DBMS support for deep learning over image data

Parmita Mehta, Magdalena Balazinska, Andrew Connolly, and Ariel Rokem
University of Washington
Modern Data Management Requirements

- Manage image and video data
- Build complex machine learning models

**Astronomy:**
1. Data cleaning
2. Object extraction
3. Classification

**Ophthalmology:**
1. Classification
2. Segmentation
3. Clustering

**Neuroscience:**
1. Image processing
2. Denoising
3. Model fitting

**Consumer data:**
1. Object detection
2. Classification
3. Description
Use case: Optical coherence tomography (OCT)

OCT uses light waves to take cross-section pictures of retina to diagnose:

- macular hole, pucker, and edema
- age-related macular degeneration
- central serous retinopathy
- diabetic retinopathy

We got some good results

Model Building is a Messy Process

1. Different versions of the data with different metadata
2. Choose data and prepare it (e.g., crop it)
3. Build a model, train it, and evaluate it on development subset of data
4. Try to figure out why results terrible
5. Clean data, re-organize data, enhance data
6. Think of a new model and go back to step 3
7. Now compare the various models
8. Keep track of data subsets, models, model parameters, etc.
9. Maybe one day finally write the paper
10. And then when revision request comes back, try to remember all above
Key Challenges

- Large data volumes
- Slowness of lifecycle: train/test/change
- Cognitive burden of keeping track of data and models
- Correctness - don’t use test set to tune the model

Not seeking to replace ML libraries! But extend them with data management capabilities
Our Approach: ODIN DB
ODIN Architecture

Extend RDBMS with constructs to easily express tasks associated with model building and debugging.

Not seeking to replace ML libraries! But extend them with data management capabilities.

API: DSL

Python

SQL

... 

Query Optimizer

Physical Tuner

Parallel Execution

Relational Engine

Query

Optimizer

Physical

Tuner

Parallel

Execution

Relational

Engine
VDMS is a new system from Intel, designed specifically to store and query image databases.
# Our Data Model and Domain Specific Language

<table>
<thead>
<tr>
<th>Images</th>
<th>Models</th>
<th>Experiments</th>
<th>Per Image Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Image ID</td>
<td>- Model ID</td>
<td>- Experiment ID</td>
<td>- Experiment ID</td>
</tr>
<tr>
<td>- Image (as blob)</td>
<td>- Name</td>
<td>- Model ID</td>
<td>- Image ID</td>
</tr>
<tr>
<td>- Label</td>
<td>- Definition (JSON)</td>
<td>- Data Sets (test set, training set etc)</td>
<td>- Image ID</td>
</tr>
<tr>
<td>- Meta-data (e.g. age, patientID etc.)</td>
<td>- Meta-data(e.g. # of classes, type etc.)</td>
<td>- Results (accuracy, F1, recall etc)</td>
<td>- Activation for all neurons</td>
</tr>
<tr>
<td>- Insert / Delete / Update</td>
<td>- Insert / Delete / Update</td>
<td>- Insert / Delete / Update</td>
<td>- Generate Attribution for Image ID(s)</td>
</tr>
<tr>
<td>- Select (e.g. create training set)</td>
<td>- Select</td>
<td>- Select</td>
<td>- Select</td>
</tr>
<tr>
<td>- Crop, Rotate, Blur, Resize ...</td>
<td>- Generate Maximized Image</td>
<td>- Generate Maximized Image</td>
<td>- Generate Attribution for Image ID(s)</td>
</tr>
</tbody>
</table>
### Example Database
**Images:** OCT_Images

<table>
<thead>
<tr>
<th>Image-ID</th>
<th>Label</th>
<th>Slice-ID</th>
<th>Patient-ID</th>
<th>Age</th>
<th>G</th>
<th>Visual Acuity</th>
<th>Diag</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>b06e7bfc444c93db26a7c6a5d4d234-00033918-026.png</td>
<td>ERM</td>
<td>26</td>
<td>b06e7bfc444c93db26a7c6a5d4d234</td>
<td>52.28</td>
<td>1</td>
<td>0.48</td>
<td>[1, 0, 0, 0]</td>
<td><img src="image1.png" alt="Image" /></td>
</tr>
<tr>
<td>6cc38578fc7f24f21519d14f776d4c-00168131-029.png</td>
<td>AMD</td>
<td>29</td>
<td>6cc38578fc7f24f21519d14f776d4c</td>
<td>90.05</td>
<td>1</td>
<td>0.7</td>
<td>[0, 1, 0, 0]</td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
</tbody>
</table>
## Example Database

Models: OCT_Models

<table>
<thead>
<tr>
<th>Model-ID</th>
<th>Name</th>
<th>Definition</th>
<th>Classes</th>
<th>Type</th>
<th>Input</th>
<th>Number of Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VGG-16-BN</td>
<td>JSON</td>
<td>4</td>
<td>Multi-class</td>
<td>(256,256)</td>
<td>134,276,034</td>
</tr>
<tr>
<td>2</td>
<td>Inception-V3</td>
<td>JSON</td>
<td>4</td>
<td>Multi-label</td>
<td>(299,299)</td>
<td>24,348,324</td>
</tr>
<tr>
<td>Experiment-ID</td>
<td>Model-ID</td>
<td>Train</td>
<td>Test</td>
<td>Acc</td>
<td>Epochs</td>
<td>Initial-LR</td>
</tr>
<tr>
<td>--------------</td>
<td>----------</td>
<td>-------------</td>
<td>-------------</td>
<td>--------</td>
<td>--------</td>
<td>------------</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>retina-train2</td>
<td>retina-test2</td>
<td>78.8</td>
<td>50</td>
<td>1e-3</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>retina-train2</td>
<td>retina-test2</td>
<td>90.05</td>
<td>150</td>
<td>1e-4</td>
</tr>
</tbody>
</table>
## Example Database
Per Image Parameters: OCT_LIP

- Experiment ID
- Image ID
- Activation for all neurons
- Predicted class

<table>
<thead>
<tr>
<th>Experiment-ID</th>
<th>Image-ID</th>
<th>Activation</th>
<th>Predicted class</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>b06e7bfc444c93db26a7c6a5d4d234-00033918-026.png</td>
<td>JSON</td>
<td>2</td>
</tr>
<tr>
<td>25</td>
<td>6cc38578fc7f24f21519d14f776d4c-00168131-029.png</td>
<td>JSON</td>
<td>3</td>
</tr>
</tbody>
</table>
1. Basic queries
   a. Select images/models/experiments based on metadata
   b. Execute user-defined code on any of the data (e.g., train model)
2. Model-debugging queries
   a. What is the model learning?
   b. What are representative images that classifier gets wrong?
3. Model comparison queries
   a. Why is this model better? What are the models learning differently?
4. Data inspection queries
   a. What are the important features in my data?
Research Questions

1. Materialization vs Re-processing:
   a. Storing intermediates requires tens to hundreds of GB of storage
   b. Re-running model for each diagnostic query is slow
   c. What are the trade-offs for materialization vs regeneration?
   d. How best to compress the materialized data?

2. Expressivity:
   a. How best to extend relational model to express queries easily?

3. Extensibility:
   a. This is an active research area, how to build extensibility into the system to allow new operations and classes of machine learning?
Conclusion

- Images and videos are common data types today
- Workloads primarily focus on machine learning / deep learning
- Database management systems provide limited to no support
- ODIN DB is a new DBMS that extends relational systems with