Neural Networks in OpenCL

Jakub Zárybnický

Implementation

- My own subgoal: result in Haskell
- CUDA
  - nice API, pleasant bindings but not emulable
  - lost access to Gent datacenter @Dec 10 due to renovations
- Raw OpenCL first
- OpenCL in Haskell (but then got uncomfortably close to the existing Grenade API)
- Parallelizing an existing framework (Grenade)
  - Adding an OpenCL backend

Parallel v. original gradient descent

```
kernel void descend_slow(
    double rate,
    double momentum,
    double regulariser,
    constant double *weights,
    constant double *gradient,
    constant double *last,
    global double *outputWeights,
    global double *outputMomentum
) {
  int i = get_global_id(0);
  double momentum_ = momentum * last[i] - rate * gradient[i];
  outputMomentum[i] = momentum_;
  outputWeights[i] = mad(weights[i], 1 - rate * regulariser, momentum_);
}
```

```
-:---  kernel.cl      Top (9,33)    Git-master  (Opencl/*l FlyC Projectile[gmu-project:haske
```

```
#include "gradient_descent.h"

void descend_cpu(int len, double rate, double momentum, double regulariser,
    const double* weights,
    const double* gradient,
    const double* last,
    double* outputWeights, double* outputMomentum) {

  for (int i = 0; i < len; i++) {
      outputMomentum[i] = momentum * last[i] - rate * gradient[i];
      outputWeights[i] = weights[i] + outputMomentum[i] - (rate * regulariser) * weights[i];
  }
}
```

FFI API replacements

```
clEnqueueWriteBuffer' cq mem check off size dat =
  alloca $ \event → do
    _ ← raw_clEnqueueWriteBuffer cq mem (fromBool check) (fromIntegral off) (fromIn
    raw_clWaitForEvents 1 (castPtr event)

clEnqueueNDRangeKernel' :: CLCommandQueue → CLKernel → [CSize] → IO Int32
clEnqueueNDRangeKernel' cq krn gws =
  withArray gws $ \pgws →
  alloca $ \event →
    whenSuccess
      (raw_clEnqueueNDRangeKernel cq krn num nullPtr pgws nullPtr 0 nullPtr event)
      (raw_clWaitForEvents 1 (castPtr event))
  where
    num = fromIntegral $ length gws

clSetKernelArgSto' :: Storable a ⇒ CLKernel → Word32 → a → IO ()
clSetKernelArgSto' krn idx val = with val $ \pval → do
  () <$ raw_clSetKernelArg krn idx (fromIntegral . sizeOf $ val) (castPtr pval)

foreign import ccall "clEnqueueNDRangeKernel" raw_clEnqueueNDRangeKernel
  :: CLCommandQueue
  → CLKernel
  → Word32
  → Ptr CSize
  → Ptr CSize
  → Ptr CSize
  → Word32
  → Ptr CLEvent
  → Ptr CLEvent
  → IO Int32

foreign import ccall "clEnqueueReadBuffer" raw_clEnqueueReadBuffer
  :: CLCommandQueue
  → CLMem
  → CBool
  → CSize
  → CSize
  → Ptr ()
  → Word32
  → Ptr CLEvent
  → Ptr CLEvent
```

Technologies

- Haskell
- Nix
- OpenCL bindings
- Grenade NN framework
  - Internally uses Hmatrix (= BLAS/LAPACK) for matrix operations

- (formerly also CUDA, raw OpenCL, ...)

Desired interface

```
type MNIST
  = Network
    '[ Convolution 1 10 5 5 1 1, Pooling 2 2 2 2, Relu
     , Convolution 10 16 5 5 1 1, Pooling 2 2 2 2, Reshape, Relu
     , FullyConnected 256 80, Logit, FullyConnected 80 10, Logit]
    '[ 'D2 28 28, 'D3 24 24 10, 'D3 12 12 10, 'D3 12 12 10
     , 'D3 8 8 16, 'D3 4 4 16, 'D1 256, 'D1 256
     , 'D1 80, 'D1 80, 'D1 10, 'D1 10]

randomMnist :: MonadRandom m => m MNIST
randomMnist = randomNetwork
```

Network building block

```
data Network :: [*] -> [Shape] -> * where
    NNil  :: SingI i
          => Network '[] '[i]

    (:~>) :: (SingI i, SingI h, Layer x i h)
          => !x
          -> !(Network xs (h ': hs))
          -> Network (x ': xs) (i ': h ': hs)
```

Test network

```
type FFNet = Network
  '[ FullyConnected 2 40
   , Tanh
   , FullyConnected 40 10
   , Relu
   , FullyConnected 10 1
   , Logit
   ]
  '[ 'D1 2
   , 'D1 40
   , 'D1 40
   , 'D1 10
   , 'D1 10
   , 'D1 1
   , 'D1 1
   ]
```

Test network API

Training API

```haskell
randomNet :: (MonadIO m, MonadRandom m) ⇒ m FFNet
randomNet = randomNetwork

netTrain :: FFNet → LearningParameters → Int → IO FFNet
netTrain net0 rate n = do
    inps ← replicateM n $ do
      s  ← getRandom
      return $ S1D $ SA.randomVector s SA.Uniform * 2 - 1
    let outs = flip map inps $ \(S1D v) →
                 if v `inCircle` (fromRational 0.33, 0.33) || v `inCircle` (fromRational (-0.33), 0.33)
                   then S1D $ fromRational 1
                   else S1D $ fromRational 0

    let trained = foldl' trainEach net0 (zip inps outs)
    return trained
```

Accelerated layer internals

```
data FullyConnectedCL i o = FullyConnectedCL
  { kUpdate :: !CLKernel
  , kForward :: !CLKernel
  , kBackward :: !CLKernel
  , lpRef :: !(IORef (Maybe LearningParameters))
  , wTape :: !(CLBuffer (R i))          -- work vector
  , wIn :: !(CLBuffer (R i))          -- work vector
  , wOut :: !(CLBuffer (R o))          -- work vector
  , wGradient :: !(FullyConnected'CL i o)   -- work gradient
  , wWeights :: !(FullyConnected'CL i o)   -- Neuron weights
  , wMomentum :: !(FullyConnected'CL i o)   -- Neuron momentum
  }

data FullyConnected'CL i o = FullyConnected'CL
  !(CLBuffer (R o))   -- Bias
  !(CLBuffer (L o i)) -- Activations
```

```haskell
randomFullyConnectedCL :: forall i o m. (MonadIO m, MonadRandom m, KnownNat i, KnownNat o)
                       => m (FullyConnectedCL i o)
randomFullyConnectedCL = do
  s1 <- getRandom
  s2 <- getRandom
  liftIO . withCL $ \cl -> do
    oB <- mkBufferR cl (randomVector s1 Uniform * 2 - 1)
    oA <- mkBufferL cl (uniformSample s2 (-1) 1)
    oBM <- mkBufferR cl (konst 0)
    oM <- mkBufferL cl (konst 0)
    bG <- mkBufferR cl (konst 0)
    aG <- mkBufferL cl (konst 0)
    wIn <- mkBufferR cl (konst 0)
    wOut <- mkBufferR cl (konst 0)
    wTape <- mkBufferR cl (konst 0)
    let wWeights = FullyConnected'CL oB oA
        wMomentum = FullyConnected'CL oBM oM
        wGradient = FullyConnected'CL bG aG

    kUpdate <- clCreateKernel (clProgram cl) "fcnn_update"
    clSetKernelArgSto' kUpdate 0 (fromIntegral @_ @Int32 (natVal (Proxy @o)))
    clSetKernelArgSto' kUpdate 1 (fromIntegral @_ @Int32 (natVal (Proxy @i)))
    clSetKernelArgSto' kUpdate 5 oB
    clSetKernelArgSto' kUpdate 6 oA
    clSetKernelArgSto' kUpdate 7 bG
    clSetKernelArgSto' kUpdate 8 aG
    clSetKernelArgSto' kUpdate 9 oBM
    clSetKernelArgSto' kUpdate 10 oM

    kForward <- clCreateKernel (clProgram cl) "fcnn_forward"
    clSetKernelArgSto' kForward 0 (fromIntegral @_ @Int32 (natVal (Proxy @o)))
    clSetKernelArgSto' kForward 1 (fromIntegral @_ @Int32 (natVal (Proxy @i)))
    clSetKernelArgSto' kForward 2 oA
    clSetKernelArgSto' kForward 3 wTape
    clSetKernelArgSto' kForward 4 oB
    clSetKernelArgSto' kForward 5 wOut

    kBackward <- clCreateKernel (clProgram cl) "fcnn_backward"
    clSetKernelArgSto' kBackward 0 (fromIntegral @_ @Int32 (natVal (Proxy @o)))
    clSetKernelArgSto' kBackward 1 (fromIntegral @_ @Int32 (natVal (Proxy @i)))
    clSetKernelArgSto' kBackward 2 bG
    clSetKernelArgSto' kBackward 3 wTape
    clSetKernelArgSto' kBackward 4 oA
    clSetKernelArgSto' kBackward 5 aG
    clSetKernelArgSto' kBackward 6 wIn

    lpRef <- newIORef Nothing

    pure $ FullyConnectedCL{..}
```

Backpropagation kernel

```
kernel void fcnn_backward(
    int rows,
    int cols,
    constant double *dEdy,
    constant double *x,
    constant double *wN,
    global double *mm,
    global double *dWs
) {
  int i = get_global_id(0);

  // let mm' = dEdy `outer` x
  if (i < rows) {
    const double d = dEdy[i];
    for (unsigned j = 0; j < cols; j++) {
      mm[i * rows + j] = d * x[j];
    }
  }

  // dWs  = tr wN #> dEdy
  if (i < cols) {
    double acc = 0;
    for (unsigned j = 0; j < rows; j++) {
      acc = mad(wN[i + rows * j], dEdy[j], acc);
    }
    dWs[i] = acc;
  }
}
```

Bottleneck – readVectorFromBuffer

```haskell
readBufferL :: forall m n. (KnownNat m, KnownNat n) ⇒ CLState → CLBuffer (L m n) → Maybe (L m n) → IO (L m n)
readBufferL cl (CLBuffer buf) mMat = do
  fPtr ← case mMat of
    Nothing → mallocForeignPtrArray len
    Just mat → pure $ fst (unsafeToForeignPtr0 (flatten . tr $ extract mat))
  _ ← withForeignPtr fPtr $ \ptr →
    clEnqueueReadBuffer' (clQueue cl) buf True 0 size (castPtr ptr)
  pure (fromJust . create . matrixFromVector ColumnMajor rows cols $ unsafeFromForeignPtr0 fPtr len)
  where
    rows = fromIntegral (natVal (Proxy @m))
    cols = fromIntegral (natVal (Proxy @n))
    len = rows * cols
    size = sizeOf (undefined :: CDouble) * len
```

```
Tue Jan  5 13:18 2021 Time and Allocation Profiling Report  (Final)

   feedforward +RTS -p -s -RTS

   total time  =        1.65 secs   (1654 ticks @ 1000 us, 1 processor)
   total alloc = 1,245,923,136 bytes  (excludes profiling overheads)

COST CENTRE          MODULE                SRC                                                %time %alloc

runNetwork           Grenade.Core.Network  src/Grenade/Core/Network.hs:(103,1)-(116,17)         30.6   32.2
netTrain             Main                  main/feedforward.hs:(38,1)-(53,53)                 14.8    8.9
clEnqueueNDRangeKernel' Grenade.OpenCL.Context src-gmu/Grenade/OpenCL/Context.hs:(229,1)-(236,35)  11.7    9.3
readBufferR          Grenade.OpenCL.Context src-gmu/Grenade/OpenCL/Context.hs:(132,1)-(141,46)  11.0   12.2
applyUpdate          Grenade.Core.Network  src/Grenade/Core/Network.hs:(152,1)-(156,8)          8.8    6.3
clEnqueueWriteBuffer' Grenade.OpenCL.Context src-gmu/Grenade/OpenCL/Context.hs:(223,1)-(226,41)  5.3    2.7
clEnqueueReadBuffer' Grenade.OpenCL.Context src-gmu/Grenade/OpenCL/Context.hs:(216,1)-(219,41)  4.4    2.7
runGradient          Grenade.Core.Network  src/Grenade/Core/Network.hs:(131,1)-(144,17)         4.1   12.2
getStdRandom         System.Random         System/Random.hs:(586,1)-(587,26)                  3.3   10.2
writeBufferR         Grenade.OpenCL.Context src-gmu/Grenade/OpenCL/Context.hs:(124,1)-(129,46)  2.9    0.4


COST CENTRE                        MODULE                           SRC                                                no.  entries  individual %time %alloc  inherited %time %alloc

MAIN                               MAIN                             <built-in>                                         317        0    0.0   0.0   100.0  100.0
 CAF                               Main                             <entire-module>                                    633        0    0.0   0.0     0.0    0.0
  feedForward'                     Main                             main/feedforward.hs:(80,1)-(88,56)                 674        1    0.0   0.0     0.0    0.0
   long                            Options.Applicative.Builder      src/Options/Applicative/Builder.hs:161:1-32        675        6    0.0   0.0     0.0    0.0
    fieldMod                       Options.Applicative.Builder.Internal src/Options/Applicative/Builder/Internal.hs:132:1-28  676        6    0.0   0.0     0.0    0.0
   option                          Options.Applicative.Builder      src/Options/Applicative/Builder.hs:(367,1)-(372,65)  680        4    0.0   0.0     0.0    0.0
    mkParser                       Options.Applicative.Builder.Internal src/Options/Applicative/Builder/Internal.hs:(163,1)-(167,36)  681        4    0.0   0.0     0.0    0.0
   value                           Options.Applicative.Builder      src/Options/Applicative/Builder.hs:173:1-50        679        4    0.0   0.0     0.0    0.0
   short                           Options.Applicative.Builder      src/Options/Applicative/Builder.hs:157:1-34        677        2    0.0   0.0     0.0    0.0
    fieldMod                       Options.Applicative.Builder.Internal src/Options/Applicative/Builder/Internal.hs:132:1-28  678        2    0.0   0.0     0.0    0.0
   strOption                       Options.Applicative.Builder      src/Options/Applicative/Builder.hs:352:1-22        682        2    0.0   0.0     0.0    0.0
    option                         Options.Applicative.Builder      src/Options/Applicative/Builder.hs:(367,1)-(372,65)  683        2    0.0   0.0     0.0    0.0
     mkParser                      Options.Applicative.Builder.Internal src/Options/Applicative/Builder/Internal.hs:(163,1)-(167,36)  684        2    0.0   0.0     0.0    0.0
  main                             Main                             main/feedforward.hs:(91,1)-(96,16)                 634        1    0.0   0.0     0.0    0.0
   idm                             Options.Applicative.Builder      src/Options/Applicative/Builder.hs:526:1-12        647        1    0.0   0.0     0.0    0.0
   info                            Options.Applicative.Builder      src/Options/Applicative/Builder.hs:(443,1)-(452,34)  646        1    0.0   0.0     0.0    0.0
   netScore                        Main                             main/feedforward.hs:(61,1)-(75,29)                 808        0    0.0   0.0     0.0    0.0
   netTrain                        Main                             main/feedforward.hs:(38,1)-(53,53)                 818        0    0.0   0.0     0.0    0.0
    unId                           Data.Vector.Fusion.Util          Data/Vector/Fusion/Util.hs:25:21-24                 822        7    0.0   0.0     0.0    0.0
    sChunks                        Data.Vector.Fusion.Bundle.Monadic Data/Vector/Fusion/Bundle/Monadic.hs:122:30-36     821        2    0.0   0.0     0.0    0.0
    sSize                          Data.Vector.Fusion.Bundle.Monadic Data/Vector/Fusion/Bundle/Monadic.hs:124:30-34     820        2    0.0   0.0     0.0    0.0
    upperBound                     Data.Vector.Fusion.Bundle.Size   Data/Vector/Fusion/Bundle/Size.hs:(126,1)-(128,30)  819        2    0.0   0.0     0.0    0.0
 CAF                               Options.Applicative.Builder      <entire-module>                                    632        0    0.0   0.0     0.0    0.0
  hidden                           Options.Applicative.Builder      src/Options/Applicative/Builder.hs:(205,1)-(206,54)  692        1    0.0   0.0     0.0    0.0
   optionMod                       Options.Applicative.Builder.Internal src/Options/Applicative/Builder/Internal.hs:129:1-25  693        0    0.0   0.0     0.0    0.0
  abortOption                      Options.Applicative.Builder      src/Options/Applicative/Builder.hs:(341,1)-(344,16)  697        0    0.0   0.0     0.0    0.0
   metavar                         Options.Applicative.Builder      src/Options/Applicative/Builder.hs:201:1-55        699        1    0.0   0.0     0.0    0.0
    optionMod                      Options.Applicative.Builder.Internal src/Options/Applicative/Builder/Internal.hs:129:1-25  700        0    0.0   0.0     0.0    0.0
   value                           Options.Applicative.Builder      src/Options/Applicative/Builder.hs:173:1-50        698        1    0.0   0.0     0.0    0.0
  option                           Options.Applicative.Builder      src/Options/Applicative/Builder.hs:(367,1)-(372,65)  663        0    0.0   0.0     0.0    0.0
   metavar                         Options.Applicative.Builder      src/Options/Applicative/Builder.hs:201:1-55        664        1    0.0   0.0     0.0    0.0
    optionMod                      Options.Applicative.Builder.Internal src/Options/Applicative/Builder/Internal.hs:129:1-25  665        0    0.0   0.0     0.0    0.0
:
```

runNet	Grenade.Core.Runner	src/Grenade/Core/Runner.hs:57:1-33	809	1071	0.0	0.0	0.7	0.8
runNetwork	Grenade.Core.Network	src/Grenade/Core/Network.hs:(103,1)-(116,17)	811	0	0.4	0.5	0.7	0.8
unId	Data.Vector.Fusion.Util	Data/Vector/Fusion/Util.hs:25:21-24	867	34272	0.0	0.0	0.0	0.0
clEnqueueNDRangeKernel'	Grenade.OpenCL.Context	src-gmu/Grenade/OpenCL/Context.hs:(229,1)-(236,35)	859	3213	0.0	0.0	0.0	0.0
clQueue	Grenade.OpenCL.Context	src-gmu/Grenade/OpenCL/Context.hs:59:5-11	860	3213	0.0	0.0	0.0	0.0
kForward	Grenade.OpenCL.FullyConnected	src-gmu/Grenade/OpenCL/FullyConnected.hs:29:5-12	861	3213	0.0	0.0	0.0	0.0
readBufferR	Grenade.OpenCL.Context	src-gmu/Grenade/OpenCL/Context.hs:(132,1)-(141,46)	862	3213	0.2	0.1	0.2	0.2
unId	Data.Vector.Fusion.Util	Data/Vector/Fusion/Util.hs:25:21-24	863	12852	0.0	0.0	0.0	0.0
clEnqueueReadBuffer'	Grenade.OpenCL.Context	src-gmu/Grenade/OpenCL/Context.hs:(216,1)-(219,41)	866	3213	0.0	0.0	0.0	0.0
clQueue	Grenade.OpenCL.Context	src-gmu/Grenade/OpenCL/Context.hs:59:5-11	864	3213	0.0	0.0	0.0	0.0
sChunks	Data.Vector.Fusion.Bundle.Monadic	Data/Vector/Fusion/Bundle/Monadic.hs:122:30-36	870	3213	0.0	0.0	0.0	0.0
sSize	Data.Vector.Fusion.Bundle.Monadic	Data/Vector/Fusion/Bundle/Monadic.hs:124:30-34	869	3213	0.0	0.0	0.0	0.0
upperBound	Data.Vector.Fusion.Bundle.Size	Data/Vector/Fusion/Bundle/Size.hs:(126,1)-(128,30)	868	3213	0.0	0.0	0.0	0.0
wOut	Grenade.OpenCL.FullyConnected	src-gmu/Grenade/OpenCL/FullyConnected.hs:34:5-8	865	3213	0.0	0.0	0.0	0.0
wTape	Grenade.OpenCL.FullyConnected	src-gmu/Grenade/OpenCL/FullyConnected.hs:32:5-9	857	3213	0.0	0.0	0.0	0.0
writeBufferR	Grenade.OpenCL.Context	src-gmu/Grenade/OpenCL/Context.hs:(124,1)-(129,46)	850	3213	0.1	0.0	0.1	0.0
clEnqueueWriteBuffer'	Grenade.OpenCL.Context	src-gmu/Grenade/OpenCL/Context.hs:(223,1)-(226,41)	858	3213	0.0	0.0	0.0	0.0
clQueue	Grenade.OpenCL.Context	src-gmu/Grenade/OpenCL/Context.hs:59:5-11	856	3213	0.0	0.0	0.0	0.0
unId	Data.Vector.Fusion.Util	Data/Vector/Fusion/Util.hs:25:21-24	855	3213	0.0	0.0	0.0	0.0
netTrain	Main	main/feedforward.hs:(38,1)-(53,53)	802	1	14.8	8.9	99.2	98.9
unId	Data.Vector.Fusion.Util	Data/Vector/Fusion/Util.hs:25:21-24	817	1300000	0.0	0.0	0.0	0.0
train	Grenade.Core.Runner	src/Grenade/Core/Runner.hs:(50,1)-(52,38)	812	100000	0.3	0.0	81.0	79.8
applyUpdate	Grenade.Core.Network	src/Grenade/Core/Network.hs:(152,1)-(156,8)	840	700000	8.8	6.3	15.3	11.3
lpRef	Grenade.OpenCL.FullyConnected	src-gmu/Grenade/OpenCL/FullyConnected.hs:31:5-9	841	300003	0.0	0.0	0.0	0.0
clEnqueueNDRangeKernel'	Grenade.OpenCL.Context	src-gmu/Grenade/OpenCL/Context.hs:(229,1)-(236,35)	848	300000	6.0	4.6	6.0	4.6
clQueue	Grenade.OpenCL.Context	src-gmu/Grenade/OpenCL/Context.hs:59:5-11	849	300000	0.5	0.4	0.5	0.4
kUpdate	Grenade.OpenCL.FullyConnected	src-gmu/Grenade/OpenCL/FullyConnected.hs:28:5-11	845	300000	0.0	0.0	0.0	0.0
clSetKernelArgSto'	Grenade.OpenCL.Context	src-gmu/Grenade/OpenCL/Context.hs:(239,1)-(240,79)	844	9	0.0	0.0	0.0	0.0
learningMomentum	Grenade.Core.LearningParameters	src/Grenade/Core/LearningParameters.hs:19:5-20	846	3	0.0	0.0	0.0	0.0
learningRate	Grenade.Core.LearningParameters	src/Grenade/Core/LearningParameters.hs:18:5-16	842	3	0.0	0.0	0.0	0.0
learningRegulariser	Grenade.Core.LearningParameters	src/Grenade/Core/LearningParameters.hs:20:5-23	847	3	0.0	0.0	0.0	0.0
backPropagate	Grenade.Core.Runner	src/Grenade/Core/Runner.hs:(37,1)-(40,13)	813	100000	1.0	0.3	65.4	68.5
runGradient	Grenade.Core.Network	src/Grenade/Core/Network.hs:(131,1)-(144,17)	815	100000	4.1	12.2	4.1	12.2
runNetwork	Grenade.Core.Network	src/Grenade/Core/Network.hs:(103,1)-(116,17)	814	0	30.2	31.7	60.4	56.1
wTape	Grenade.OpenCL.FullyConnected	src-gmu/Grenade/OpenCL/FullyConnected.hs:32:5-9	825	600000	0.5	0.8	0.5	0.8
unId	Data.Vector.Fusion.Util	Data/Vector/Fusion/Util.hs:25:21-24	839	500000	0.0	0.0	0.0	0.0
clEnqueueNDRangeKernel'	Grenade.OpenCL.Context	src-gmu/Grenade/OpenCL/Context.hs:(229,1)-(236,35)	827	300000	5.7	4.6	5.7	4.6
clQueue	Grenade.OpenCL.Context	src-gmu/Grenade/OpenCL/Context.hs:59:5-11	828	300000	0.3	0.4	0.3	0.4
kForward	Grenade.OpenCL.FullyConnected	src-gmu/Grenade/OpenCL/FullyConnected.hs:29:5-12	829	300000	0.3	0.4	0.3	0.4
readBufferR	Grenade.OpenCL.Context	src-gmu/Grenade/OpenCL/Context.hs:(132,1)-(141,46)	830	300000	10.8	12.1	15.1	14.8
unId	Data.Vector.Fusion.Util	Data/Vector/Fusion/Util.hs:25:21-24	835	1100000	0.0	0.0	0.0	0.0
clEnqueueReadBuffer'	Grenade.OpenCL.Context	src-gmu/Grenade/OpenCL/Context.hs:(216,1)-(219,41)	838	300000	4.4	2.7	4.4	2.7
clQueue	Grenade.OpenCL.Context	src-gmu/Grenade/OpenCL/Context.hs:59:5-11	836	300000	0.0	0.0	0.0	0.0
wOut	Grenade.OpenCL.FullyConnected	src-gmu/Grenade/OpenCL/FullyConnected.hs:34:5-8	837	300000	0.2	0.4	0.2	0.4
writeBufferR	Grenade.OpenCL.Context	src-gmu/Grenade/OpenCL/Context.hs:(124,1)-(129,46)	816	300000	2.8	0.4	8.1	3.1
clEnqueueWriteBuffer'	Grenade.OpenCL.Context	src-gmu/Grenade/OpenCL/Context.hs:(223,1)-(226,41)	826	300000	5.3	2.7	5.3	2.7
clQueue	Grenade.OpenCL.Context	src-gmu/Grenade/OpenCL/Context.hs:59:5-11	824	300000	0.0	0.0	0.0	0.0
unId	Data.Vector.Fusion.Util	Data/Vector/Fusion/Util.hs:25:21-24	823	300000	0.0	0.0	0.0	0.0
getStdRandom	System.Random	System/Random.hs:(586,1)-(587,26)	803	0	3.3	10.2	3.3	10.2
randomNet	Main	main/feedforward.hs:35:1-25	705	1	0.0	0.0	0.0	0.0
randomFullyConnectedCL	Grenade.OpenCL.FullyConnected	src-gmu/Grenade/OpenCL/FullyConnected.hs:(99,1)-(145,31)	706	3	0.0	0.0	0.0	0.0
unId	Data.Vector.Fusion.Util	Data/Vector/Fusion/Util.hs:25:21-24	774	592	0.0	0.0	0.0	0.0
clSetKernelArgSto'	Grenade.OpenCL.Context	src-gmu/Grenade/OpenCL/Context.hs:(239,1)-(240,79)	801	63	0.0	0.0	0.0	0.0
mkBufferR	Grenade.OpenCL.Context	src-gmu/Grenade/OpenCL/Context.hs:(158,1)-(163,46)	769	18	0.0	0.0	0.0	0.0
unId	Data.Vector.Fusion.Util	Data/Vector/Fusion/Util.hs:25:21-24	780	50	0.0	0.0	0.0	0.0
clContext	Grenade.OpenCL.Context	src-gmu/Grenade/OpenCL/Context.hs:57:5-13	772	18	0.0	0.0	0.0	0.0

```haskell
instance (KnownNat i, KnownNat o) ⇒ UpdateLayer (FullyConnectedCL i o) where
  type Gradient (FullyConnectedCL i o) = (FullyConnected'CL i o)

  --   let (nB, nBM) = descendVector lp oB bG oBM
  --       (nA, nM) = descendMatrix lp oA aG oM
  --   in FullyConnectedCL (FullyConnected'CL nB nA) (FullyConnected'CL nBM nM)
  runUpdate lp n _ =
    unsafeWithCL $ \cl → do
      let k = kUpdate n
      lastLp ← readIORef (lpRef n)
      unless (Just lp == lastLp) $ do
        clSetKernelArgSto' k 2 (learningRate lp)
        clSetKernelArgSto' k 3 (learningMomentum lp)
        clSetKernelArgSto' k 4 (learningRegulariser lp)
        writeIORef (lpRef n) (Just lp)
      _ ← clEnqueueNDRangeKernel' (clQueue cl) k [len]
      pure n
    where
      len = fromIntegral (natVal (Proxy @o))

  createRandom = randomFullyConnectedCL

instance (KnownNat i, KnownNat o) ⇒ Layer (FullyConnectedCL i o) ('D1 i) ('D1 o) where
  type Tape (FullyConnectedCL i o) ('D1 i) ('D1 o) = CLBuffer (R i)

  -- Do a matrix vector multiplication and return the result.
  -- (v, S1D (wB + wN #> v))
  runForwards n (S1D v) =
    unsafeWithCL $ \cl → do
      writeBufferR cl (wTape n) v
      _ ← clEnqueueNDRangeKernel' (clQueue cl) (kForward n) [len]
      res ← readBufferR cl (wOut n) (Just (konst 0))
      pure (wTape n, S1D res)
    where
      len = fromIntegral (natVal (Proxy @o))

  -- Run a backpropogation step for a full connected layer.
  -- let wB' = dEdy; mm' = dEdy `outer` x
  --     dWs  = tr wN #> dEdy
  -- in  (FullyConnected'CL wB' mm', S1D dWs)
  runBackwards n _ (S1D dEdy) =
    unsafeWithCL $ \cl → do
      let FullyConnected'CL bG _ = wGradient n
      writeBufferR cl bG dEdy
      _ ← clEnqueueNDRangeKernel' (clQueue cl) (kBackward n) [len]
      dWs ← readBufferR cl (wIn n) (Just (konst 0))
      pure (wGradient n, S1D dWs)
    where
      len = fromIntegral (max (natVal (Proxy @o)) (natVal (Proxy @i)))
```

```
{
  inputs.nixpkgs.url = github:nixos/nixpkgs/master;
  inputs.CLUtil = { url = github:acowley/CLUtil/master; flake = false; };

  outputs = { self, nixpkgs, CLUtil }: let
    inherit (nixpkgs.lib) flip mapAttrs mapAttrsToList;
    inherit (pkgs.nix-gitignore) gitignoreSourcePure;

    pkgs = import nixpkgs {
      system = "x86_64-linux";
      config.allowUnfree = true;
      overlays = [ self.overlay ];
    };
    hsPkgs = pkgs.haskellPackages;
    getSrc = dir: gitignoreSourcePure [./.gitignore] dir;
  in {
    overlay = final: prev: let
      inherit (prev.haskell.lib) doJailbreak dontCheck unmarkBroken overrideCabal dontHaddock;
    in {
      haskell = prev.haskell // {
        packageOverrides = prev.lib.composeExtensions
          (prev.haskell.packageOverrides or (_: _: {}))
          (hself: hsuper: {
            grenade = hself.callCabal2nix "grenade" (getSrc ./grenade) {};
            CLUtil = dontHaddock (dontCheck (hself.callCabal2nix "CLUtil" CLUtil {}));
            OpenCL = overrideCabal (dontCheck (doJailbreak (unmarkBroken hsuper.openCL))) (drv: {
              configureFlags = (drv.configureFlags or []) ++ [
                "--extra-lib-dirs=${pkgs.ocl-icd}/lib"
                "--extra-include-dirs=${pkgs.opencl-headers}/include"
              ];
            });
            profiteur = doJailbreak (unmarkBroken hsuper.profiteur);
          });
      };
    };

    devShell.x86_64-linux = hsPkgs.shellFor {
      withHoogle = true;
      packages = p: [ p.grenade ];
      buildInputs = [
        hsPkgs.cabal-install
        hsPkgs.haskell-language-server
        hsPkgs.stylish-haskell
        hsPkgs.profiteur
      ];
    };
  };
}
```