Incremental and Approximate Inference for Faster Occlusion-based Deep CNN Explanations

Supun Nakandala, Arun Kumar, and Yannis Papakonstantinou
University of California, San Diego
Introduction

Deep Convolutional Neural Networks (CNNs) are revolutionizing many image analytics tasks.

Surveillance

Autonomous Vehicles

FDA Approves Marketing for First AI Device for Diabetic Retinopathy Detection
Background: What is a CNN?

*Simplified representation of a CNN

- **Input Image**
  - 3-D array

- **Series of Convolution Layer Transformations**

- **Predict Class Probability**
  - $P$(pneumonia)

- **Convolution Layer**

- **Input 3-D Array**

- **Output 3-D Array**

*Simplified representation of a CNN*
Explainability of CNN predictions is important in many critical applications such as in healthcare!
How to Explain CNN Predictions?

An active research area

**Occlusion-based explanation (OBE)** is widely used by practitioners
Occlusion-based Explanations (OBE)

Original Image

Occluded Image

P(pneumonia)

Occlusion heatmap localizes the region of interest

Source: http://blog.qure.ai/notes/visualizing_deep_learning
Problem: OBE is Highly Time Consuming

CNN Inference is time consuming.

E.g. Inception3: 35 MFLOPS, ResNet152 : 65 MFLOPS

Can take between several seconds to several minutes!

Our Idea
Cast OBE as a query optimization task
Database-inspired optimization techniques

*MFLOPS: Mega Floating Point Operations
Outline

1. Background on CNN Internals

2. Incremental CNN Inference
   Inspired by MQO + IVM

3. Approximate CNN Inference
   Inspired by AQP + Vision Science

4. Experimental Results

*IVM: Incremental View Maintenance
*MQO: Multi-Query Optimization
*AQP: Approximate Query Processing
Background: What is a CNN?

*Simplified representation of a CNN

- Input Image 3-D array
- Series of Convolution Layer Transformations
- Predict Class Probability
  - $P($pneumonia$)$

*Convolution Layer*

Input 3-D Array → Output 3-D Array

*Simplified representation of a CNN*
Background: Convolution Layer

Input 3-D Array

3-D Filter Kernel (learned)

$\sum(K1 \circ X')$

2-D slice of Output

Convolution Layer

$X$

$K1$

$K2$

$Kn$

$\circ$: Hadamard product
Reimagining Convolution as a Query

Input: A

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Filter Kernels: K

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>N</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FROM K, (SELECT A.X, A.Y, A.Z, A.V,
A.X - T.X + FW /2 AS A_K.X,
A.Y - T.Y + FH /2 AS A_K.Y
FROM A, A AS T
WHERE ABS(A.X - T.X) <= FW /2
AND ABS(A.Y - T.Y) <= FH /2)
WHERE A_K.X = K.X AND A_K.Y = K.Y
GROUP BY A.X, A.Y, K.N

Takeaway: CNN performs series of Joins and Aggregates
Linear Algebra data model improves hardware utilization

*FW: Filter Width, FH: Filter Height
Outline

1. Background on CNN Internals

2. Incremental CNN Inference
   Inspired by MQO + IVM

3. Approximate CNN Inference
   Inspired by AQP + Vision Science

4. Experimental Results

*IVM: Incremental View Maintenance
*MWO: Multi-Query Optimization
*AQP: Approximate Query Processing
Observation: Redundant Computations

This is a new instance of the Incremental View Maintenance task in databases.

Geometric properties of CNN determines how to propagate changes

* Only a cross section is shown. Changed region spans the depth dimension
Our Solution: Incremental Inference

Cast OBE as a set of sequence of “queries”

Original Image

Materialized Views

\textbf{Algebraic Framework} for Incremental Propagation: No redundant computations

Multiple occluded images. Sequential execution throttles the performance, especially on GPUs!
Our Solution: Batched Incremental Inference

Share and reuse materialized views across all occluded images

Multiple IVM queries run in one go (form of MQO). We create a custom GPU kernel for parallel memory copies. Improves hardware utilization.

additional context
Theoretical speedups for popular deep CNNs with our IVM

Issue: “Avalanche Effect” causes low speedups in some CNNs
Outline

1. Background on CNN Internals
2. Incremental CNN Inference
3. Approximate CNN Inference
   Inspired by AQP + Vision Science
4. Experimental Results
Approximate CNN Inference

Basic Idea: Trade off visual quality of heatmap to reduce runtime

How do we quantify new heatmap quality?

Structural similarity index (SSIM)

- Exact heatmap
- Approximate heatmap

SSIM values close to 0.9 are widely used

1.0 : identical

-1.0
Overview of our Approximations

a. Projective Field Thresholding
   Combats Avalanche Effect by pruning computations
   Makes every query in OBE faster

b. Adaptive Drill-down
   Lower granularity queries for less sensitive regions
   Reduces total number of queries in OBE
Avalanche Effect

Example: 1-D Convolution

Gains from Incremental Inference diminishes at latter layers!

Filter Kernel

Projective Field
Our Solution: Projective Field Thresholding

Number of different paths:

Projective Field Threshold

\[ \tau = \frac{5}{9} \]
How do we pick $\tau$?

Auto tune $\tau$ for SSIM target using a sample image set
Done once upfront during system configuration
Outline

1. Background on CNN Internals
2. Incremental CNN Inference
3. Approximate CNN Inference
   a. Projective Field Thresholding
   b. Adaptive Drill-down
4. Experimental Results
# Workload

<table>
<thead>
<tr>
<th>Images</th>
<th>Chest X-Ray Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td>Predicting Pneumonia</td>
</tr>
<tr>
<td>CNNs</td>
<td>VGG16, ResNet18, Inception3</td>
</tr>
<tr>
<td>Occluding Patch Color</td>
<td>Black</td>
</tr>
<tr>
<td>Occluding Patch Size</td>
<td>16 x 16</td>
</tr>
<tr>
<td>Occluding Patch Stride</td>
<td>4</td>
</tr>
<tr>
<td>SSIM Target</td>
<td>0.90</td>
</tr>
</tbody>
</table>

More datasets in the paper
# Experimental Setup

<table>
<thead>
<tr>
<th>CPU</th>
<th>Intel i7 @ 3.4 GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU</td>
<td>One Nvidia Titan Xp</td>
</tr>
<tr>
<td>Memory</td>
<td>32 GB</td>
</tr>
<tr>
<td>Deep Learning Toolkit</td>
<td>PyTorch version 0.4.0</td>
</tr>
</tbody>
</table>
Explaining CNN predictions is important. OBE is widely used.

DB-inspired incremental and approximate inference optimizations to accelerate OBE.

Our optimizations make OBE more amenable to interactive diagnosis of CNN predictions.

Project Web Page: https://adalabucsd.github.io/krypton.html
Video: https://tinyurl.com/y2oy9hqq
snakanda@eng.ucsd.edu
0: Invoke incremental inference.
1: Initialize the input tensors, kernel weights and output buffer in the GPU memory.
2: Invoke the Custom Kernel Interface (written in C) using Python foreign function interface (FFI) support. Pass memory references of input tensors, kernel weights and output buffer.
3: Forward the call to the Custom Kernel Implementation (written in CUDA).
4: Parallely copy the memory regions from the input tensor to an intermediate memory buffer.
5: Invoke the CNN transformation using cuDNN.
6: cuDNN reads the input from intermediate buffer and writes the transformed output to the output buffer.
7: Read the output to the main memory or pass reference as the input to the next transformation.
Read Context

Filter kernel

Input patch that needs to be read into the transformation operator

Updated patch in the input

Updated patch in the output

Input

Padding

Output

$P_{x,l}$

$P_{y,l}$

$W^R_{p,l}$

$W^l_{p,l}$

$H^R_{p,l}$

$H^l_{p,l}$

$S_{x,l}$

$S_{y,l}$

$W^R_{o,l}$

$W^l_{o,l}$

$y^R_{p,l}$

$y^l_{p,l}$

$x^R_{p,l}$

$x^l_{p,l}$
Incremental and Approximate Inference for Faster Occlusion-Based Deep CNN Explanations

Supun Nakandala, Arun Kumar, and Yannis Papakonstantinou
University of California, San Diego

Observation
Explainability of CNN predictions is important. Occlusion-based explainability (OBE) is widely used.

Problem
OBE is highly compute intensive.

This Work
Cast OBE as an instance of view-materialization problem.
Perform incremental and approximate inference.
~5x and ~35x speedups for exact and approximate heatmaps.