Using BigDL to Build Image Similarity-Based House Recommendations

By Jason Dai (Intel), Yang, Yuhao (Intel), Jiao (Jennie) Wang, published on November 22, 2017

Overview

This paper introduces an image-based house recommendation system that was built between MLSListings® and Intel® using BigDL on Microsoft Azure®. Using Intel's BigDL distributed deep learning framework, the recommendation system is designed to play a role in the home buying experience through efficient index and query operations among millions of house images. Users can select a listing photo and have the system recommend listings of similar visual characteristics that may be of interest. The following provides additional parameters to the image similarity search:

- Recommend houses based on title image characteristics and similarity. Most title images are front exterior, while others can be a representative image for the house.
• Low latency API for online querying (< 0.1s).

Background

MLSListings Inc., the premier Multiple Listing Service (MLS) for real estate listings in Northern California, is collaborating with Intel and Microsoft to integrate artificial intelligence (AI) into their authorized trading platform to better serve its customers. Together, the technologies enhance the home buying search process using visual images through an integration between Real Estate Standard Organization (RESO) APIs and Intel’s BigDL open source deep learning library for Apache Spark*. The project is paving the road for innovation in advanced analytics applications for the real estate industry.

A large number of problems in the computer vision domain can be solved by ranking images according to their similarity. For instance, e-retailers show customers products that are similar items from past purchases, to sell more online. Practically every industry sees this as a game changer, including the real estate industry, as it has become increasingly digital over the past decade. More than 90 percent of homebuyers search online in the process of seeking a property. Homeowners and real estate professionals provide information on house characteristics such as location, size, and age, as well as many interior and exterior photos for real estate listing searches. However, due to technical constraints, the enormous amount of information in the photos cannot be extracted and indexed to enhance search or serve real estate listing results. In fact, show me similar homes is a top wish list request among users. By tapping into the available reservoir of image data to power web plus mobile digital experiences, the opportunity to drive greater user satisfaction from improved search relevancy is now a reality.

Enter the Intel BigDL framework. As an emerging distributed deep learning framework, BigDL provides easy and integrated deep learning capabilities for big data communities. With a rich set of support for deep learning applications, BigDL allows developers to write their deep learning applications as standard Spark programs, which can directly run on top of existing Apache Spark or Apache Hadoop* clusters.

Overview of Image Similarity

In the research community, image similarity can mean either semantic similarity or visual similarity. Semantic similarity means that both images contain the same category of objects. For example, a ranch house and a traditional house are similar in terms of category (both houses), but may look completely different. Visual similarity, on the other hand, does not care about the object categories but measures how images look like each other from a visual perspective; for example, an apartment image and a traditional house image may be quite similar.

Semantic similarity:

![Semantic similarity image 1](image1)

![Semantic similarity image 2](image2)

Visual similarity:

![Visual similarity image 1](image3)

![Visual similarity image 2](image4)
For semantic similarity, usually it's an image classification problem, and can be efficiently resolved with
the popular image perception models like GoogLeNet\textsuperscript{3} or VGG\textsuperscript{4}.

For visual similarity, there have been many techniques applied across the history:

- **SIFT, SURF, color histogram\textsuperscript{5}**
  Conventional feature descriptors can be used to compare image similarity. SIFT feature descriptor
  is invariant to uniform scaling, orientation, and illumination changes, and makes it useful for
  applications like finding a small image within a larger image.

- **pHash\textsuperscript{6}**
  This mathematical algorithm analyzes an image's content and represents it using a 64-bit number
  fingerprint. Two images' pHash values are close to one another if the images' content features are
  similar.

- **Image embedding with convolutional neural networks (convnet)\textsuperscript{8}**
  Finding the image embedding from the convnet; usually it's the first linear layer after the
  convolution and pooling.

- **Siamese Network or Deep Ranking\textsuperscript{8}**
  A more thorough deep learning solution, but the result model depends heavily on the training data,
and may lose generality.

**Solution with BigDL**

To recommend houses based on image similarity, we first compare the query image of the selected listing photo with the title images of candidate houses. Next, a similarity score for each candidate house is generated. Only the top results are chosen based on ranking. By working with domain experts, the following measure for calculating image similarity for house images was developed.

For each image in the candidates, compare with query image {

- **class** score: Both house front? (Binary Classification)
- **tag** score: Compare important semantic tags. (Multinomial Classification)
- **visual** score: Visually similarity score, higher is better

\[
\text{final Score} = \frac{\text{class score (decisive)}}{1} + \frac{\text{tag score (significant)}}{0.3} + \frac{\text{visual score}}{[0,1]} \]

In this project, both semantic similarity and visual similarity were used. BigDL provides a rich set of functionalities to support training or inference image similarity models, including:

- Providing useful image readers and transformers based on Apache Spark and OpenCV* for parallel image preprocessing on Spark.
- Natively supporting the Spark ML* Estimator/Transformer interface, so that users can perform deep learning training and inference within the Spark ML pipeline.
- Providing convenient model fine-tuning support and a flexible programming interface for model adjustment.
- Users can load pretrained Caffe*, Torch* or TensorFlow* models into BigDL for fine-tuning or inference.
Semantic Similarity Model

For semantic similarity, three image classification models are required in the project.

Model 1. Image classification: Determines whether the house front is exterior. We need to distinguish if the title image is or is not the house front. The model is fine-tuned from pretrained GoogLeNet v1 on the Places* dataset (https://github.com/CSAILVision/places365). We used the Places dataset for the training.

Following is the code for the model training with the DLClassifier* in BigDL. We loaded the Caffe model pretrained from the Places dataset, in which the last two layers (linear (1024 -> 365 and Softmax) were removed from the Caffe model definition. Then, a new linear layer with classNum was added, to help train the classification model we required.

```scala
val caffeModel =
  Module.loadCaffeModel("/googlenet_places365.prototxt_modified",
  "./googlenet_places365.caffemodel").asInstanceOf[Graph[Float]]
val model = Sequential().add(caffeModel)
  .add(View(Array(1024))).add(Linear(1024, classNum)).add(LogSoftMax())

val dClassifier = new DLClassifier[Float](model,
  criterion = new ClassNLLCriterion[Float](),
  featureSize = Array(3, imageSize,imageSize))
  .setBatchSize(batchSize)
  .setLearningRate(learningRate)
  .setOptimMethod(new Adam[Float]())
  .setMaxEpoch(maxEpoch)
  .setFeaturesCol("features")
```

Model 2. Image classification: House style (contemporary, ranch, traditional, Spanish). Similar to 1, the model is fine-tuned from pretrained GoogLeNet v1 on the Places dataset. We sourced the training dataset from photos for which MLSListings have been assigned copyrights.

Model 3. Image classification: House story (single story, two story, three or more stories). Similar to 1, the model is fine-tuned from pretrained GoogLeNet v1 on the Places dataset. We sourced the training dataset from photos for which MLSListings have been assigned copyrights.
Visual Similarity Model

We need to compute visual similarity to derive a ranking score.

For each query, the user will input an image for comparison against the thousands of candidate images, returning the top 1000 result in 0.1 second. To meet the latency requirement, we performed a direct comparison against precalculated features from images.

We first built an evaluation dataset to choose the best options for image similarity computation. In the evaluation dataset, each record contains three images.

*Triplet (query image, positive image, negative image)*, where positive image is more similar to the query image. For each record, we can evaluate different similarity functions.

```python
if (similarity(query image, positive image) > similarity(query image, negative image))
    correct += 1
else
    incorrect += 1
```

In the four methods listed above for computing image similarity, Siamese Network or Deep Ranking appear to be more precise, but due to the lack of training data to support meaningful models the results were inconclusive. With the help of the evaluation dataset we tried the remaining three methods, and both SIFT and pHash produced unreasonable results. We suspect that was because both of them cannot represent the essential characteristics of real estate images.

Using image embedding from the pretrained deep learning models on the Places dataset, the expected precision accuracy level was achieved:

<table>
<thead>
<tr>
<th>Network</th>
<th>Feature</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deepbit*</td>
<td>1024 binary output</td>
<td>80%</td>
</tr>
<tr>
<td>Model</td>
<td>Flops</td>
<td>Accuracy</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>GoogLeNet*</td>
<td>1024</td>
<td>84%</td>
</tr>
<tr>
<td>VGG-16</td>
<td>25088</td>
<td>93%</td>
</tr>
</tbody>
</table>

Similarity \( (m_1, m_2) = \cos (\text{embedding} (m_1), \text{embedding} (m_2)) \).

After L2 normalization, cosine similarity can be computed very efficiently. While VGG-16 embedding has a clear advantage, we also tried the SVM model trained from the evaluation dataset to assign different weight to each of the embedding features, but this only gives limited improvement, and we are concerned that the SVM model may not be general enough to cover the real-world images.

**Image Similarity-Based House Recommendations**

The complete data flow and system architecture is displayed as follows:
In production, the project can be separated into three parts:

1. **Model training (offline)**
   The model training mainly refers to the semantic models (GoogLeNet v1 fine-tuned on the Place dataset) and also finding the proper embedding for visual similarity calculation. Retraining may happen periodically depending on model performance or requirement changes.

2. **Image inference (online)**
   With the trained semantic models (GoogLeNet v1) in the first step and the pretrained VGG-16, we can convert the images to tags and embeddings, and save the results in a key-value cache. (Apache HBase* or SQL* can also be used).
All the existing images and new images need to go through the inference above and converted into a table structure, as shown:
The inference process can happen periodically (for example, one day) or triggered by a new image upload from a real estate listing entry. Each production image only needs to go through the inference process once. With the indexed image tagging and similarity feature, fast query performance is supported in a high concurrency environment.

3. API serving for query (online)
   The house recommendation system exposes a service API to its upstream users. Each query sends a query image and candidate images as parameters. With the indexed image information shown in the table above, we can quickly finish the one-versus-many query. For cosine similarity, processing is very efficient and scalable.

**Demo**

We provided two examples from the online website:

**Example 1**
Example 2
Summary

This paper described how to build a house recommendation system based on image analysis utilizing Intel’s BigDL library on Microsoft Azure integrated to MLSListings through RESO APIs. Three deep learning classification models were trained and fine-tuned from pretrained Caffe models in order to extract the important semantic tags from real estate images. We further compared different visual similarity computation methods and found image embedding from VGG to be the most helpful inference model in our case. As an end-to-end industry example, we demonstrated how to leverage deep learning with BigDL to enable greater deep learning-based image recognition innovation for the real estate industry.

References

1. Intel-Analytics/BigDL, https://github.com/intel-analytics/BigDL.
For more complete information about compiler optimizations, see our Optimization Notice (/en-us/articles/optimization-notice#opt-en).