SLDC: an open-source workflow for object detection in multi-gigapixel images

Romain Mormont, Jean-Michel Begon, Renaud Hoyoux, Raphaël Marée

Systems and modeling, Department of EE & CS, University of Liège, Belgium

12th September 2016
Outline

1. Context
2. SLDC
   - Framework
   - How it works
   - Features
3. SLDC at work: thyroid nodule malignancy
   - Thyroid case
   - Cytomine
   - Data
   - Workflow
   - Results
4. Conclusion and future works
Context

Microscope slide smeared with thyroid cell samples (15 gigapixels).
Microscope slide smeared with core samples (11 gigapixels).
Microscope slide smeared with lung cell samples (3 gigapixels).
Context

- Huge slides usually analysed manually!
- Machine learning (ML) and image processing (IP) could be used to assist humans
- Problems of object detection and classification
SLDC: framework

*SLDC* is an **open-source Python framework** created for accelerating development of large image analysis workflows.

**How?**
- It encapsulates problem-independent logic (parallelism, memory limitation due to large images handling,...)
- It provides a concise way of declaring problem dependant components (segmentation, object classification,...)
SLDC: how it works
SLDC: features

- **Tile-based processing** to avoid loading a full image into memory
- Several level of **parallelism**: tiles, objects, images,...
- A **customizable logging system** providing a rich feedback about the execution
- **Effortless integration** with other Python libraries: scikit-learn (ML), open-cv (IP), PyCuda (GPU),...
SLDC at work: thyroid case

Aim: detect **cells with inclusion** and **proliferative architectural patterns**
SLDC at work: Cytomine

*cytomine* is a web-based environment enabling collaborative multi-gigapixel image analysis. (Website: [www.cytomine.be](http://www.cytomine.be). Marée & al., Bioinformatics; 2016).
SLDC at work: data

- **84 images** with size ranging from 4 to 18 gigapixels
- **68 annotated images**
- **5921 labelled annotations** made by cytopathologists

![Graphs]

(a) Annot. per group

(b) Annot. per term

---

Team of Pr. Isabelle Salmon, Department of Pathology, Faculty of Medicine, ULB
SLDC at work: data (cont’d)

(c) Pattern annot. per group

(d) Pattern annot. per term

(e) Proliferative (malignant)

(f) Normal patterns (benign)
SLDC at work: data (cont’d)

(g) Cell annot. per group

(h) Cell annot. per term

(i) Cells with incl. (malignant)

(j) Normal cells (benign)
SLDC at work: workflow

Slide segmentation → Dispatch classifier → Pattern classifier

- Pattern (0) → Normal (0)
- Proliferative (1)

- Cell (1) → Normal (0)
- Inclusion (1)

Other (2)
SLDC at work: workflow (cont’d)
Classification is performed based on the detected object’s crop image using random subwindows and extremely randomized trees\textsuperscript{2}.

**Cell with inclusion vs. normal cells:**

<table>
<thead>
<tr>
<th></th>
<th>Normal</th>
<th>Inclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>881</td>
<td>62</td>
</tr>
<tr>
<td>Inclusion</td>
<td>109</td>
<td>106</td>
</tr>
</tbody>
</table>

**Proliferative vs. normal patterns:**

<table>
<thead>
<tr>
<th></th>
<th>Normal</th>
<th>Prolif.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>158</td>
<td>55</td>
</tr>
<tr>
<td>Prolif.</td>
<td>15</td>
<td>281</td>
</tr>
</tbody>
</table>

\textsuperscript{2}Marée et al., Pattern Recognition Letters ; 2016
SLDC at work: results

Size: 131072 × 57856

Time (1st pass): 4 min 38 sec
Time (2nd pass): 2 min 04 sec

Objects found: 18882
Cells found: 17802
Patterns found: 1080
Jobs: 64
Max memory usage: 159.414 Go

Size: 163840 × 95744

Time (1st pass): 12 min 24 sec
Time (2nd pass): 7 min 10 sec

Objects found: 76133
Cells found: 69820
Patterns found: 6313
Jobs: 64
Max memory usage: 179.855 Go
Conclusion and future works

1. Framework
   - Production-ready!
   - Open-source and generic.
   - Still some minor improvements to make (parallelization, dispatching, ...)
   - **Feel free to use it:** https://github.com/waliens/sldc

2. Thyroid workflow:
   - At this point, too many false positives.
   - Need to improve the classifiers and the segmentation procedures
Thank you for your attention!
Any question?
SLDC: toy example

The aim is to detect circles in the following image. As a bonus, we want to know their center color.
# Defining a segmenter

class CustomSegementer(Segmenter):
    '''All non-black pixels are in an object of interest'''
    def segment(self, image):
        return (image > 0).astype(np.uint8)

# Defining a dispatching rule

class CircleRule(DispatchingRule):
    '''A rule which matches circle polygons'''
    def evaluate_batch(self, image, polygons):
        return [circularity(p) > 0.85 for p in polygons]

# Defining a polygon classifier

class ColorClassifier(PolygonClassifier):
    '''
    A classifier which returns the color (greyscale) of the center pixel of the object
    '''
    def predict_batch(self, image, polygons):
        classes = [center_pxl_color(image, p) for p in polygons]
        probas = [1.0] * len(polygons)
        return classes, probas
SLDC: toy example (cont’d)

```python
# Build the workflow
builder = WorkflowBuilder()
builder.set_n_jobs(100)
builder.set_segmenter(CustomSegmenter())
builder.add_classifier(CircleRule(), ColorClassifier(), disp_label="circle")
workflow = builder.get()

# Process an image
results = workflow.process(image)

# Go through the detected objects
for polygon, dispatch, label, proba in results:
    print "Detected polygon {}".format(polygon)
    print "Dispatched by '{}'".format(dispatch)
    print "Predicted class {}".format(label)
    print "Probability {}".format(proba)
    print ""
```
Detected polygon POLYGON (((...)))
Dispatched by 'circle'
Predicted class 128
Probability 1.0

Detected polygon POLYGON (((...)))
Dispatched by 'circle'
Predicted class 255
Probability 1.0
SLDC: scalability

(a) Evolution of the execution times when varying the number of available processors

(b) Execution times per giga-pixels.