nnirror

Claude Heiland-Allen

2018–2019

Contents

1 .gitignore 2
2 index.html 2
3 LICENSE.md 3
4 Makefile 8
5 nnirror.cabal 8
6 nnirror.gnuplot 9
7 nnirror.hs 9
8 nnirror.jpg 24
9 nnirror-plot.sh 24
10 nnirror.png 25
11 nnirror-score.c 25
12 nnirror.sh 26
13 scoreboard.png 26
14 Setup.hs 27
15 training.png 27

1 .gitignore

```
dist/
dist-newstyle/
.ghc.environment*
cabal.project
*.log
*.net
*.png
*.ppm
*.mkv
*.mp4
*.prof
*.hp
score
-/
```

2 index.html

```
<!DOCTYPE html>
<html><head><title>nnirror :: mathr</title>
<meta name="description" content="A neural network looking at itself." />
<meta property="og:description" content="A neural network looking at itself." />
<style>
html { color: white; background-color: black; }
```

```
body { color: black; background-color: white; width: 512px; margin: auto; ↙
    ↳ padding-left: 10px; padding-right: 10px; }
h1, h2 { text-align: left; }
p { text-align: justify; }
img { vertical-align: middle; }
a { text-decoration: none; }
a img { border: 1px solid; }
</style></head><body>
<h1>nnirror</h1>
<p><a href="nnirror.png" title="nnirror"><img src="nnirror.jpg" alt="nnirror" ↙
    ↳ /></a><p>
A neural network looking at itself.
</p>
<h2>Process</h2>
<p>
There are two neural networks, Ego and Id.
Ego ideally outputs 1 when the input is its own weights, and 0 otherwise.
Id ideally outputs weights for the Ego net, that Ego scores highly.
These are continually trained against each other using a generative
adversarial network.
</p>
<h2>Output</h2>
<p><a href="nnirror.mp4" title="video">Video excerpt of training process (100MB)↙
    ↳ </a>.</p>
<p>
<a href="1" title="hi score table">Top ten highest scoring networks</a>
out of 1000 training sessions.
</p>
<h2>Source</h2>
<p>
<a href="https://code.mathr.co.uk/nnirror" title="nnirror">nnirror</a>
is implemented in Haskell using
<a href="https://hackage.haskell.org/package/grenade" title="grenade">grenade</a↙
    ↳ >.
</p>
<h2>Legal</h2>
<p>
Copyleft: This is a free work, you can copy, distribute, and
modify it under the terms of the
<a href="http://artlibre.org/licence/lal/en/" title="Free Art License">Free Art ↙
    ↳ License</a>.
</p>
<h2>Author</h2>
<p><a href=".." title="mathr.co.uk">Claude Heiland-Allen</a></p>
</body></html>
```

3 LICENSE.md

```
# Free Art License 1.3 (FAL 1.3)

## Preamble

The Free Art License grants the right to freely copy, distribute, and
transform creative works without infringing the author's rights.

The Free Art License recognizes and protects these rights. Their
implementation has been reformulated in order to allow everyone to use
```

creations of the human mind in a creative manner, regardless of their types and ways of expression.

While the public's access to creations of the human mind usually is restricted by the implementation of copyright law, it is favoured by the Free Art License. This license intends to allow the use of a work's resources; to establish new conditions for creating in order to increase creation opportunities. The Free Art License grants the right to use a work, and acknowledges the right holder's and the user's rights and responsibility.

The invention and development of digital technologies, Internet and Free Software have changed creation methods: creations of the human mind can obviously be distributed, exchanged, and transformed. They allow to produce common works to which everyone can contribute to the benefit of all.

The main rationale for this Free Art License is to promote and protect these creations of the human mind according to the principles of copyleft: freedom to use, copy, distribute, transform, and prohibition of exclusive appropriation.

Definitions

work either means the initial work, the subsequent works or the common work as defined hereafter:

common work means a work composed of the initial work and all subsequent contributions to it (originals and copies). The initial author is the one who, by choosing this license, defines the conditions under which contributions are made.

Initial work means the work created by the initiator of the common work (as defined above), the copies of which can be modified by whoever wants to

Subsequent works means the contributions made by authors who participate in the evolution of the common work by exercising the rights to reproduce, distribute, and modify that are granted by the license.

Originals (sources or resources of the work) means all copies of either the initial work or any subsequent work mentioning a date and used by their author(s) as references for any subsequent updates, interpretations, copies or reproductions.

Copy means any reproduction of an original as defined by this license.

1. OBJECT

The aim of this license is to define the conditions under which one can use this work freely.

2. SCOPE

This work is subject to copyright law. Through this license its author specifies the extent to which you can copy, distribute, and modify it.

2.1 FREEDOM TO COPY (OR TO MAKE REPRODUCTIONS)

You have the right to copy this work for yourself, your friends or any other person, whatever the technique used.

2.2 FREEDOM TO DISTRIBUTE, TO PERFORM IN PUBLIC

You have the right to distribute copies of this work; whether modified or not, whatever the medium and the place, with or without any charge, provided that you:

- attach this license without any modification to the copies of this work or indicate precisely where the license can be found,
- specify to the recipient the names of the author(s) of the originals, including yours if you have modified the work,
- specify to the recipient where to access the originals (either initial or subsequent).

The authors of the originals may, if they wish to, give you the right to distribute the originals under the same conditions as the copies.

2.3 FREEDOM TO MODIFY

You have the right to modify copies of the originals (whether initial or subsequent) provided you comply with the following conditions:

- all conditions in article 2.2 above, if you distribute modified copies;
- indicate that the work has been modified and, if it is possible, what kind of modifications have been made;
- distribute the subsequent work under the same license or any compatible license.

The author(s) of the original work may give you the right to modify it under the same conditions as the copies.

3. RELATED RIGHTS

Activities giving rise to author's rights and related rights shall not challenge the rights granted by this license.
For example, this is the reason why performances must be subject to the same license or a compatible license. Similarly, integrating the work in a database, a compilation or an anthology shall not prevent anyone from using the work under the same conditions as those defined in this license.

4. INCORPORATION OF THE WORK

Incorporating this work into a larger work that is not subject to the Free Art License shall not challenge the rights granted by this license.
If the work can no longer be accessed apart from the larger work in which it is incorporated, then incorporation shall only be allowed under the condition that the larger work is subject either to the Free Art License or a compatible license.

5. COMPATIBILITY

A license is compatible with the Free Art License provided:

- it gives the right to copy, distribute, and modify copies of the work including for commercial purposes and without any other restrictions than those required by the respect of the other compatibility criteria;
- it ensures proper attribution of the work to its authors and access to previous versions of the work when possible;
- it recognizes the Free Art License as compatible (reciprocity);
- it requires that changes made to the work be subject to the same license or to a license which also meets these compatibility criteria.

6. YOUR INTELLECTUAL RIGHTS

This license does not aim at denying your author's rights in your contribution or any related right. By choosing to contribute to the development of this common work, you only agree to grant others the same rights with regard to your contribution as those you were granted by this license. Conferring these rights does not mean you have to give up your intellectual rights.

7. YOUR RESPONSIBILITIES

The freedom to use the work as defined by the Free Art License (right to copy, distribute, modify) implies that everyone is responsible for their own actions.

8. DURATION OF THE LICENSE

This license takes effect as of your acceptance of its terms. The act of copying, distributing, or modifying the work constitutes a tacit agreement. This license will remain in effect for as long as the copyright which is attached to the work. If you do not respect the terms of this license, you automatically lose the rights that it confers.
If the legal status or legislation to which you are subject makes it impossible for you to respect the terms of this license, you may not make use of the rights which it confers.

9. VARIOUS VERSIONS OF THE LICENSE

This license may undergo periodic modifications to incorporate improvements by its authors (instigators of the Copyleft Attitude movement) by way of new, numbered versions.
You will always have the choice of accepting the terms contained in the version under which the copy of the work was distributed to you, or alternatively, to use the provisions of one of the subsequent versions.

10. SUB-LICENSING

Sub-licenses are not authorized by this license. Any person wishing to make use of the rights that it confers will be directly bound to the authors of the common work.

11. LEGAL FRAMEWORK

This license is written with respect to both French law and the Berne

```
Convention for the Protection of Literary and Artistic Works.

## USER GUIDE

### How to use the Free Art License?

To benefit from the Free Art License, you only need to mention the
following elements on your work:

    [Name of the author, title, date of the work. When applicable,
    names of authors of the common work and, if possible, where to
    find the originals].

    Copyleft: This is a free work, you can copy, distribute, and
    modify it under the terms of the Free Art License
    <http://artlibre.org/licence/lal/en/>

### Why to use the Free Art License?

1. To give the greatest number of people access to your work.
2. To allow it to be distributed freely.
3. To allow it to evolve by allowing its copy, distribution, and
   transformation by others.
4. So that you benefit from the resources of a work when it is under the
   Free Art License: to be able to copy, distribute or transform it
   freely.
5. But also, because the Free Art License offers a legal framework to
   disallow any misappropriation. It is forbidden to take hold of your work
   and bypass the creative process for one's exclusive possession.

### When to use the Free Art License?

Any time you want to benefit and make others benefit from the right to
copy, distribute and transform creative works without any exclusive
appropriation, you should use the Free Art License. You can for example
use it for scientific, artistic or educational projects.

### What kinds of works can be subject to the Free Art License?

The Free Art License can be applied to digital as well as physical
works.
You can choose to apply the Free Art License on any text, picture,
sound, gesture, or whatever sort of stuff on which you have sufficient
author's rights.

### Historical background of this license:

It is the result of observing, using and creating digital technologies,
free software, the Internet and art. It arose from the Copyleft
Attitude meetings which took place in Paris in 2000. For the first
time, these meetings brought together members of the Free Software
community, artists, and members of the art world. The goal was to adapt
the principles of Copyleft and free software to all sorts of creations.

------------------------------------------------------------------------

<http://www.artlibre.org>
```

```
Copyleft Attitude, 2007.

You can make reproductions and distribute this license verbatim
(without any changes).

Translation: Jonathan Clarke, Benjamin Jean, Griselda Jung, Fanny Mourguet,
Antoine Pitrou. Thanks to <http://framalang.org>
```

4 Makefile

```
nnirror-score: nnirror-score.c
	gcc -o nnirror-score nnirror-score.c -lm
```

5 nnirror.cabal

```
name:                nnirror
version:             0

license:             OtherLicense
license-file:        LICENSE.md
author:              Claude Heiland-Allen
maintainer:          claude@mathr.co.uk
copyright:           (c) 2018,2019 Claude Heiland-Allen

cabal-version:       >= 1.10
build-type:          Simple

category:            Art
synopsis:            self-reflective neural net
description:
  A neural network looking at itself.
  .
  Copyleft: This is a free work, you can copy, distribute, and
  modify it under the terms of the Free Art License
  <http://artlibre.org/licence/lal/en/>.

homepage:            https://mathr.co.uk/nnirror

source-repository head
    type: git
    location: https://code.mathr.co.uk/nnirror.git

executable           nnirror
    main-is:         nnirror.hs
    default-language: Haskell2010
    other-extensions:
        BangPatterns
        DataKinds
        FlexibleContexts
        FlexibleInstances
        GADTs
        NoStarIsType
        PolyKinds
        ScopedTypeVariables
        TypeOperators
        UndecidableInstances
    build-depends:   base        == 4.12.*
```

```
                      , async        == 2.2.*
                      , cereal       == 0.5.*
                      , gl           == 0.8.*
                      , GLFW-b       == 3.2.*
                      , grenade      == 0.1.*
                      , hmatrix      == 0.19.*
                      , JuicyPixels  == 3.3.*
                      , MonadRandom  == 0.5.*
                      , singletons   == 2.5.*
                      , vector       == 0.12.*
    ghc-options: -rtsopts -threaded "-with-rtsopts=-N4 -A64M"
```

6 nnirror.gnuplot

```
u(x) = -log(1/x-1)
set autoscale
set xtics 1000
set ytics 2
set mxtics 10
set mytics 2
set grid xtics
set grid mxtics
set grid ytics
set grid mytics
unset key
set term pngcairo enhanced size 10080,7200 font "LMSans10,100" linewidth 3
set output "out.png"
set multiplot layout 4,1 title "becoming self-aware: training a neural network ↙
    ↳ to recognize itself"
set title "self-recognition"
set key bottom right maxrows 1 font ",75"
plot [0:9999][0:10] for [i=0:9] sprintf("%d.log", i) u 1:(u(column(2))) w l lc ↙
    ↳ rgbcolor hsv2rgb((i+1.5)/10,1,1) dt 1 t sprintf("%d", i)
set title "self-other demarcation"
set key bottom right maxrows 1 font ",75"
plot [0:9999][0:10] for [i=0:9] sprintf("%d.log", i) u 1:(-u(column(3))) w l lc ↙
    ↳ rgbcolor hsv2rgb((i+1.5)/10,1,1) dt 1 t sprintf("%d", i)
set title "pareidolia"
set key top right maxrows 1 font ",75"
plot [0:9999][-10:0] for [i=0:9] sprintf("%d.log", i) u 1:(u(column(4))) w l lc ↙
    ↳ rgbcolor hsv2rgb((i+1.5)/10,1,1) dt 1 t sprintf("%d", i)
set title "score"
set key bottom right maxrows 1 font ",75"
plot [0:9999][0:10] for [i=0:9] sprintf("%d.log", i) u 1:((2*u(column(2))-u(↙
    ↳ column(3))-u(column(4)))/4) w l lc rgbcolor hsv2rgb((i+1.5)/10,1,1) dt 1 t↙
    ↳  sprintf("%d", i)
unset multiplot
```

7 nnirror.hs

```
{-
  nnirror (c) 2018,2019 Claude Heiland-Allen
  A neural network looking at itself.

  Copyleft: This is a free work, you can copy, distribute, and
  modify it under the terms of the Free Art License
  <http://artlibre.org/licence/lal/en/>.
```

```
-}

{-# LANGUAGE BangPatterns #-}
{-# LANGUAGE DataKinds #-}
{-# LANGUAGE FlexibleContexts #-}
{-# LANGUAGE FlexibleInstances #-}
{-# LANGUAGE GADTs #-}
{-# LANGUAGE NoStarIsType #-}
{-# LANGUAGE PolyKinds #-}
{-# LANGUAGE ScopedTypeVariables #-}
{-# LANGUAGE TypeOperators #-}
{-# LANGUAGE UndecidableInstances #-}

import Grenade
import Codec.Picture (readImage, DynamicImage(ImageRGBA8), Image(Image))
import Control.Concurrent.Async (async, wait)
import Control.Exception (catch, SomeException)
import Control.Monad (forM_, replicateM, unless, when)
import Control.Monad.Random (MonadRandom, randomRIO)
import Data.Either (lefts, rights)
import Data.IORef (newIORef, modifyIORef, readIORef)
import Data.List (sort, sortBy, transpose)
import Data.Ord (comparing)
import Data.Proxy (Proxy(Proxy))
import Data.Serialize (Serialize(..), runGet, runPut)
import Data.Singletons
import Data.Vector.Storable (Vector)
import qualified Data.Vector.Storable as V
import Data.Word (Word8)
import Foreign (allocaBytes, castPtr, nullPtr, peek, peekArray, plusPtr, sizeOf,↙
    ↳ with, withArray)
import Foreign.C.String (withCString)
import GHC.TypeLits
import Graphics.GL.Core33
import qualified Graphics.UI.GLFW as GLFW
import qualified Numeric.LinearAlgebra as LA
import Numeric.LinearAlgebra.Static as LAS hiding (C, (|||), (===), (<>))
import System.Environment (getArgs)
import System.Exit (exitFailure, exitSuccess)
import System.IO (hFlush, hPutBuf, hPutStr, hPutStrLn, stderr, stdout)

type C = 10
type F = 48

type Ego = Network
  '[ Pad 1 1 1 1, Convolution 3 C 3 3 1 1, Pooling 2 2 2 2, Logit
   , Pad 1 1 1 1, Convolution C C 3 3 1 1, Pooling 2 2 2 2, Logit
   , Pad 1 1 0 1, Convolution C C 3 3 1 1, Pooling 2 2 2 2, Logit
   , Pad 1 1 1 1, Convolution C C 3 3 1 1, Pooling 2 2 2 2, Logit
   , Reshape
   , FullyConnected (4 * 2 * C) F, Logit
   , FullyConnected F 1, Logit
   ]
  '[ 'D3 64 36 3
   , 'D3 66 38 3, 'D3 64 36 C, 'D3 32 18 C, 'D3 32 18 C
   , 'D3 34 20 C, 'D3 32 18 C, 'D3 16  9 C, 'D3 16  9 C
   , 'D3 18 10 C, 'D3 16  8 C, 'D3  8  4 C, 'D3  8  4 C
```

```
      , 'D3 10  6 C, 'D3  8  4 C, 'D3  4  2 C, 'D3  4  2 C
      , 'D1 (4 * 2 * C)
      , 'D1 F, 'D1 F
      , 'D1 1, 'D1 1
      ]

type Id = Network
   '[ FullyConnected F (4 * 2 * C), Logit
    , Reshape
    , Deconvolution C C 3 3 2 2, Logit
    , Pad 1 1 0 0
    , Convolution C C 3 3 1 1, Logit
    , Deconvolution C C 3 3 2 2, Logit
    , Pad 1 1 1 0
    , Convolution C C 3 3 1 1, Logit
    , Deconvolution C C 3 3 2 2, Logit
    , Pad 1 1 0 0
    , Convolution C C 3 3 1 1, Logit
    , Deconvolution C C 3 3 2 2, Logit
    , Pad 1 1 0 0
    , Convolution C 3 3 3 1 1, Logit
    ]
   '[ 'D1 F
    , 'D1 (4 * 2 * C), 'D1 (4 * 2 * C)
    , 'D3  4  2 C
    , 'D3  9  5 C, 'D3  9  5 C
    , 'D3 10  6 C
    , 'D3 8   4 C, 'D3  8  4 C
    , 'D3 17  9 C, 'D3 17  9 C
    , 'D3 18 11 C
    , 'D3 16  9 C, 'D3 16  9 C
    , 'D3 33 19 C, 'D3 33 19 C
    , 'D3 34 20 C
    , 'D3 32 18 C, 'D3 32 18 C
    , 'D3 65 37 C, 'D3 65 37 C
    , 'D3 66 38 C
    , 'D3 64 36 3, 'D3 64 36 3
    ]
-- deconvolution  outputRows ~ (inputRows - 1) * stride + kernelRows
-- convolution    outputRows ~ (inputRows - kernelRows) / stride + 1

reflE :: Ego -> S ('D3 64 36 3)
reflE = sigmoid . S3D . (\(Just j) -> j) . LAS.create . LA.reshape 36 . V.take ↙
    ↳ (64 * 36 * 3) . (<> V.replicate (64 * 36 * 3) 0) . reflect

reflI :: Id -> S ('D3 64 55 3)
reflI = sigmoid . S3D . (\(Just j) -> j) . LAS.create . LA.reshape 55 . V.take ↙
    ↳ (64 * 55 * 3) . (<> V.replicate (64 * 55 * 3) 0) . reflect

sigmoid x = 1 / (1 + exp (-x))
unsigmoid y = - log (1 / y - 1)

bashNaN x
  | isNaN x || isInfinite x = 0
  | otherwise = x

reiE :: S ('D3 64 36 3) -> Ego
```

```
reiE input@(S3D s) = case reify . map bashNaN . LA.toList . LA.flatten . extract
    ↳ . unsigmoid $ s of
  Just (ego, _) -> ego

reiI :: S ('D3 64 55 3) -> Id
reiI input@(S3D s) = case reify . map bashNaN . LA.toList . LA.flatten . extract
    ↳ . unsigmoid $ s of
  Just (id, _) -> id

gan :: (LearningParameters, LearningParameters) -> (Ego, Id) -> IO ((Ego, Id), [
    ↳ Double])
gan (learnE, learnI) (ego, id) = do
  noise <- randomOfShape
  poopE <- randomOfShape
  let up :: LearningParameters -> Gradients l -> Network l s -> Network l s
      up l grad net = applyUpdate l net grad

      upE = up learnE
      upI = up learnI

  a_realE <- async $ do
    let !realE             = reflE ego
    return realE
  a_fakeE <- async $ do
    let r@(!idT, !fakeE)   = {-# SCC "runNetwork/id" #-} (runNetwork id noise)
    return r
  realE <- wait a_realE
  (idT, fakeE) <- wait a_fakeE

      -- train Ego

  a_real <- async $ do
    let r@(!realTE, !realGE) = {-# SCC "runNetwork/ego/real"   #-} (runNetwork
        ↳ ego realE)
    return r
  a_fake <- async $ do
    let r@(!fakeTE, !fakeGE) = {-# SCC "runNetwork/ego/fake"   #-} (runNetwork
        ↳ ego fakeE)
    return r
  a_poop <- async $ do
    let r@(!poopTE, !poopGE) = {-# SCC "runNetwork/ego/poop"   #-} (runNetwork
        ↳ ego poopE)
    return r

  (realTE, realGE) <- wait a_real
  (fakeTE, fakeGE) <- wait a_fake
  (poopTE, poopGE) <- wait a_poop

  a_real'E <- async $ do
    let (!real'E, !_)      = {-# SCC "runGradient/ego/real1" #-} (runGradient
        ↳ ego realTE (realGE - 1))
    return real'E
  a_fake'E <- async $ do
    let (!fake'E, !_)      = {-# SCC "runGradient/ego/fake"  #-} (runGradient
        ↳ ego fakeTE  fakeGE)
    return fake'E
  a_poop'E <- async $ do
```

```
    let (!poop'E, !_)      = {-# SCC "runGradient/ego/poop"  #-} (runGradient ↙
        ↳ ego poopTE  poopGE)
    return poop'E

  a_realPE <- async $ do
    let (!_, !realPE)      = {-# SCC "runGradient/ego/real"  #-} (runGradient ↙
        ↳ ego realTE  realGE)
    return realPE
  a_fakePE <- async $ do
    let (!_, !fakePE)      = {-# SCC "runGradient/ego/fake1" #-} (runGradient ↙
        ↳ ego fakeTE (fakeGE - 1))
    return fakePE
  a_poopPE <- async $ do
    let (!_, !poopPE)      = {-# SCC "runGradient/ego/poop1" #-} (runGradient ↙
        ↳ ego poopTE (poopGE - 1))
    return poopPE

  a_real'I <- async $ do
    realPE <- wait a_realPE
    let (!real'I, !_)      = {-# SCC "runGradient/id/real" #-} (runGradient id  ↙
        ↳ idT     realPE)
    return real'I
  a_fake'I <- async $ do
    fakePE <- wait a_fakePE
    let (!fake'I, !_)      = {-# SCC "runGradient/id/fake" #-} (runGradient id  ↙
        ↳ idT     fakePE)
    return fake'I
  a_poop'I <- async $ do
    poopPE <- wait a_poopPE
    let (!poop'I, !_)      = {-# SCC "runGradient/id/poop" #-} (runGradient id  ↙
        ↳ idT     poopPE)
    return poop'I

  a_ego <- async $ do
    real'E <- wait a_real'E
    fake'E <- wait a_fake'E
    poop'E <- wait a_poop'E
    let !newEgo = {-# SCC "update/ego" #-} (upE poop'E . upE fake'E . upE real'E↙
        ↳ $ ego)
    return newEgo

  a_id <- async $ do
    real'I <- wait a_real'I
    fake'I <- wait a_fake'I
    poop'I <- wait a_poop'I
    let !newId  = {-# SCC "update/id"  #-} (upI poop'I . upI fake'I . upI real'I↙
        ↳ $ id)
    return newId

  newEgo <- wait a_ego
  newId <- wait a_id
  let !real = fromSingleton realGE
      !fake = fromSingleton fakeGE
      !poop = fromSingleton poopGE

  return ((newEgo, newId), [real, fake, poop, (2 * unsigmoid real - unsigmoid ↙
      ↳ fake - unsigmoid poop) / 4])
```

```
215
    pixel :: Double -> Word8
    pixel x = round $ 255 * clamp 0 1 x

    clamp :: Ord a => a -> a -> a -> a
220 clamp lo hi x = lo `max` x `min` hi

    {-
    image :: Int -> Int -> Int -> [Double] -> Image
    image h w c = take h . chunk w . chunk c . map pixel . (++ repeat 0)
225 -}
    chunk :: Int -> [a] -> [[a]]
    chunk _ [] = []
    chunk n xs = case splitAt n xs of (ys, zs) -> ys : chunk n zs
    {-
230 type Image = [[[Word8]]]

    (|||), (===) :: Image -> Image -> Image
    (|||) = zipWith (++)
    (===) = (++)
235
    hcat, vcat :: [Image] -> Image
    hcat = foldr (|||) (repeat [])
    vcat = foldr (===) []
    -}
240 toList :: S ('D3 w h c) -> [Double]
    toList (S3D f) = LA.toList . LA.flatten . extract $ f

    {-
    display :: FilePath -> Ego -> Ego -> Ego -> IO ()
245 display file oldEgo ego newEgo = do
      let newEgoW = reflE newEgo
          egoW    = reflE ego
          oldEgoW = reflE oldEgo
          b = (7200 - 6912) `div` 2
250       border = (replicate b 1 ++) . (++ replicate b 1)
          newDelta = 0.5 + 0.5 * signum (egoW - oldEgoW)
          oldDelta = 0.5 + 0.5 * signum (newEgoW - egoW)
          diff = 1 - abs (newDelta - oldDelta)
          row = B.pack . map pixel . border . toList $ diff
255   B.appendFile file row
    -}

    recognition :: Ego -> S ('D3 64 36 3) -> Double
    recognition ego = fromSingleton . snd . runNetwork ego
260
    fromSingleton :: S ('D1 1) -> Double
    fromSingleton (S1D s) = case LA.toList (extract s) of [t] -> t

    {-
265 load :: Serialize a => FilePath -> IO a
    load file = either fail return . runGet get =<< B.readFile file

    save :: Serialize a => FilePath -> a -> IO ()
    save file = B.writeFile file . runPut . put
270 -}
```

```
orElse :: IO a -> IO a -> IO a
orElse a e = a `catch` \(_::SomeException) -> e

randomEgoId :: IO (Ego, Id)
randomEgoId = do
  ego <- randomNetwork
  id <- randomNetwork
  return (ego, id)

randomLearning :: IO LearningParameters
randomLearning =
  LearningParameters <$> randomRIO (0.25, 0.75) <*> randomRIO (0.25, 0.75) <*> ↙
      ↳ fmap (10**) (randomRIO (-16, 0))

--random :: IO (Ego, Id, LearningParameters, LearningParameters)
random = do
  (ego, id) <- randomEgoId
  learnE <- randomLearning
  learnI <- randomLearning
  return (ego, id, reflE ego, reflI id, learnE, learnI)

image :: S ('D3 w h 3) -> Vector Word8
image (S3D i) = normalize . LA.flatten . extract $ i

image' :: S ('D3 w h 3) -> Vector Word8
image' (S3D i) = V.map (pixel . (+ 0.5) . (/ 12) . unsigmoid) . LA.flatten . ↙
    ↳ extract $ i

normalize xs =
  let n = V.length xs
      mi = V.minimum xs
      ma = V.maximum xs
      f x = pixel $ (x - mi) / (ma - mi)
  in  V.map f xs

main :: IO ()
main = do
      let width, height :: Num a => a
          width  = 1920
          height = 1080
      args <- getArgs
      let record = "record" `elem` args
      GLFW.setErrorCallback (Just $ \err msg -> hPutStrLn stderr msg >> ↙
          ↳ exitFailure)
      True <- GLFW.init
      GLFW.windowHint $ GLFW.WindowHint'Resizable False
      GLFW.windowHint $ GLFW.WindowHint'ContextVersionMajor 3
      GLFW.windowHint $ GLFW.WindowHint'ContextVersionMinor 3
      GLFW.windowHint $ GLFW.WindowHint'OpenGLProfile GLFW.OpenGLProfile'Core
      GLFW.windowHint $ GLFW.WindowHint'OpenGLForwardCompat True
      Just window <- GLFW.createWindow width height "nnirror" Nothing Nothing
      GLFW.makeContextCurrent $ Just window
      --GLFW.swapInterval 2
      glViewport 0 0 width height
      glPixelStorei GL_PACK_ALIGNMENT 1
      glPixelStorei GL_UNPACK_ALIGNMENT 1
```

```
        tex <- withArray [0,0,0,0,0,0,0] $ \ptr -> glGenTextures 7 ptr >> ↙
            ↳ peekArray 7 ptr
        glActiveTexture GL_TEXTURE0
        glBindTexture GL_TEXTURE_2D (tex !! 0)
        glTexImage2D GL_TEXTURE_2D 0 GL_RGB 36 64 0 GL_RGB GL_UNSIGNED_BYTE ↙
            ↳ nullPtr
330     glTexParameteri GL_TEXTURE_2D GL_TEXTURE_MIN_FILTER GL_NEAREST
        glTexParameteri GL_TEXTURE_2D GL_TEXTURE_MAG_FILTER GL_NEAREST
        glTexParameteri GL_TEXTURE_2D GL_TEXTURE_WRAP_S GL_CLAMP_TO_EDGE
        glTexParameteri GL_TEXTURE_2D GL_TEXTURE_WRAP_T GL_CLAMP_TO_EDGE
        glActiveTexture GL_TEXTURE1
335     glBindTexture GL_TEXTURE_2D (tex !! 1)
        glTexImage2D GL_TEXTURE_2D 0 GL_RGB 55 64 0 GL_RGB GL_UNSIGNED_BYTE ↙
            ↳ nullPtr
        glTexParameteri GL_TEXTURE_2D GL_TEXTURE_MIN_FILTER GL_NEAREST
        glTexParameteri GL_TEXTURE_2D GL_TEXTURE_MAG_FILTER GL_NEAREST
        glTexParameteri GL_TEXTURE_2D GL_TEXTURE_WRAP_S GL_CLAMP_TO_EDGE
340     glTexParameteri GL_TEXTURE_2D GL_TEXTURE_WRAP_T GL_CLAMP_TO_EDGE
        glActiveTexture GL_TEXTURE2
        glBindTexture GL_TEXTURE_2D (tex !! 2)
        glTexImage2D GL_TEXTURE_2D 0 GL_RGB 36 64 0 GL_RGB GL_UNSIGNED_BYTE ↙
            ↳ nullPtr
        glTexParameteri GL_TEXTURE_2D GL_TEXTURE_MIN_FILTER GL_NEAREST
345     glTexParameteri GL_TEXTURE_2D GL_TEXTURE_MAG_FILTER GL_NEAREST
        glTexParameteri GL_TEXTURE_2D GL_TEXTURE_WRAP_S GL_CLAMP_TO_EDGE
        glTexParameteri GL_TEXTURE_2D GL_TEXTURE_WRAP_T GL_CLAMP_TO_EDGE
        glActiveTexture GL_TEXTURE3
        glBindTexture GL_TEXTURE_2D (tex !! 3)
350     glTexImage2D GL_TEXTURE_2D 0 GL_RGB 55 64 0 GL_RGB GL_UNSIGNED_BYTE ↙
            ↳ nullPtr
        glTexParameteri GL_TEXTURE_2D GL_TEXTURE_MIN_FILTER GL_NEAREST
        glTexParameteri GL_TEXTURE_2D GL_TEXTURE_MAG_FILTER GL_NEAREST
        glTexParameteri GL_TEXTURE_2D GL_TEXTURE_WRAP_S GL_CLAMP_TO_EDGE
        glTexParameteri GL_TEXTURE_2D GL_TEXTURE_WRAP_T GL_CLAMP_TO_EDGE
355     glActiveTexture GL_TEXTURE4
        glBindTexture GL_TEXTURE_2D (tex !! 4)
        glTexImage2D GL_TEXTURE_2D 0 GL_RGB 143 1 0 GL_RGB GL_UNSIGNED_BYTE ↙
            ↳ nullPtr
        glTexParameteri GL_TEXTURE_2D GL_TEXTURE_MIN_FILTER GL_NEAREST
        glTexParameteri GL_TEXTURE_2D GL_TEXTURE_MAG_FILTER GL_NEAREST
360     glTexParameteri GL_TEXTURE_2D GL_TEXTURE_WRAP_S GL_CLAMP_TO_EDGE
        glTexParameteri GL_TEXTURE_2D GL_TEXTURE_WRAP_T GL_CLAMP_TO_EDGE
        glActiveTexture GL_TEXTURE5
        glBindTexture GL_TEXTURE_2D (tex !! 5)
        e <- readImage "training.png"
365     case e of
          Right (ImageRGBA8 (Image w h dat)) -> V.unsafeWith dat $ \ptr -> do
            glTexImage2D GL_TEXTURE_2D 0 GL_RGBA (fromIntegral w) (fromIntegral h)↙
                ↳  0 GL_RGBA GL_UNSIGNED_BYTE (castPtr ptr)
            glGenerateMipmap GL_TEXTURE_2D
            glTexParameteri GL_TEXTURE_2D GL_TEXTURE_MIN_FILTER ↙
                ↳ GL_LINEAR_MIPMAP_LINEAR
370         glTexParameteri GL_TEXTURE_2D GL_TEXTURE_MAG_FILTER GL_LINEAR
            glTexParameteri GL_TEXTURE_2D GL_TEXTURE_WRAP_S GL_CLAMP_TO_EDGE
            glTexParameteri GL_TEXTURE_2D GL_TEXTURE_WRAP_T GL_CLAMP_TO_EDGE
        glActiveTexture GL_TEXTURE6
        glBindTexture GL_TEXTURE_2D (tex !! 6)
```

```
375       e <- readImage "scoreboard.png"
          case e of
            Right (ImageRGBA8 (Image w h dat)) -> V.unsafeWith dat $ \ptr -> do
              glTexImage2D GL_TEXTURE_2D 0 GL_RGBA (fromIntegral w) (fromIntegral h) 0 GL_RGBA GL_UNSIGNED_BYTE (castPtr ptr)
              glGenerateMipmap GL_TEXTURE_2D
380           glTexParameteri GL_TEXTURE_2D GL_TEXTURE_MIN_FILTER GL_LINEAR_MIPMAP_LINEAR
              glTexParameteri GL_TEXTURE_2D GL_TEXTURE_MAG_FILTER GL_LINEAR
              glTexParameteri GL_TEXTURE_2D GL_TEXTURE_WRAP_S GL_CLAMP_TO_EDGE
              glTexParameteri GL_TEXTURE_2D GL_TEXTURE_WRAP_T GL_CLAMP_TO_EDGE

385       frag <- glCreateShader GL_FRAGMENT_SHADER
          withCString (unlines
            [ "#version 330 core"
            , "uniform sampler2D dego, did, ego, id, score, mask;"
            , "uniform ivec2 size;"
390         , "layout(location = 0) out vec4 cout;"
            , "void main() {"
            , "  vec4 m = texture(mask, vec2(0.0, 1.0) + vec2(1.0, -1.0) * gl_FragCoord.xy / vec2(size));"
            , "  vec4 c = vec4(vec3(0.0), 1.0);" -- texture(score, vec2(1.0, 0.5)).yxzw;"
            , "  vec2 t = gl_FragCoord.xy * vec2(254.0, 143.0) / vec2(size);"
395         , "  if (5.0 <= t.x && t.x < 5.0 + 36.0 && 5.0 <= t.y && t.y < 5.0 + 64.0) { c = texture(dego, (t - vec2(5.0, 5.0)) / vec2(36.0, 64.0)); }"
            , "  if (5.0 + 36.0 + 5.0 <= t.x && t.x < 5.0 + 36.0 + 5.0 + 55.0 && 5.0 <= t.y && t.y < 5.0 + 64.0) { c = texture(did, (t - vec2(5.0 + 36.0 + 5.0, 5.0)) / vec2(55.0, 64.0)); }"
            , "  if (5.0 <= t.x && t.x < 5.0 + 36.0 && 5.0 + 64.0 + 5.0 <= t.y && t.y < 5.0 + 64.0 + 5.0 + 64.0) { c = texture(ego, (t - vec2(5.0, 5.0 + 64.0 + 5.0)) / vec2(36.0, 64.0)); }"
            , "  if (5.0 + 36.0 + 5.0 <= t.x && t.x < 5.0 + 36.0 + 5.0 + 55.0 && 5.0 + 64.0 + 5.0 <= t.y && t.y < 5.0 + 64.0 + 5.0 + 64.0) { c = texture(id, (t - vec2(5.0 + 36.0 + 5.0, 5.0 + 64.0 + 5.0)) / vec2(55.0, 64.0)); }"
            , "  if (5.0 + 36.0 + 5.0 + 55.0 + 5.0 <= t.x && t.x < 5.0 + 36.0 + 5.0 + 55.0 + 5.0 + 143.0 && 5.0 <= t.y && t.y < 5.0 + 41.0 ) { c = vec4(vec3((t.y - 5.0) / 41.0 < texture(score, vec2((t.x - (5.0 + 36.0 + 5.0 + 55.0 + 5.0))/143.0, 0.5))[2] ? 0.75 : 0.25), 1.0); }"
400         , "  if (5.0 + 36.0 + 5.0 + 55.0 + 5.0 <= t.x && t.x < 5.0 + 36.0 + 5.0 + 55.0 + 5.0 + 143.0 && 5.0 + 41.0 + 5.0 <= t.y && t.y < 5.0 + 41.0 + 5.0 + 41.0) { c = vec4(vec3((t.y - (5.0 + 41.0 + 5.0)) / 41.0 < texture(score, vec2((t.x - (5.0 + 36.0 + 5.0 + 55.0 + 5.0))/143.0, 0.5))[1] ? 0.75 : 0.25), 1.0); }"
            , "  if (5.0 + 36.0 + 5.0 + 55.0 + 5.0 <= t.x && t.x < 5.0 + 36.0 + 5.0 + 55.0 + 5.0 + 143.0 && 5.0 + 41.0 + 5.0 + 41.0 + 5.0 <= t.y && t.y < 5.0 + 41.0 + 5.0 + 41.0 + 5.0 + 41.0) { c = vec4(vec3((t.y - (5.0 + 41.0 + 5.0 + 41.0 + 5.0)) / 41.0 < texture(score, vec2((t.x - (5.0 + 36.0 + 5.0 + 55.0 + 5.0))/143.0, 0.5))[0] ? 0.75 : 0.25), 1.0); }"
            , "  cout = mix(c, m, m.a);\n"
            , "}"
            ]) $ \s -> with s $ \string -> glShaderSource frag 1 string nullPtr
405       glCompileShader frag
          vert <- glCreateShader GL_VERTEX_SHADER
```

```
        withCString (unlines
          [ "#version 330 core"
          , "void main() {"
410       , "  if (gl_VertexID == 0) gl_Position = vec4(-1.0, -1.0, 0.0, 1.0);"
          , "  if (gl_VertexID == 1) gl_Position = vec4(-1.0,  1.0, 0.0, 1.0);"
          , "  if (gl_VertexID == 2) gl_Position = vec4( 1.0, -1.0, 0.0, 1.0);"
          , "  if (gl_VertexID == 3) gl_Position = vec4( 1.0,  1.0, 0.0, 1.0);"
          , "}"
415       ]) $ \s -> with s $ \string -> glShaderSource vert 1 string nullPtr
        glCompileShader vert
        prog <- glCreateProgram
        glAttachShader prog vert
        glAttachShader prog frag
420     glLinkProgram prog
        glUseProgram prog
        flip glUniform1i 0 =<< (withCString "dego" $ glGetUniformLocation prog)
        flip glUniform1i 1 =<< (withCString "did"  $ glGetUniformLocation prog)
        flip glUniform1i 2 =<< (withCString "ego"  $ glGetUniformLocation prog)
425     flip glUniform1i 3 =<< (withCString "id"   $ glGetUniformLocation prog)
        flip glUniform1i 4 =<< (withCString "score"$ glGetUniformLocation prog)
        flip glUniform1i 5 =<< (withCString "mask" $ glGetUniformLocation prog)
        (\l -> glUniform2i l width height) =<< (withCString "size" $ ↙
            ↳ glGetUniformLocation prog)
        vao <- with 0 $ \p -> glGenVertexArrays 1 p >> peek p
430     glBindVertexArray vao

        frag <- glCreateShader GL_FRAGMENT_SHADER
        withCString (unlines
          [ "#version 330 core"
435       , "in float hue;"
          , "layout(location = 0) out vec4 cout;"
          , "// http://lolengine.net/blog/2013/07/27/rgb-to-hsv-in-glsl"
          , "vec3 hsv2rgb(vec3 c)"
          , "{"
440       , "    vec4 K = vec4(1.0, 2.0 / 3.0, 1.0 / 3.0, 3.0);"
          , "    vec3 p = abs(fract(c.xxx + K.xyz) * 6.0 - K.www);"
          , "    return c.z * mix(K.xxx, clamp(p - K.xxx, 0.0, 1.0), c.y);"
          , "}"
          , "void main() {"
445       , "  cout = vec4(1.0);//vec4(hsv2rgb(vec3(hue, 1.0, 1.0)), 1.0);\n"
          , "}"
          ]) $ \s -> with s $ \string -> glShaderSource frag 1 string nullPtr
        glCompileShader frag
        vert <- glCreateShader GL_VERTEX_SHADER
450     withCString (unlines
          [ "#version 330 core"
          , "layout(location = 0) in vec4 v;"
          , "out float hue;"
          , "void main() {"
455       , "  hue = -log(v.z) / (20.0 * log(10.0));"
          , "  int n = int(round(abs(v.w))) - 1;"
          , "  float s = float((n % 30) / 5 * 5);"
          , "  float t = float(n / 30);"
          , "  vec2 p;"
460       , "  p.x = float(n % 5);"
          , "  p.y = float(gl_VertexID % 2);"
          , "  if ((gl_VertexID % 2) == 1 && (n % 5) == 4) { p.x = 3.0 - p.x; }"
```

```
    , "  if (v.w < 0.0) { p.y = 1.0 - p.y; }"
    , "  gl_Position = vec4(  (1.0 + p.x + 0.25 * (v.x - 0.5) + s + 0.5) / 32.0 * (v.w > 0.0 ? -1.0 : 1.0) + (v.w > 0.0 ? 1.0 : -1.0)"
    , "                     , ((p.y + 0.25 * (v.y - 0.5) + t / 0.5) / 32.0 - 0.5) * (16.0/9.0), 0.0, 1.0);"
    , "}"
    ]) $ \s -> with s $ \string -> glShaderSource vert 1 string nullPtr
  glCompileShader vert
  prog2 <- glCreateProgram
  glAttachShader prog2 vert
  glAttachShader prog2 frag
  glLinkProgram prog2
  glUseProgram prog2
  vbo <- with 0 $ \p -> glGenBuffers 1 p >> peek p
  glBindBuffer GL_ARRAY_BUFFER vbo
  glBufferData GL_ARRAY_BUFFER (fromIntegral $ 1000 * 2 * 4 * sizeOf (0 :: Float)) nullPtr GL_STREAM_DRAW
  vao2 <- with 0 $ \p -> glGenVertexArrays 1 p >> peek p
  glBindVertexArray vao2
  glVertexAttribPointer 0 4 GL_FLOAT GL_FALSE 0 nullPtr
  glEnableVertexAttribArray 0

  frag <- glCreateShader GL_FRAGMENT_SHADER
  withCString (unlines
    [ "#version 330 core"
    , "uniform sampler2D mask;"
    , "uniform ivec2 size;"
    , "layout(location = 0) out vec4 cout;"
    , "void main() {"
    , "  cout = texture(mask, vec2(0.0, 1.0) + vec2(1.0, -1.0) * gl_FragCoord.xy / vec2(size));"
    , "}"
    ]) $ \s -> with s $ \string -> glShaderSource frag 1 string nullPtr
  glCompileShader frag
  vert <- glCreateShader GL_VERTEX_SHADER
  withCString (unlines
    [ "#version 330 core"
    , "void main() {"
    , "  if (gl_VertexID == 0) gl_Position = vec4(-1.0, -1.0, 0.0, 1.0);"
    , "  if (gl_VertexID == 1) gl_Position = vec4(-1.0,  1.0, 0.0, 1.0);"
    , "  if (gl_VertexID == 2) gl_Position = vec4( 1.0, -1.0, 0.0, 1.0);"
    , "  if (gl_VertexID == 3) gl_Position = vec4( 1.0,  1.0, 0.0, 1.0);"
    , "}"
    ]) $ \s -> with s $ \string -> glShaderSource vert 1 string nullPtr
  glCompileShader vert
  prog3 <- glCreateProgram
  glAttachShader prog3 vert
  glAttachShader prog3 frag
  glLinkProgram prog3
  glUseProgram prog3
  flip glUniform1i 6 =<< (withCString "mask" $ glGetUniformLocation prog3)
  (\l -> glUniform2i l width height) =<< (withCString "size" $ glGetUniformLocation prog3)

  framesR <- newIORef 0
  let showScoreboard 0 _ = return ()
      showScoreboard n count = do
```

```
515             when record $ capturePPM width height
                modifyIORef framesR (+1)
                frames <- readIORef framesR
                when (record && frames == 9000) exitSuccess
               GLFW.swapBuffers window
520             glClearColor 0 0 0 1
                glClear (GL_COLOR_BUFFER_BIT + GL_DEPTH_BUFFER_BIT)
                glUseProgram prog2
                glBindVertexArray vao2
                glDrawArrays GL_LINES 0 (count * 2)
525             glUseProgram prog3
                glBindVertexArray vao
                glEnable GL_BLEND
                glBlendFunc GL_SRC_ALPHA GL_ONE_MINUS_SRC_ALPHA
                glDrawArrays GL_TRIANGLE_STRIP 0 4
530             glDisable GL_BLEND
               GLFW.pollEvents
                stop <- GLFW.windowShouldClose window
                e <- glGetError
                when (e /= 0) $ hPutStrLn stderr $ "ERROR   " ++ show e
535             unless stop $ showScoreboard (n - 1) count

            uploadScoreboard scoreboard = do
              withArray (concat
                [ map realToFrac [ x0, y0, z0, w, x1, y1, z1, w ]
540             | ((LearningParameters x0 y0 z0, LearningParameters x1 y1 z1), w)
                  <- (lefts  scoreboard `zip` [-1,-2..]) ++
                     (rights scoreboard `zip` [1..])
                ] :: [Float]) $
                glBufferSubData GL_ARRAY_BUFFER 0 (fromIntegral $ fromIntegral (↙
                    ↳ length scoreboard) * 2 * 4 * sizeOf (0 :: Float)) . castPtr
545
            loop n m history scoreboard lastTime = do
              s@(ego, id, e0, i0, learnE, learnI) <- case m of
                Nothing -> random
                Just s -> pure s
550             a <- async $ do gan (learnE, learnI) (ego, id)
                when record $ capturePPM width height
                modifyIORef framesR (+1)
                frames <- readIORef framesR
                when (record && frames == 9000) exitSuccess
555             GLFW.swapBuffers window
                ((newEgo, newId), scores@(real:fake:poop:total:_)) <- wait a
                let s' = (newEgo, newId, e1, i1, learnE, learnI)
                    history' = drop 3 history ++ [ real, fake, poop ]
                    px :: S ('D3 143 1 3)
560                 px = S3D . LAS.fromList $ history'
                    !e1 = reflE newEgo
                    !i1 = reflI newId
                    !de = e1 - e0
                    !di = i1 - i0
565             glActiveTexture GL_TEXTURE0
                V.unsafeWith (image  de) $ glTexSubImage2D GL_TEXTURE_2D 0 0 0 36 64↙
                    ↳ GL_RGB GL_UNSIGNED_BYTE . castPtr
                glActiveTexture GL_TEXTURE1
                V.unsafeWith (image  di) $ glTexSubImage2D GL_TEXTURE_2D 0 0 0 55 64↙
                    ↳ GL_RGB GL_UNSIGNED_BYTE . castPtr
```

```
                glActiveTexture GL_TEXTURE2
570             V.unsafeWith (image  e1) $ glTexSubImage2D GL_TEXTURE_2D 0 0 0 36 64↙
                    ↳ GL_RGB GL_UNSIGNED_BYTE . castPtr
                glActiveTexture GL_TEXTURE3
                V.unsafeWith (image  i1) $ glTexSubImage2D GL_TEXTURE_2D 0 0 0 55 64↙
                    ↳ GL_RGB GL_UNSIGNED_BYTE . castPtr
                glActiveTexture GL_TEXTURE4
                V.unsafeWith (image' px) $ glTexSubImage2D GL_TEXTURE_2D 0 0 0 143 1↙
                    ↳ GL_RGB GL_UNSIGNED_BYTE . castPtr
575             glClearColor 0 0 0 1
                glClear (GL_COLOR_BUFFER_BIT + GL_DEPTH_BUFFER_BIT)
                glUseProgram prog
                glBindVertexArray vao
                glDrawArrays GL_TRIANGLE_STRIP 0 4
580             GLFW.pollEvents
                stop <- GLFW.windowShouldClose window
                e <- glGetError
                when (e /= 0) $ hPutStrLn stderr $ "ERROR   " ++ show e
                unless stop $ case () of
585               _ | total > 4.5 -> do
                        Just now <- GLFW.getTime
                        hPutStrLn stderr $ "SUCCESS " ++ show n
                        hPutStrLn stderr $ "FPS " ++ show (fromIntegral n / (now - ↙
                            ↳ lastTime))
                        hFlush stderr
590                     let scoreboard' = take 1024 $ Right (learnE, learnI) : ↙
                            ↳ scoreboard
                        uploadScoreboard scoreboard'
                        showScoreboard (15 * 5) (fromIntegral $ length scoreboard')
                        Just now <- GLFW.getTime
                        loop 0 Nothing (replicate (143 * 3) 0) scoreboard' now
595                   | n >= 999 -> do
                        Just now <- GLFW.getTime
                        hPutStrLn stderr $ "FAILURE " ++ show total
                        hPutStrLn stderr $ "FPS " ++ show (fromIntegral n / (now - ↙
                            ↳ lastTime))
                        hFlush stderr
600                     let scoreboard' = take 1024 $ Left (learnE, learnI) : ↙
                            ↳ scoreboard
                        uploadScoreboard scoreboard'
                        showScoreboard (15 * 5) (fromIntegral $ length scoreboard')
                        Just now <- GLFW.getTime
                        loop 0 Nothing (replicate (143 * 3) 0) scoreboard' now
605                   | otherwise -> do
                        loop (n + 1) (Just s') history' scoreboard lastTime
          glClearColor 0 0 0 1
          glClear (GL_COLOR_BUFFER_BIT + GL_DEPTH_BUFFER_BIT)
          Just now <- GLFW.getTime
610       loop 0 Nothing (replicate (143 * 3) 0) [] now

showN :: Show a => Char -> Int -> a -> String
showN c n = reverse . take n . (++ repeat c) . reverse . show

615 capturePPM :: Int -> Int -> IO ()
capturePPM vw vh = do
  let p6 = "P6\n" ++ show vw ++ " " ++ show vh ++ "\n255\n"
  allocaBytes (vw*vh*3) $ \ptr -> do
```

```
    glReadPixels 0 0 (fromIntegral vw) (fromIntegral vh) GL_RGB GL_UNSIGNED_BYTE↙
      ↳ (castPtr ptr)
    hPutStr stdout p6
    forM_ [0..vh-1] $ \y -> hPutBuf stdout (ptr `plusPtr` ((vh-1-y)*vw*3)) (vw↙
      ↳ *3)
    hFlush stdout

class Concrete a where
  reflect :: a -> Vector Double
  reify   :: [Double] -> Maybe (a, [Double])

instance Concrete (Pad a b c d) where
  reflect _ = V.empty
  reify xs = Just (Pad, xs)

instance Concrete (Crop a b c d) where
  reflect _ = V.empty
  reify xs = Just (Crop, xs)

instance Concrete (Pooling a b c d) where
  reflect _ = V.empty
  reify xs = Just (Pooling, xs)

instance Concrete Reshape where
  reflect _ = V.empty
  reify xs = Just (Reshape, xs)

instance Concrete Relu where
  reflect _ = V.empty
  reify xs = Just (Relu, xs)

instance Concrete Logit where
  reflect _ = V.empty
  reify xs = Just (Logit, xs)

instance ( KnownNat i
         , KnownNat o
         , KnownNat kx
         , KnownNat ky
         , KnownNat sx
         , KnownNat sy
         , KnownNat (kx * ky * i)
         ) => Concrete (Convolution i o kx ky sx sy) where
  reflect (Convolution ws _) = LA.flatten . extract $ ws
  reify xs = case splitAt n xs of
    (me, rest)
      | length me == n -> do
          ws <- create . LA.reshape o . LA.fromList $ me
          pure (Convolution ws (konst 0), rest)
    _ -> Nothing
    where
      n  = i * o * kx * ky
      i  = fromIntegral $ natVal (Proxy :: Proxy i)
      o  = fromIntegral $ natVal (Proxy :: Proxy o)
      kx = fromIntegral $ natVal (Proxy :: Proxy kx)
      ky = fromIntegral $ natVal (Proxy :: Proxy ky)
```

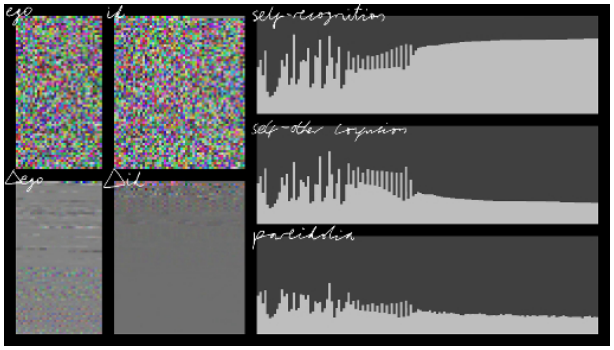
```
instance ( KnownNat i
         , KnownNat o
         , KnownNat kx
         , KnownNat ky
         , KnownNat sx
         , KnownNat sy
         , KnownNat (kx * ky * o)
         ) => Concrete (Deconvolution i o kx ky sx sy) where
  reflect (Deconvolution ws _) = LA.flatten . extract $ ws
  reify xs = case splitAt n xs of
    (me, rest)
      | length me == n -> do
          ws <- create . LA.reshape i . LA.fromList $ me
          pure (Deconvolution ws (konst 0), rest)
    _ -> Nothing
    where
      n  = i * o * kx * ky
      i  = fromIntegral $ natVal (Proxy :: Proxy i)
      o  = fromIntegral $ natVal (Proxy :: Proxy o)
      kx = fromIntegral $ natVal (Proxy :: Proxy kx)
      ky = fromIntegral $ natVal (Proxy :: Proxy ky)

instance (KnownNat i, KnownNat o) => Concrete (FullyConnected i o) where
  reflect (FullyConnected (FullyConnected' bs ws) _) = extract bs <> (LA.flatten↙
      ↪ . extract) ws
  reify xs = case splitAt o xs of
    (me, ys)
      | length me == o -> do
          bs <- create . LA.fromList $ me
          case splitAt (i * o) ys of
            (me, rest)
              | length me == i * o -> do
                  ws <- create . LA.reshape i . LA.fromList $ me
                  pure (FullyConnected (FullyConnected' bs ws) (FullyConnected' ↙
                      ↪ (konst 0) (konst 0)), rest)
            _ -> Nothing
    _ -> Nothing
    where
      i = fromIntegral $ natVal (Proxy :: Proxy i)
      o = fromIntegral $ natVal (Proxy :: Proxy o)

instance SingI i => Concrete (Network '[] '[i]) where
  reflect NNil = V.empty
  reify xs = Just (NNil, xs)

instance (SingI i, SingI o, Layer x i o, Concrete x, Concrete (Network xs (o ': ↙
    ↪ rs))) => Concrete (Network (x ': xs) (i ': o ': rs)) where
  reflect (x :~> r) = reflect x <> reflect r
  reify xs = do
    (x, ys) <- reify xs
    (r, zs) <- reify ys
    pure (x :~> r, zs)
```

8 nnirror.jpg

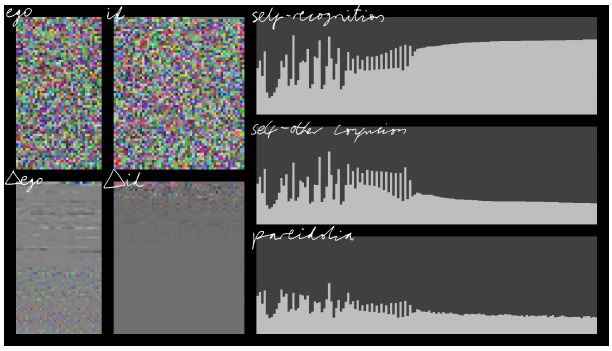


9 nnirror-plot.sh

```
#!/bin/bash
o=$1
mkdir -p "nnirror/${o}"
for i in 0 1 2 3 4 5 6 7 8 9
do
  shift
  j=$1
  rm "${i}.log"
  pushd "${j}"
  nnirror nets to images
  popd
  ln -s "${j}/nnirror.log" "${i}.log"
  convert "${j}.png" -rotate 270 -geometry 700x500 -quality 100 "nnirror/${o}/${i}-learn.jpg" &
  convert "${j}.png" -rotate 270                                 "nnirror/${o}/${i}-learn.png" &
  pnmenlarge 108 < "${j}/ego.ppm" | pnmpad -white -left 144 -right 144 -top 144 -bottom 144 > "${j}/ego_large.ppm"
  pnmenlarge 108 < "${j}/id.ppm"  | pnmpad -white -left 144 -right 144 -top 144 -bottom 144 > "${j}/id_large.ppm"
  convert "${j}/ego_large.ppm" -geometry 700x500 -quality 100 "nnirror/${o}/${i}-ego.jpg" &
  convert "${j}/ego_large.ppm"                                 "nnirror/${o}/${i}-ego.png" &
  convert "${j}/id_large.ppm"  -geometry 700x500 -quality 100 "nnirror/${o}/${i}-id.jpg" &
  convert "${j}/id_large.ppm"                                  "nnirror/${o}/${i}-id.png" &
  wait
  rm "${j}/ego_large.ppm" "${j}/id_large.ppm"
done
gnuplot < ../nnirror.gnuplot
convert "out.png" -geometry 700x500 -quality 100 "nnirror/${o}/metrics.jpg"
mv out.png "nnirror/${o}/metrics.png"
cat <<EOF > "nnirror/${o}/index.html"
<!DOCTYPE html>
<html><head><title>#${o} :: nnirror :: mathr</title><style>
body { width: 1500px; margin: auto; }
h1, h2, p { text-align: center; }
```

```
img { vertical-align: middle; }
a { text-decoration: none; }
a img { border: 1px solid; }
</style></head><body>
<h1>nnirror #${o}</h1>
<h2>metrics</h2><p><a href="metrics.png" title="metrics"><img src="metrics.jpg" alt="metrics" /></a></p>
EOF
for i in 0 1 2 3 4 5 6 7 8 9
do
  echo "<h2>${i}</h2><p><a href=\"${i}-learn.png\" title=\"${i} learn\"><img src=\"${i}-learn.jpg\" alt=\"${i} learn\" /></a> <a href=\"${i}-ego.png\" title=\"${i} ego\"><img src=\"${i}-ego.jpg\" alt=\"${i} ego\" /></a> <a href=\"${i}-id.png\" title=\"${i} id\"><img src=\"${i}-id.jpg\" alt=\"${i} id\" /></a></p>"
done >> "nnirror/${o}/index.html"
cat <<EOF >> "nnirror/${o}/index.html"
<hr />
<p><a href=".." title="nnirror">nnirror</a></p>
</body></html>
EOF
```

10 nnirror.png



11 nnirror-score.c

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

double u(double x)
{
  return -log(1 / x - 1);
}

int main(int argc, char **argv)
{
  if (argc != 4) return 1;
  double a = atof(argv[1]);
  double b = atof(argv[2]);
  double c = atof(argv[3]);
  double s = (2 * u(a) - u(b) - u(c)) / 4;
  printf("%f\n", s);
```

```
  return 0;
}
```

12 nnirror.sh

```
#!/bin/bash
nnirror1() {
  mkdir $1
  cd $1
  nnirror $1
  pnmtopng -compression 9 < nnirror.ppm > ../$1.png
  rm nnirror.ppm
  cd ..
}
export -f nnirror1
mkdir output &&
cd output &&
seq -w 0 999 |
parallel nnirror1 &&
for o in ???
do
  tail -n 1 $o/nnirror.log |
  ( read it a b c ; echo "$(../nnirror-score $a $b $c)" "$o" )
done |
sort -g -r |
cut -d\  -f 2 |
xargs -n 10 echo |
cat -n |
xargs -n 11 ../nnirror-plot.sh
cd nnirror
ls -1 |
sort -g |
while read n
do
  mv n $((n - 1))
done
rename s/^/0/ ?
cd ..
cd ..
```

13 scoreboard.png

14 Setup.hs

```
import Distribution.Simple
main = defaultMain
```

15 training.png