and 100 (glaucomatous, 6.3-7.05) using report above.
- The RNFL probability maps (red rectangle) supplied the input for all ANN models.
- The 192 x 6 and 345 x 64 eyes (153 N-G, 70 N-G, and 130 G) were divided into:
  - Training images: 395 (235 N-G, 70 N-G, and 130 G)
  - Validation images: 245 (100 N-G, 45 N-G, and 50 G)
  - Testing images: 99 (100 N-G, 41 N-G, and 56 G)
- Automated glaucoma detection was conducted with two CNN model types:
  - CNN-A-Type: without any natural image pretraining (i.e., trained only on OCT data), followed by downstream classifiers (Random Forest or Decision Trees)
  - CNN-B-Type: pretrained on ImageNet, followed by a non-parametric Random Forest classifier

RESULTS: ACCURACY RATES, ROC CURVES, POST-HOC ANALYSIS

- All models exhibited high accuracy performance and high AUC scores (see ROC curve, above left)
- CNN-A had highest accuracy with 4 false positives (FP) and 0 false negatives (FN)
- Correlation between human expert rating probabilities and model probabilities was high, R value of 0.87 (see scatterplot, below right)

CONCLUSIONS & FUTURE DIRECTIONS

- We have developed a multi-Convolutional Neural Network (Type B) CNN architecture that achieved high accuracy and high AUC score detection of glaucoma from OCT probability maps. Post-hoc analysis of false positives and false negatives, aided by Grad-CAM visualization, showed a strong correlation between human expert vs. machine performance. This work is a step towards enabling automatic eye disease detection in situations where access to vision experts may not be possible. FP and FN may be reduced with multimodal input data.
Thank you