1 Problem

Ilastik uses VIGRA convolution routines to compute features for the calssification process. For large data sets, those feature computations represent the computational bottleneck in the pipeline. As convolutions are one of the classical applications for GPUs, they seem to be the reasonable choice for speeding up those computations.

2 Current situation and idea

In the current setup, VIGRA is used via the python wrapper. This VIGRA version/method uses only a single-thread convolution implementation. We evaluate the performance of VIGRA gaussian filters in single-thread mode and multi-thread mode (Version 1.11) and Arrayfire gaussian filters.

3 Why ArrayFire?

ArrayFire (http://www.arrayfire.com) aims to be a flexible, hardware-neutral accelerator library. This means it is not only running on CUDA GPUs, but also includes an OpenCL backend (e.g. for AMD GPUs) and a CPU fallback. Due to the backends for CUDA and OpenCL, it is still able to leverage many specific features, that are essential to reach high performance on GPUs (see "Why not OpenGL").
Moreover, the library is under active development and it is to be expected that it will modified for new hardware in the future. Also the ArrayFire team aims to support users as good as possible, e.g with an active user group or personal consultation.

Why not own CUDA code?
Although CUDA implementations could improve performance by writing more hardware specific code, it would be prone to overengineering. Also such code requires enourmous effort to make it efficient and flexible and will be a lot harder to maintain.

Why not OpenCV?
OpenCV also provides GPU implementations. It also seems to achieve good performance and might be a useful alternative. Yet, we concentrated on ArrayFire.

However, there are benchmarks on ArrayFire vs OpenCV:
http://mcclanahoochie.com/blog/2011/09/opencv-vs-libjacket-gpu-sobel-filtering/

http://mcclanahoochie.com/blog/2011/10/gpu-convolution-opencv-gpu-and-libjacket-part-2/
http://opencv-gpu.blogspot.de/

Why not OpenGL?
OpenGL would provide a widely supported library. As a consequence, it does not exploit all available features of specific GPUs as it is possible with CUDA or OpenCL. For example, OpenGL usually relies on texture buffers and does not make use of shared memory which is essential for fast convolutions. Also an OpenGL implementation might result in more complicated code.

Why not SciPy?
First test indicated that SciPy only provides a small speedup and seems to use an single-thread CPU implementation.

4 Arrayfire vs VIGRA

4.1 Differences in border treatment (padding)

The convolution of ArrayFire can be configured to have the same result as the VIGRA convolution, exept for one difference. VIGRA uses reflected border padding by default, whereas ArrayFire uses zero padding at the borders. This means ArrayFire will add zeros to the image as soon as the kernel is not overlapping entirely with the image. Therefore results in border regions differ from default vigra results.
The results not affected by those differences have been tested to be equal.

4.2 Hardware used for testing

- notebook - Intel(R) Core(TM) i7-3632QM CPU @ 2.20GHz, 8 GB RAM, GeForce GT 635M (2GB), CUDA Compute 2.1
- littleheron - Intel(R) Xeon(R) CPU E5-2643 v3 @ 3.40GHz, 64 GB RAM, GeForce GTX TITAN X (12 GB) CUDA Compute 5.2

4.3 C++ tests for gaussian smoothing

4.3.1 Total GPU time

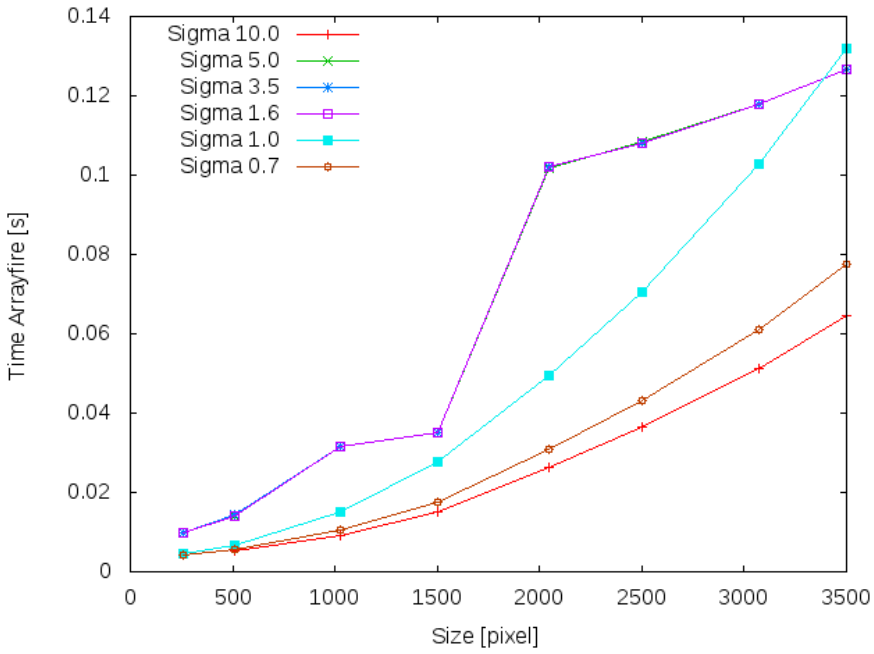

Figure 1: Total GPU times on notebook for different gaussian filter sizes and image sizes. Kernel size is calculated as follows: $r = \text{round}(3\sigma) \cdot 2 + 1$.

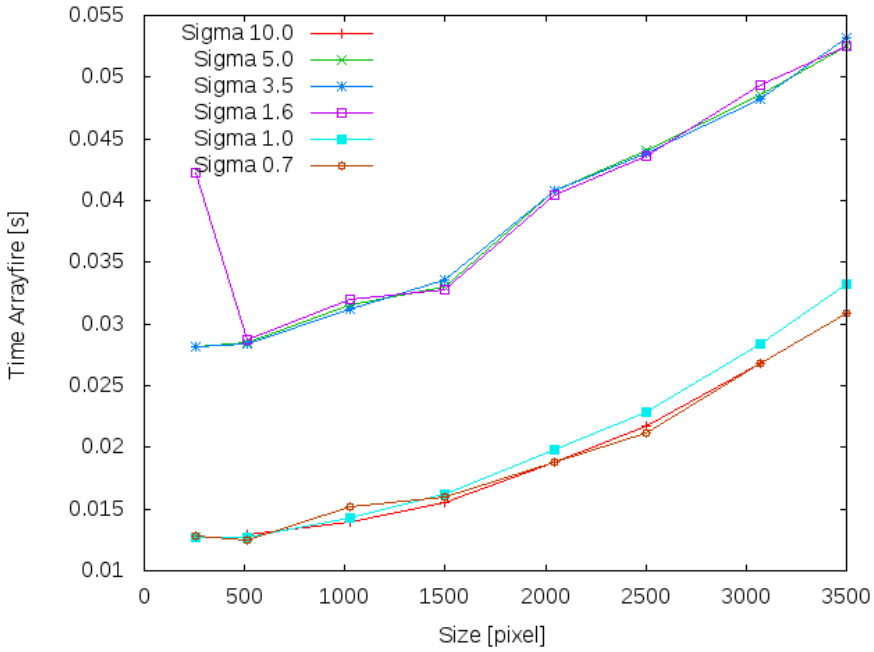

Figure 2: Total GPU times on littleheron for different gaussian filter sizes and image sizes. Kernel size is calculated as follows: $r = \text{round}(3\sigma) \cdot 2 + 1$.

4.3.2 Ratio VIGRA single-thread vs ArrayFire

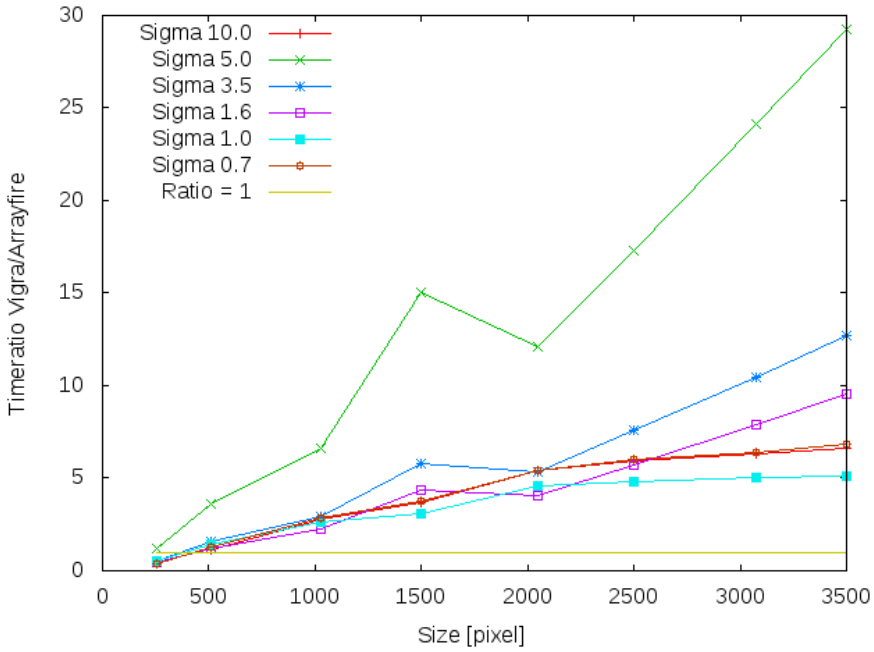

Figure 3: Ratio $\frac{\text{VIGRA time}}{\text{ArrayFire time}}$ on notebook for different gaussian filters and image sizes. We experience a speed up for ratios > 1.

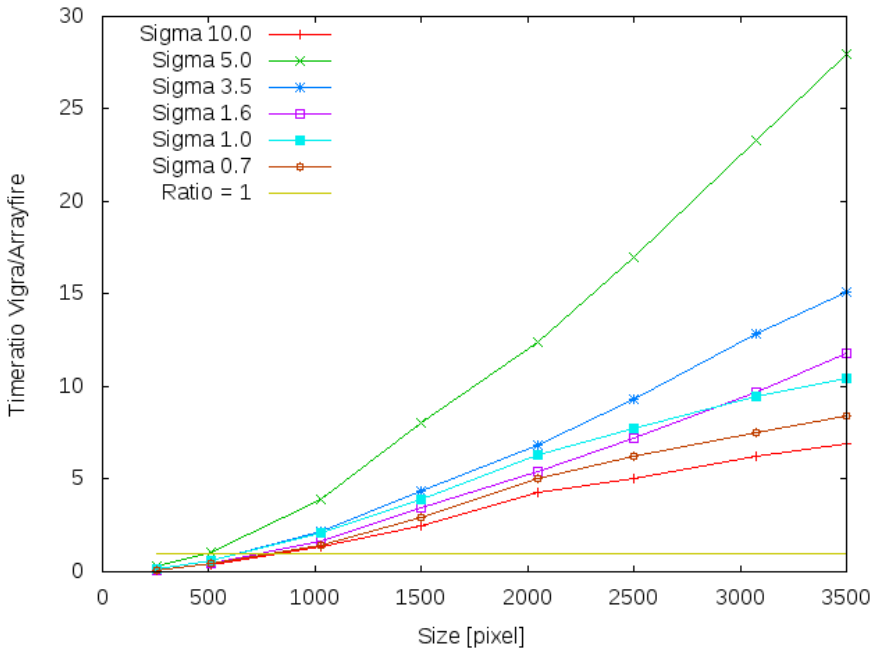

Figure 4: Ratio $\frac{\text{VIGRA time}}{\text{ArrayFire time}}$ on littleheron for different gaussian filters and image sizes. We experience a speed up for ratios > 1.

4.3.3 Ratio VIGRA multi-thread vs ArrayFire

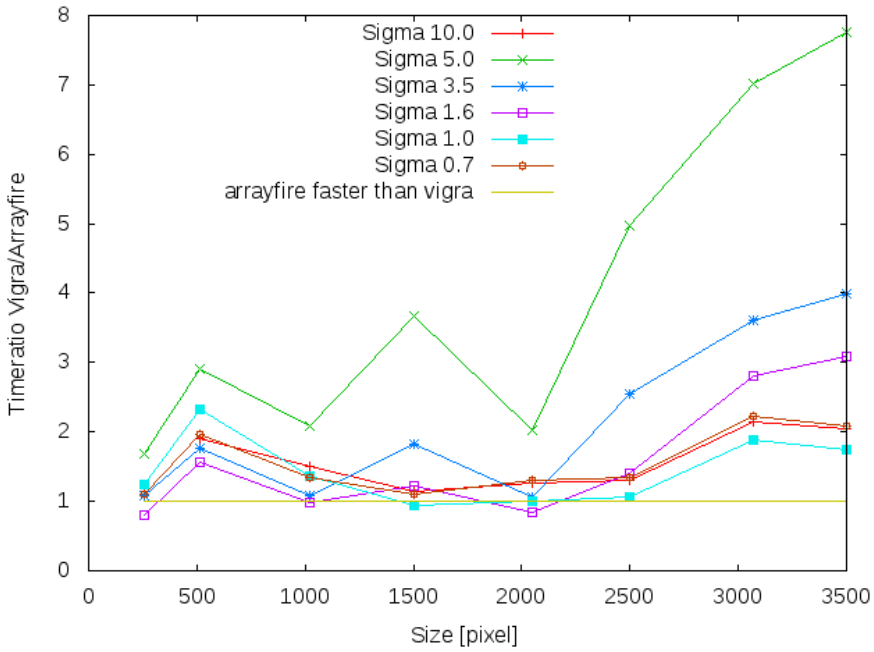

Figure 5: Ratio $\frac{\text{VIGRA time}}{\text{ArrayFire time}}$ on notebook for different gaussian filters and image sizes. We experience a speed up for ratios > 1.

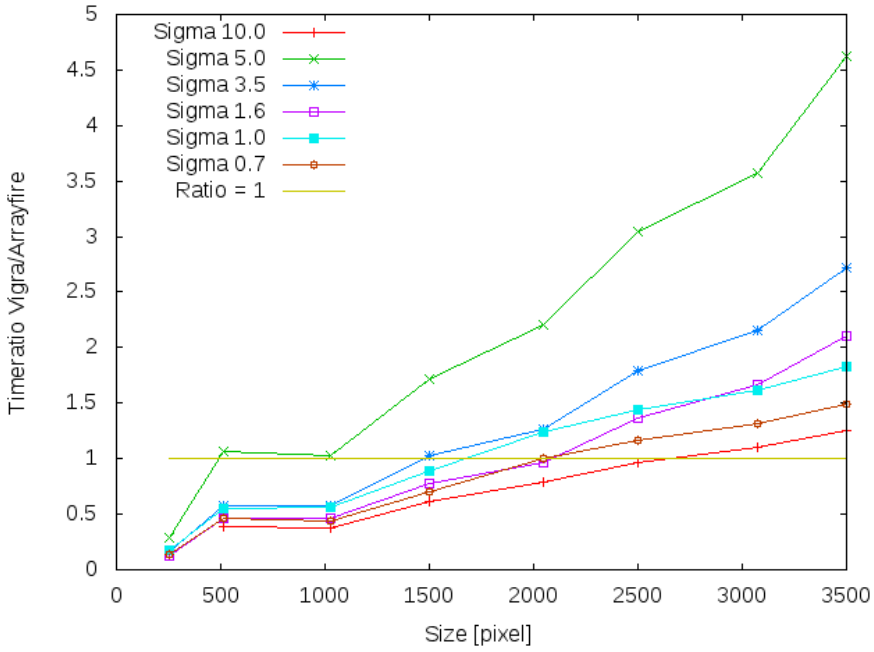

Figure 6: Ratio $\frac{\text{VIGRA time}}{\text{ArrayFire time}}$ on littleheron for different gaussian filters and image sizes. We experience a speed up for ratios > 1.