Optimization of neural computations using a functional data-parallel language

Or making neural networks performance-portable

AlexNet (Krizhevsky et al., 2012)

By Naums Mogers
Supervisor: Christophe Dubach
Neural networks (NNs) depend on hardware-specific low-level optimizations.

Manual approach:

- Requires expertise in both machine learning and performance programming
- **Costly** to develop and maintain
- **Hard to port** to new platforms

Automated approaches:

- *Caffe, Tensorflow, Theano, Torch* have **limited** functional and performance portability
- *Autotuners* are not performance-portable because of **no structural optimizations**
Lift, a functional data-parallel language
  - Abstracted from hardware, pure and safe

Lift code example:

```haskell
fully_connected(f, weights, bias, inputs) :=
  Map((neuron_weights, neuron_bias) → f()) o Reduce(add, neuron_bias) o
  Map(mult) $ Zip(inputs, neuron_weights)) $ Zip(weights, bias)
```
Solution

- **Lift**, a functional data-parallel language
  - Abstracted from hardware, pure and safe

- Introduce NN-specific primitives such as *conv, norm, pool, fully_connected*

- Implement fine-grained generic optimizations such as:
  - Parallel mappings space exploration
  - Memory tiling & coalescing
  - Float quantization
  - Neuron pruning
  - Training batch size autotuning
  - Varying precision across layers
  - Vectorization

- Optimize based on NN configuration, input dimensions and target hardware
- Generate OpenCL code for any OpenCL-supporting target hardware
Questions?