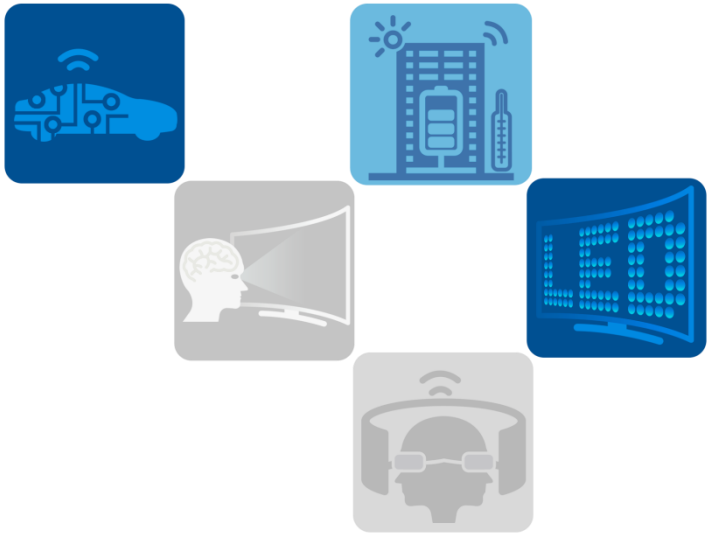
SAMSUNG

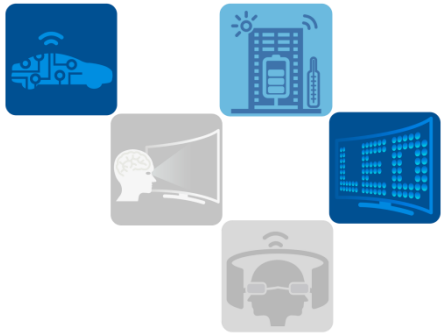


VELES Machine Learning Platform

Intelligent Data Processing Group,
Algorithm Lab, Samsung Research Russia (SRR)

Lyubov Podoynitsina

Table of Contents



- What is Veles?
- Modular structure
- Algorithms
- Easy to use
- Rapid training
- Distributed operations
- Export on device
- Applications
- Veles is Open Source
- Competitors
- Future directions

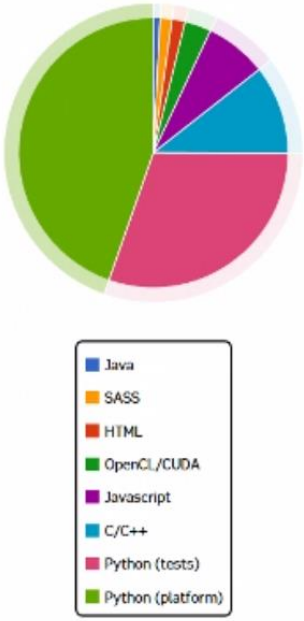
What is Veles?

VELES is a Machine Learning platform that

- provides machine learning and data processing services to users
- facilitates creation of applications by non-expert users
- was started in Mart 2013 by Samsung
- was released as Open Source in July 2015 by Samsung

Veles platform is built on top of many open source projects.

Lines of Code



If I have seen further it is by standing on the shoulders of giants.

— Isaac Newton

Veles key points

- VELES has modular structure
- Deep neural networks and Genetic algorithm
- It is easy to use
- Allows rapid development of ML applications
- Supporting of distributed operations
- Trained model can be exported on device

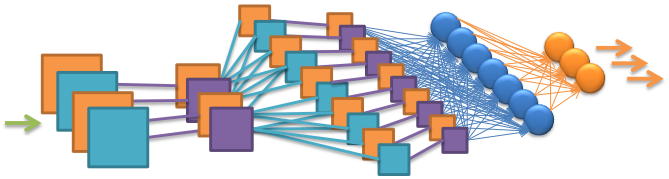
  (DTV, Web, Cloud, Mobile)

Modular structure

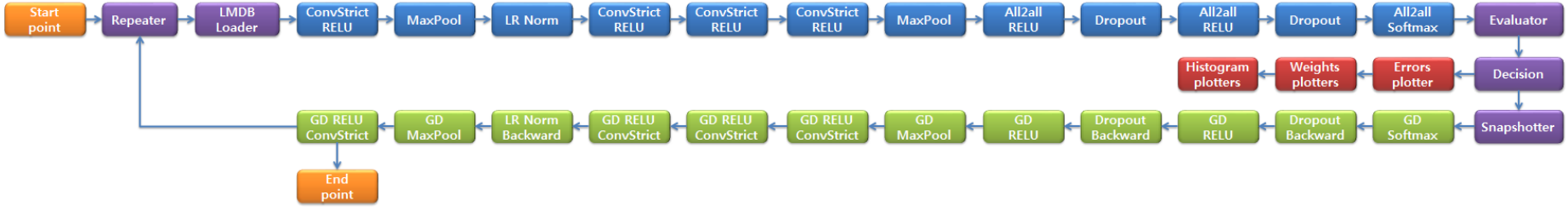
VELES uses modular paradigm for Quick and Easy Development of Machine Learning Algorithms and Models.

Building Blocks (Units)

Construction of Convolutional Neural Network



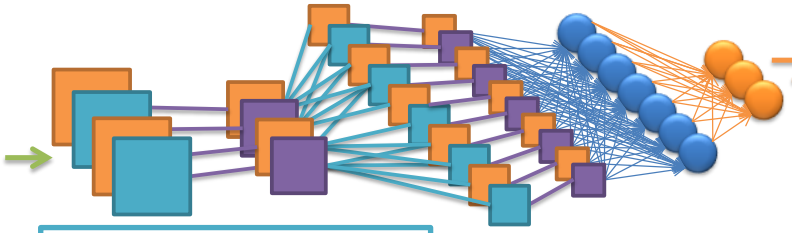
Model (Workflow)



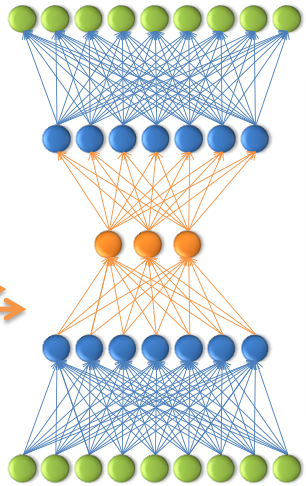
VELES includes ~225 predefined elementary building blocks - Units.
User can construct any dataflow algorithm including Neural Network model.

Veles algorithms

Deep Neural Networks



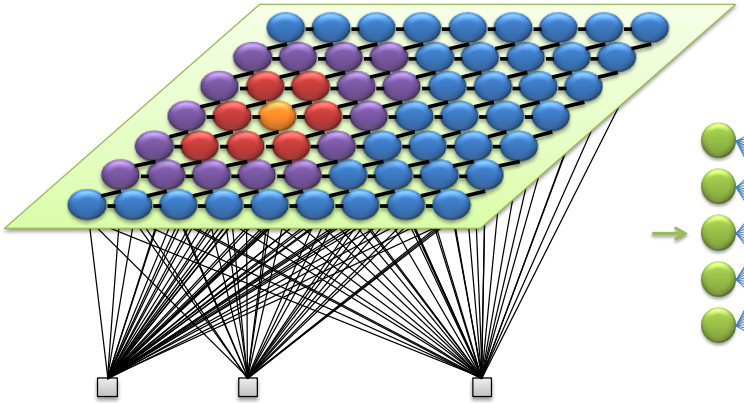
Convolutional NN



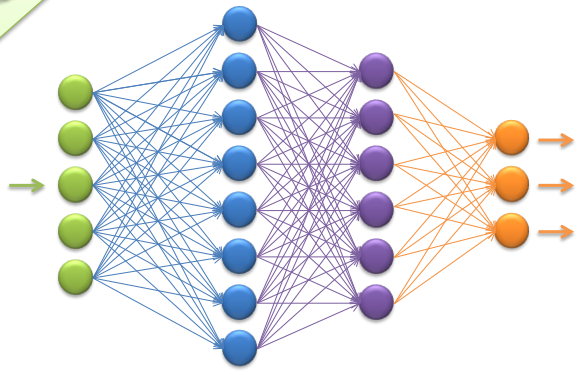
Autoencoder NN

Extras:

1. Deconvolution, Depooling
2. Dropout
3. Learning rate adjusting
4. Activation function customizati on
5. Regularization (L1, L2, custom)

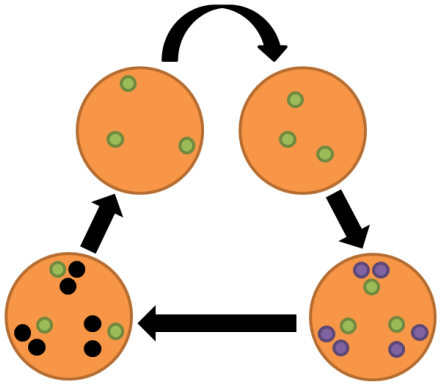


Kohonen maps



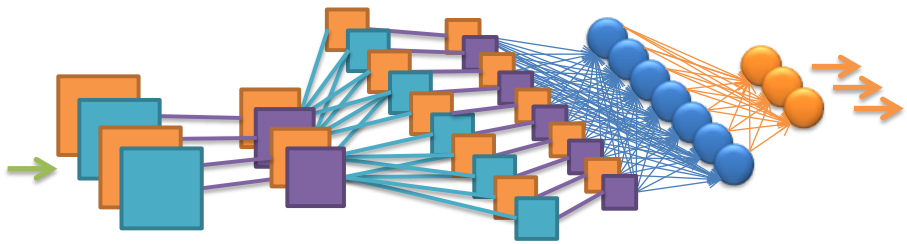
Fully connected NN

Genetic Algorithms



Veles algorithms

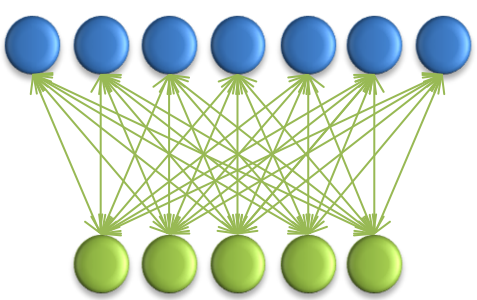
Deep Neural Networks



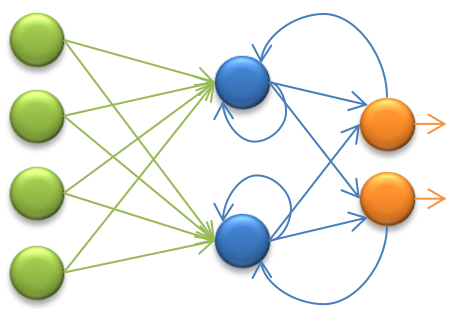
Convolutional Autoencoder NN
(with pretraining of each layer and fine tuning)

Extras:

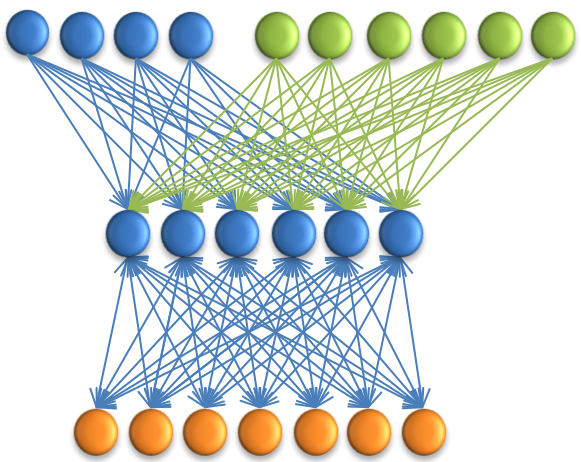
1. Last Models: AlexNet, VGG
2. Loss functions: mse, softmax
3. Stochastic gradient descent so lver with momentum
4. AdaGrad/AdaDelta solvers
5. Grouping



Restricted Boltzmann Machine (RBM)



Recurrent Neural Network (RNN)

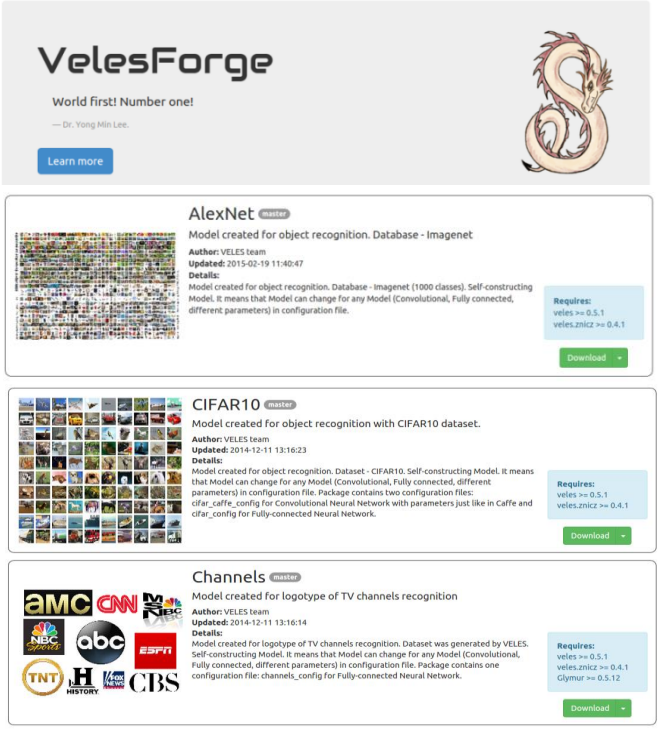


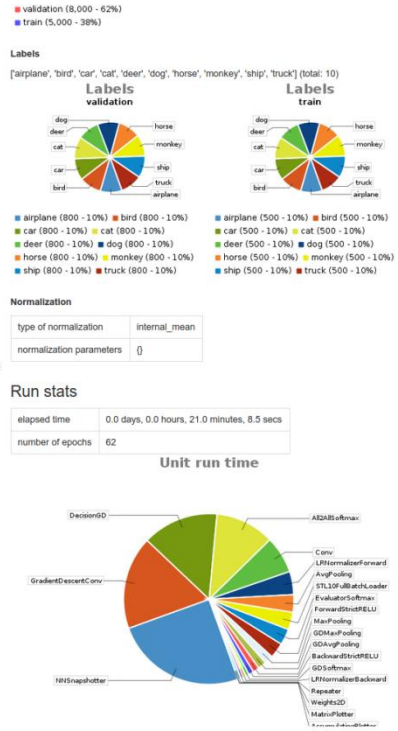
Long short-term memory (LSTM)

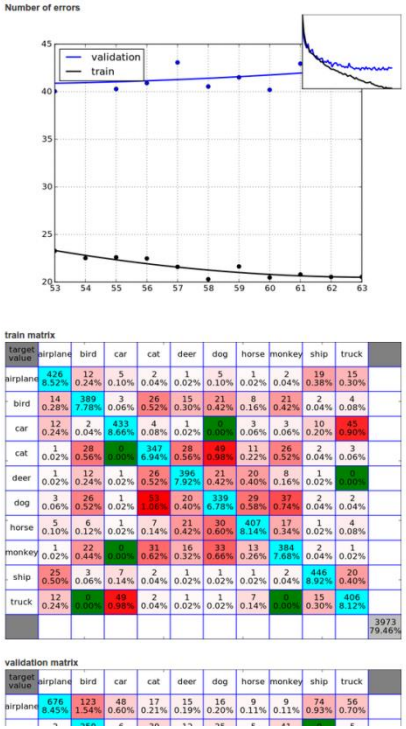
Easy to use

VELES provides different services to user to facilitate the application creation process, such as:

- Allows using predefined models and creation of model without writing any code
- Provides plotting service for debugging and control
- Can be used with Ipython
- Can automatically select best parameters for algorithm
- Provides publishing of the results
- …







Rapid training

- VELES uses optimization with CUDA or OPENCL backends on GPU and with Numpy on CPU to speed up calculations.
- Backend can be set for the entire model and for a particular unit.
- All backends have the same interface and produce the same results as the calculation of single and double precision.

python

```
...
self.execute_kernel(self._global_size_bias,
    self._local_size_bias, self._krn_bias_)
...
```

cuda4py
opencl4py

Execute CUDA/OPENCL on GPU

NVIDIA CUDA

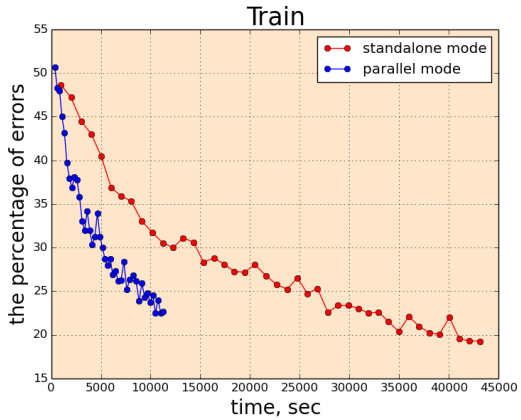
OpenCL

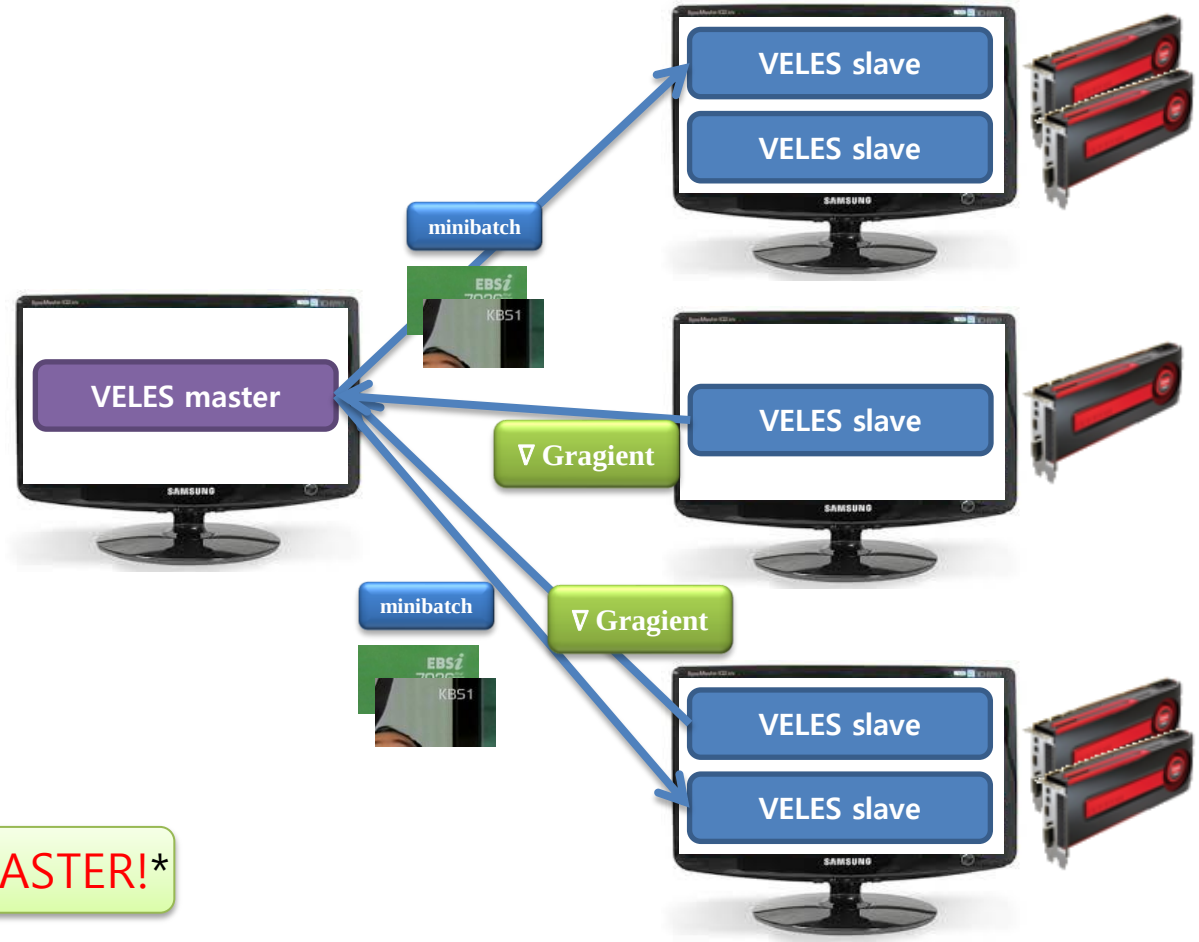
```
...
__global__ void Unpack1D(const dtype *data, dtype *unpack_data, const int l
imit) {
...
```

Distributed operations and deployment

- Any user can run the Model using parallel mode on Cluster
- VELES has Universal system of deployment on Cluster or User's computer.
- You can install VELES with one command.

# of nodes	Speedup
1	1
8	6.63
20	16.575 (Expected)
40	33.15 (Expected)

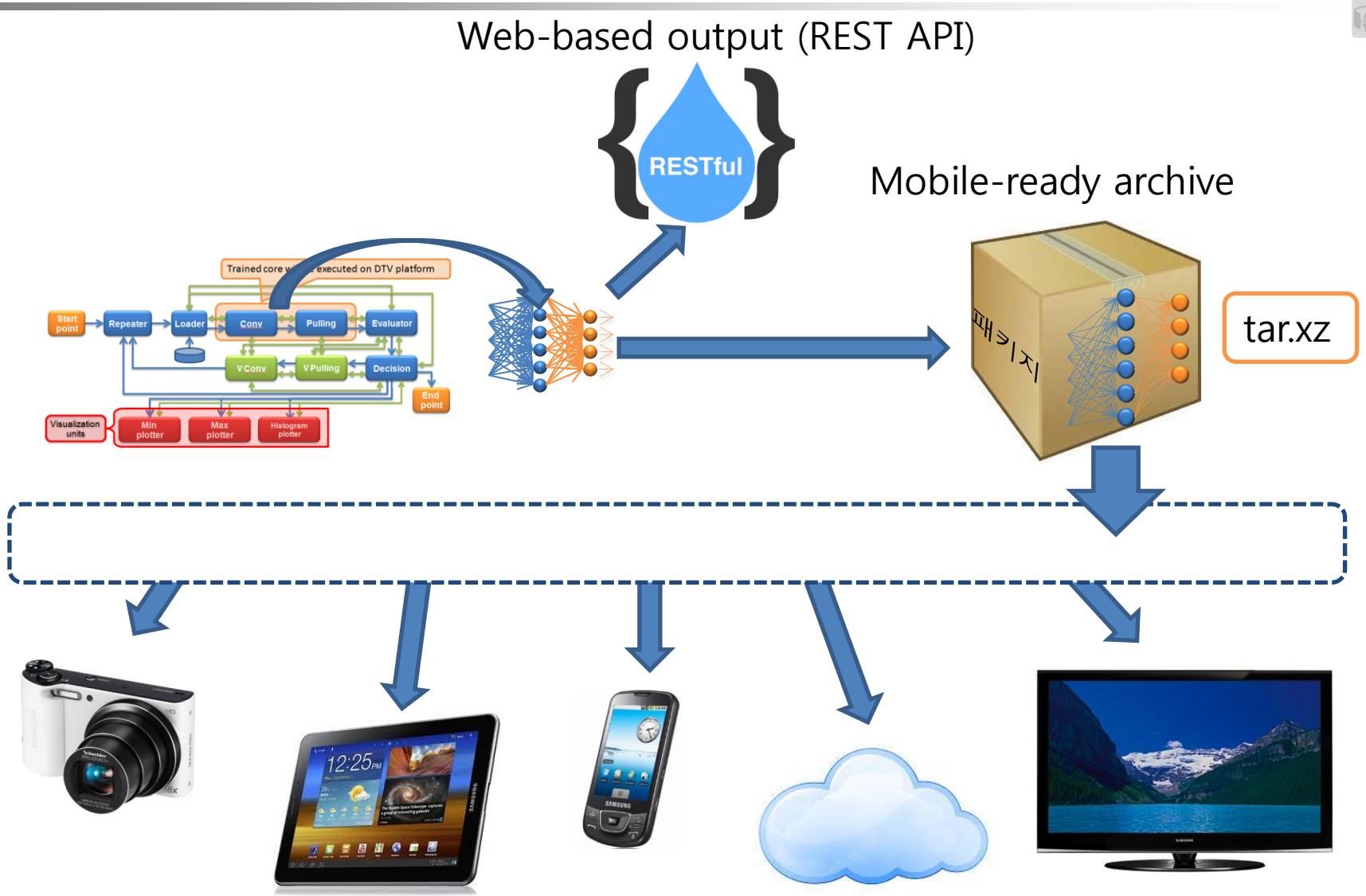




On 4 nodes ➡ 3-4 times FASTER!*

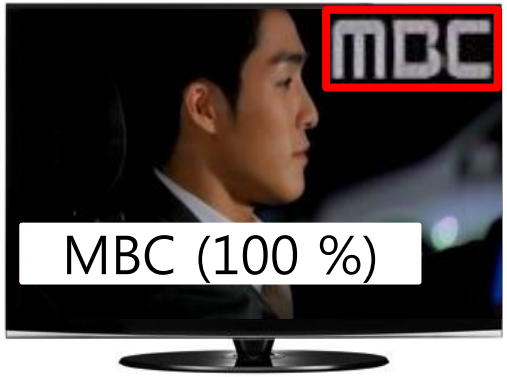
* Model dependent

Export on device



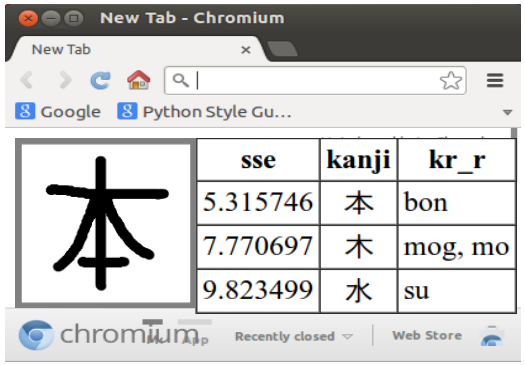
VELES can export trained core of Model to tar archive. Exported core can be executed on devices, Cloud or Web as application or service (tested on DTV, Web and Mobile).

Veles applications

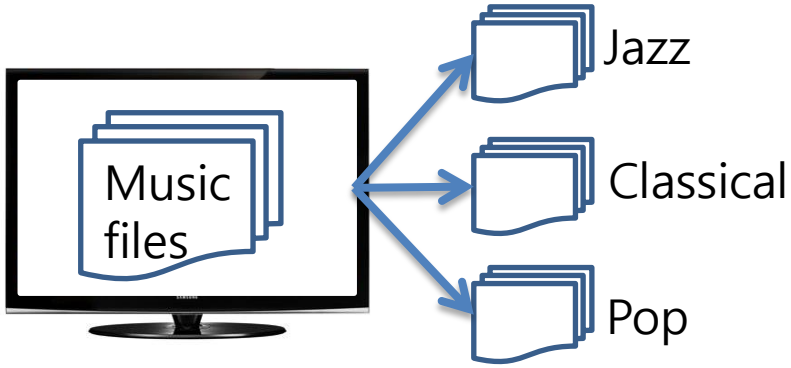


Channel logo recognition

Mobile user profiling

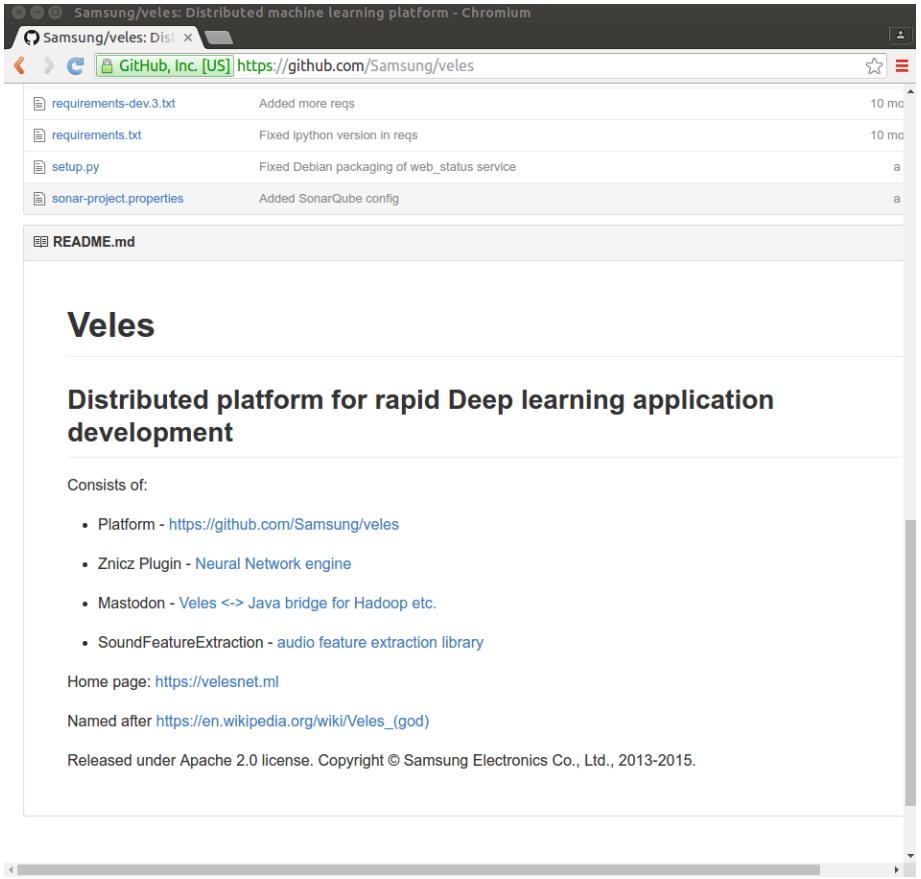


Character recognition



Music genre recognition

Veles is Open Source



Competitors

Specific focus

CAFFE

Berkeley
UNIVERSITY OF CALIFORNIA

specified for images recognition

TENSOR FLOW

Google

specified for large scaled applications

VELES

SAMSUNG

TORCH

facebook

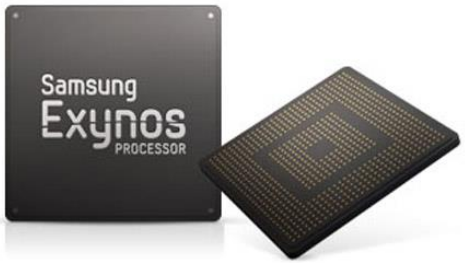
DSSTNE

amazon

Deep Learning library produced by Amazon
DSSTNE

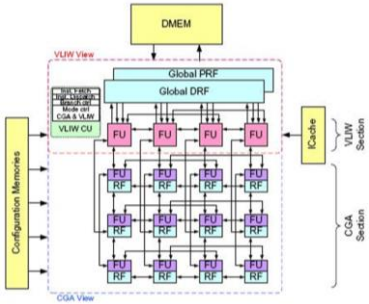
specified for sparse calculations and amazon services

Future directions

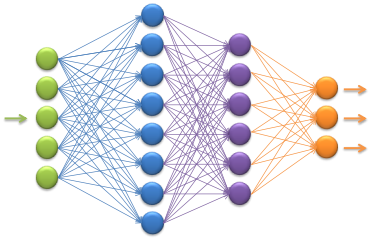


Mali

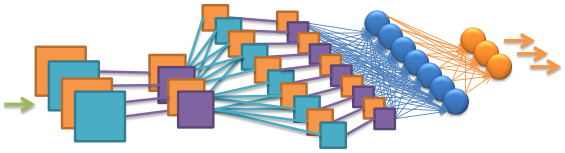
bigLITTLE



SRP

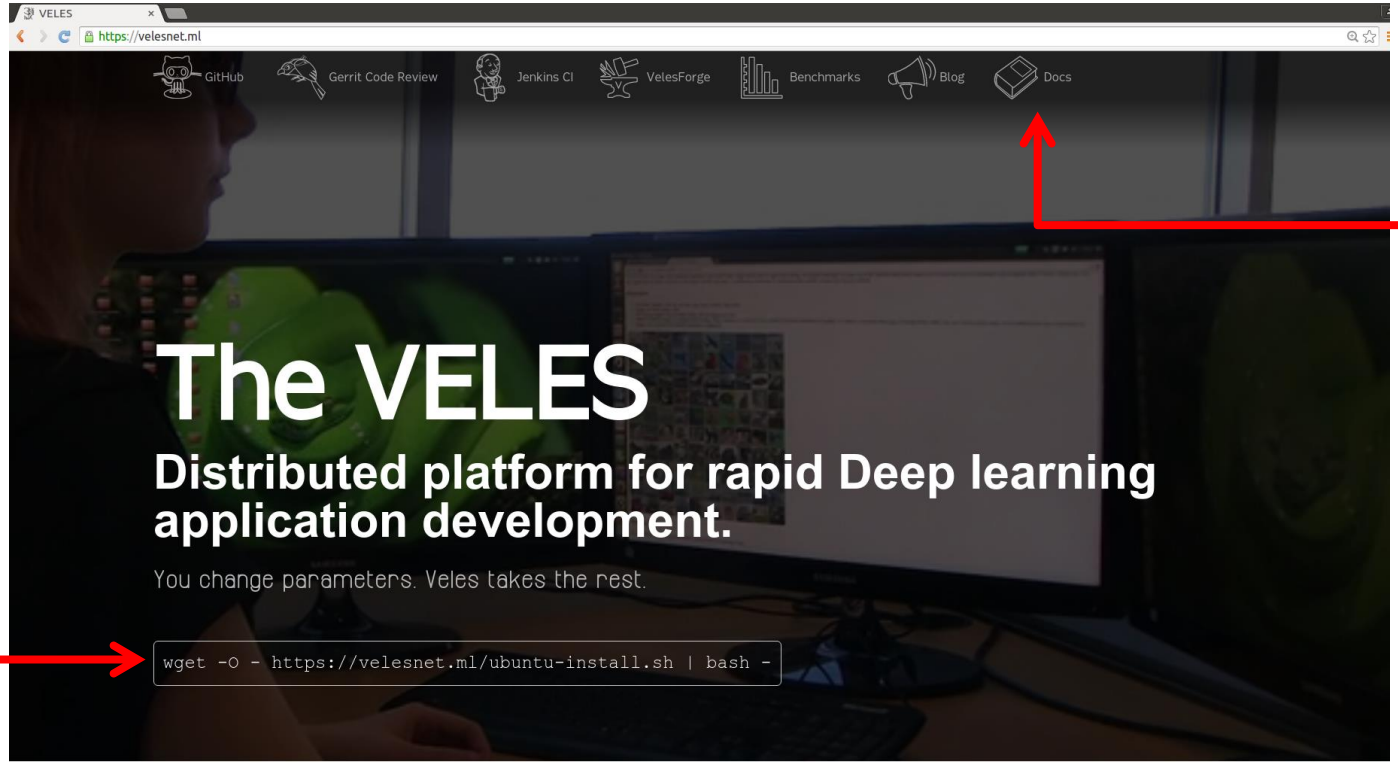


Automatic support of large deep machine learning algorithms on Samsung devices



Thank you!

Contacts



docs

install

https://velesnet.ml/

https://github.com/Samsung/veles

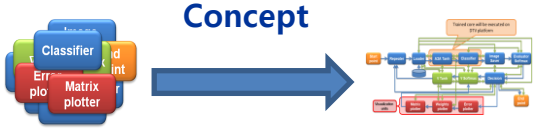
podoynitsinalv@gmail.com

Appendix

Veles history

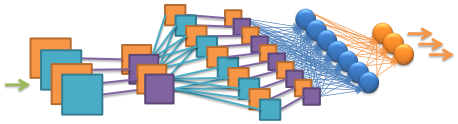
Veles ® in 2013

Implemented Building Blocks Concept



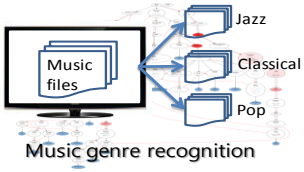
84 high performance Units

Neural Networks: fully-connected, convolutional, encoder, auto-encoder

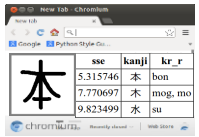


10 different Models

4 different recognition applications developed

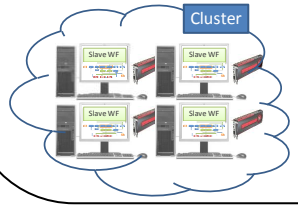


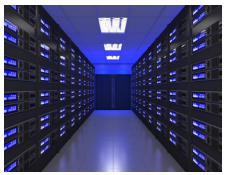
Music genre recognition



Japanese character recognition

Distributed Training on Cluster: Data Parallel Mode





Veles ® in 2014

New ML Algorithms: RBM, Kohonen map, Convolutional Autoencoder, Genetic optimization

28 predefined Models
225 high performance Units

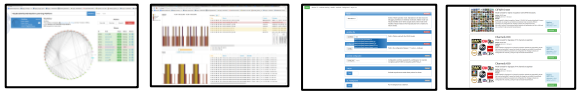
Hadoop Integration with GPU HDFS data Loader

VELES Forge: Services where User can Download Models

Easy Deployment on Cluster

ubuntu amazon CentOS

Web Frontend and GUI for Training & Introspection

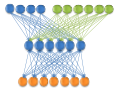


Supports Different Backbends: OpenCL, CUDA, CuBlas, Numpy

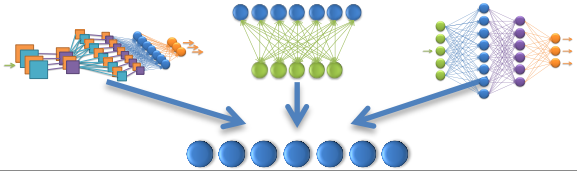
Models Development is even quicker and easier, than before

Veles ® in 2015

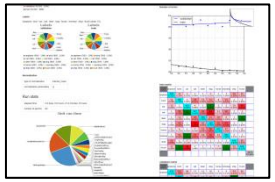
New ML Algorithms: RNN, LSTM



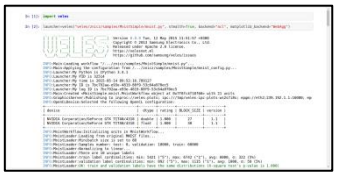
Ensembles implementation



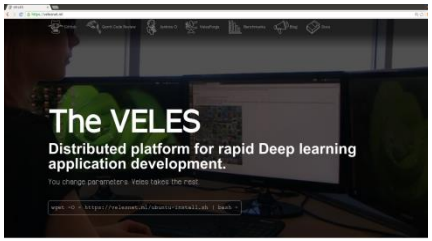
Publishing service (Markdown/HTML/ PDF/Confluence backbends)



Veles with IPython



Veles is Open Source!



Comparison table

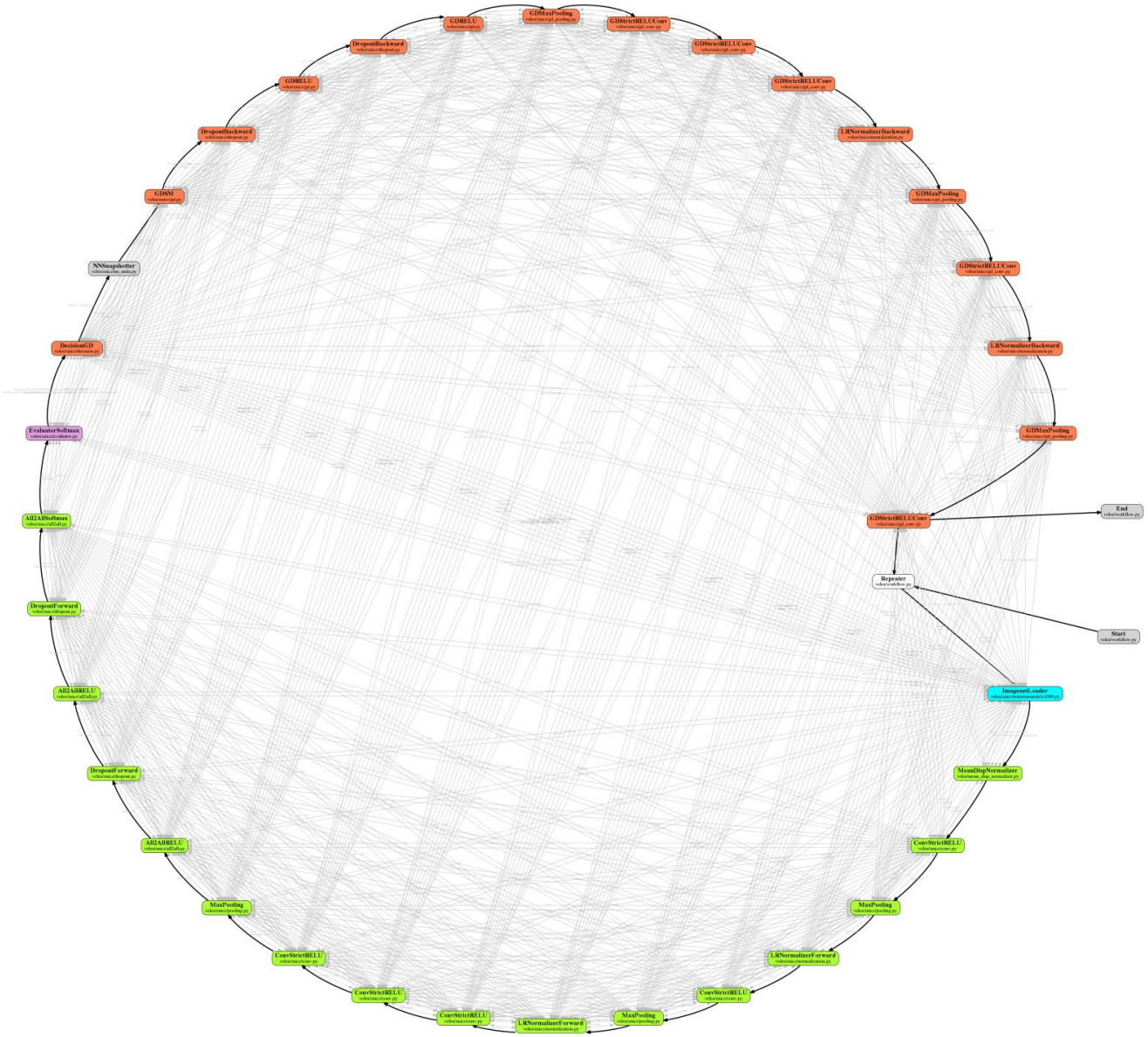
	VELES	TensorFlow	Torch, Caffe
Data preprocessing/collection	Dataset Analysis & Validation Tool; Collaborative Image Marking Tool	N/A *Requires separate tools	N/A *Requires separate tools
Distributed training	Multi nodes, multi devices	Single node, multi devices	Single node, single device
Model analysis	Advanced (Web-Based Graphics, Report Generator)	Advanced (Graphical display, graph visualization)	Basic (Error rate, etc)
Production	Web-based output (REST API) or Mobile-ready archive (JNI)	Model graph protobuf (JNI)	N/A *Requires separate tools
Languages	Python2.7, Python3.4	C++, Python2.7	C, C++
Backends	CUDA, OpenCL, Numpy	CuDNN, BLAS	CuDNN, CUDA
GUI	Web Status dashboard, Front end, Forge, Plots, Logging/Events status	Graph Visualization, Graphical Display, Visualization of Inception training	N/A *Requires separate tools
Data/control flow paradigm	+	+	-
Benchmarks (AlexNet, 1 mini batch 128x3x224x224)	344 msec	507 msec	360 msec (R2), 149 (R3)

Ipython

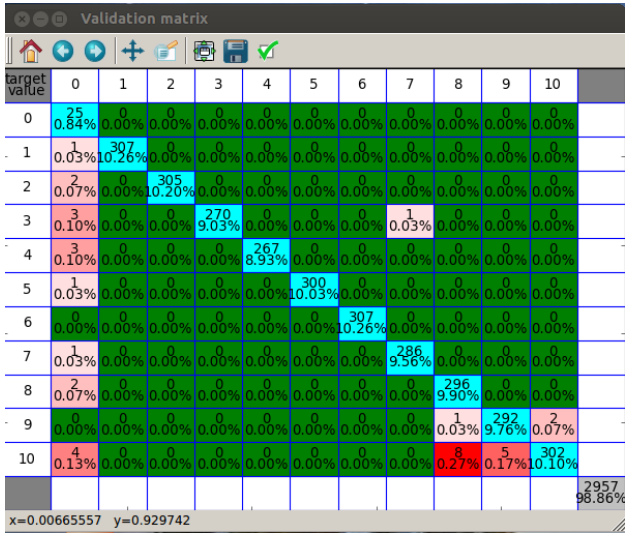
```
In [1]: import veles

In [2]: launcher=veles("veles/znicz/samples/MnistSimple/mnist.py", stealth=True, backend="ocl", matplotlib_backend="WebAgg")
```

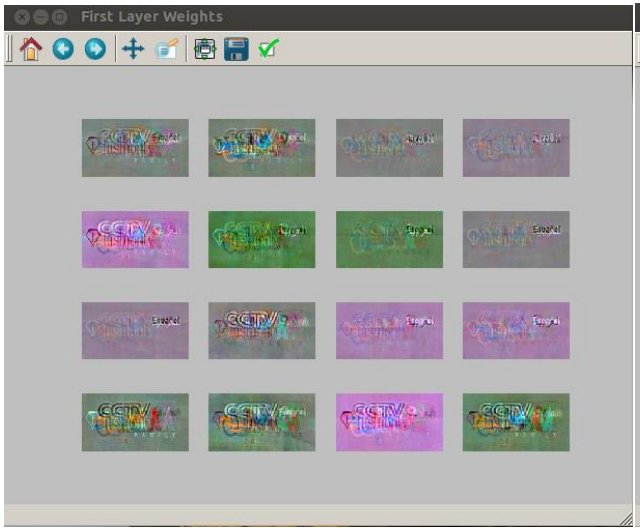
```
 _   _ _____ _     _____ _____
| | | |  ___| |   |  ___/  ___|  Version 0.8.9 Tue, 12 May 2015 11:51:57 +0300
| | | | |__ | |   | |__ \ `--.   Copyright © 2013 Samsung Electronics Co., Ltd.
| | | |  __|| |   |  __| `--. \  Released under Apache 2.0 license.
\ \_/ / |___| |___| |___/\__/ /  https://velesnet.ml
 \___/\____/\_____|____/\____/   https://github.com/samsung/veles/issues

INFO:Main:Loading workflow "/.../znicz/samples/MnistSimple/mnist.py"...
INFO:Main:Applying the configuration from /.../znicz/samples/MnistSimple/mnist_config.py...
INFO:Launcher:My Python is CPython 3.4.1
INFO:Launcher:My PID is 32514
INFO:Launcher:My time is 2015-05-14 09:52:16.789127
INFO:Launcher:My ID is 7bc792aa-e93e-4819-88f9-53c64a970ec5
INFO:Launcher:My log ID is 7bc792aa-e93e-4819-88f9-53c64a970ec5
INFO:Main:Created <MnistSimple.mnist.MnistWorkflow object at 0x7f8fc4f18f60> with 21 units
INFO:GraphicsServer:Publishing to inproc://veles-plots; ipc:///tmp/veles-ipc-plots-ws2s7i8s; epgm://eth2;239.192.1.1:16080; ep
INFO:OpenCLDevice:Selected the following OpenCL configuration:
+-------------------------------------------+--------+--------+------------+---------+
| device                                    |  dtype | rating | BLOCK_SIZE | version |
+-------------------------------------------+--------+--------+------------+---------+
| NVIDIA Corporation/GeForce GTX TITAN/4318 | double | 1.000  |     27     |   1.1   |
| NVIDIA Corporation/GeForce GTX TITAN/4318 | float  | 1.000  |     30     |   1.1   |
+-------------------------------------------+--------+--------+------------+---------+
INFO:MnistWorkflow:Initializing units in MnistWorkflow...
INFO:MnistLoader:Loading from original MNIST files...
INFO:MnistLoader:Minibatch size is set to 88
INFO:MnistLoader:Samples number: test: 0, validation: 10000, train: 60000
INFO:MnistLoader:Normalizing to linear...
INFO:MnistLoader:There are 10 unique labels
INFO:MnistLoader:train label cardinalities: min: 5421 ("5"), max: 6742 ("1"), avg: 6000, σ: 322 (5%)
INFO:MnistLoader:validation label cardinalities: min: 892 ("5"), max: 1135 ("1"), avg: 1000, σ: 59 (5%)
INFO:MnistLoader:OK: train and validation labels have the same distributions (X-square test's p-value is 1.000)
```

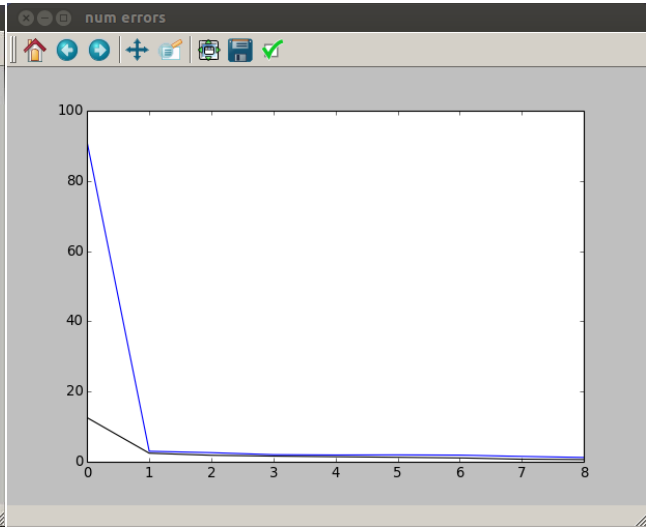
Workflow example

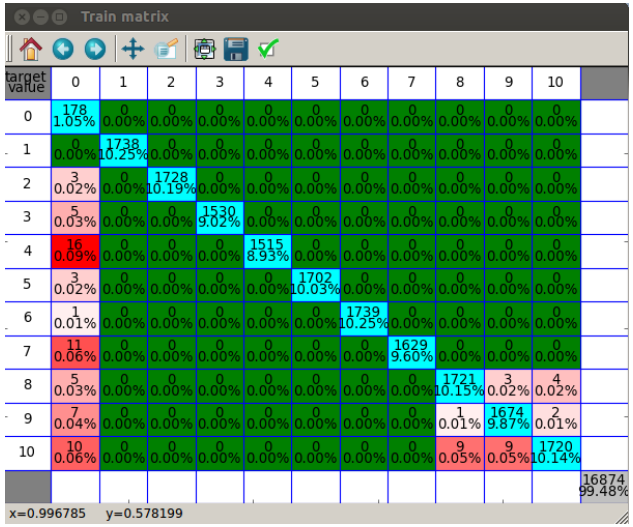


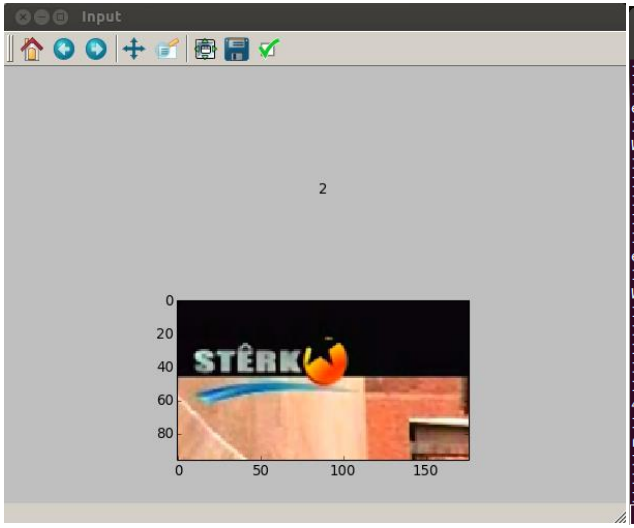
Plotters (1)

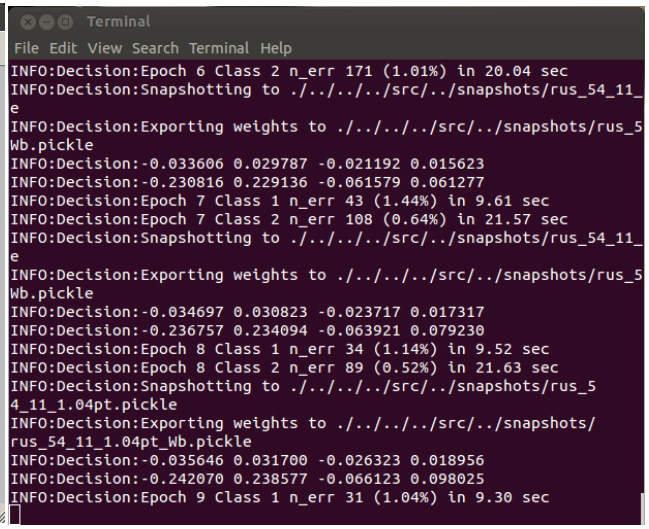




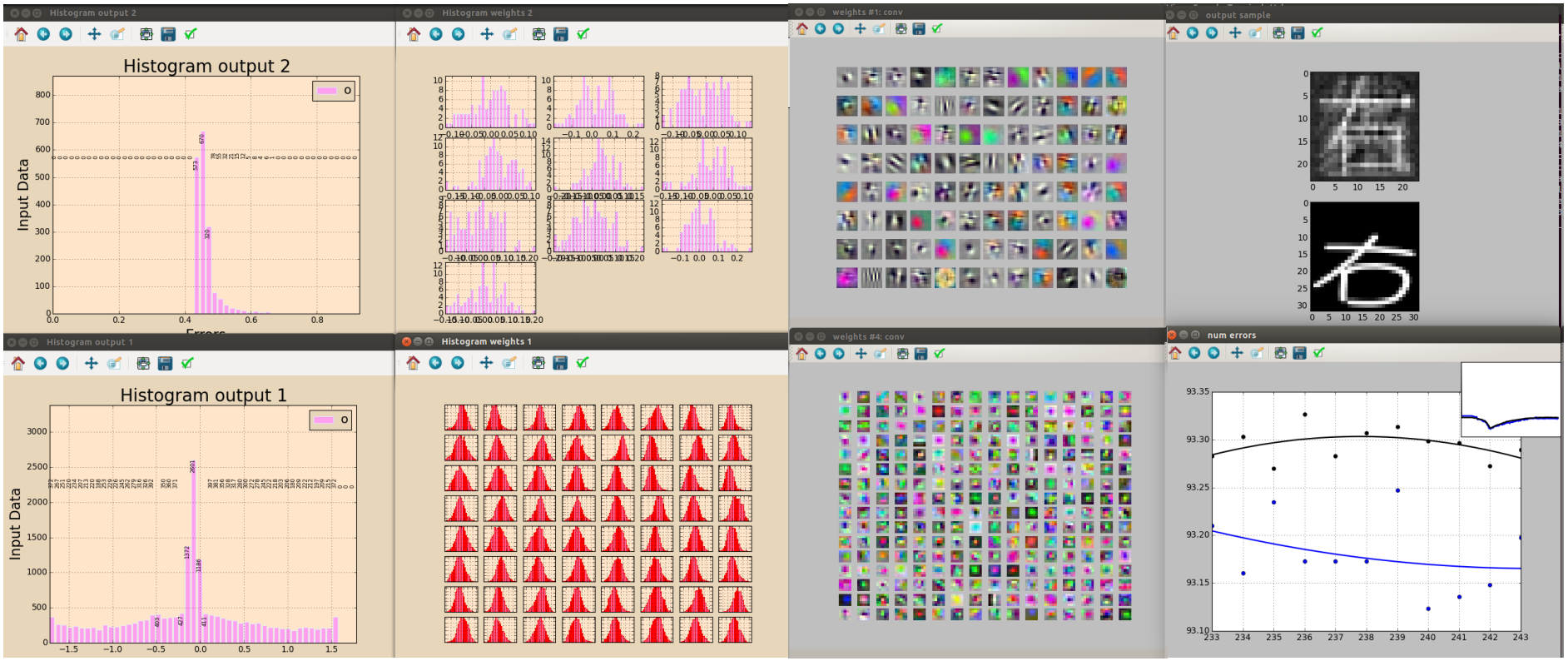




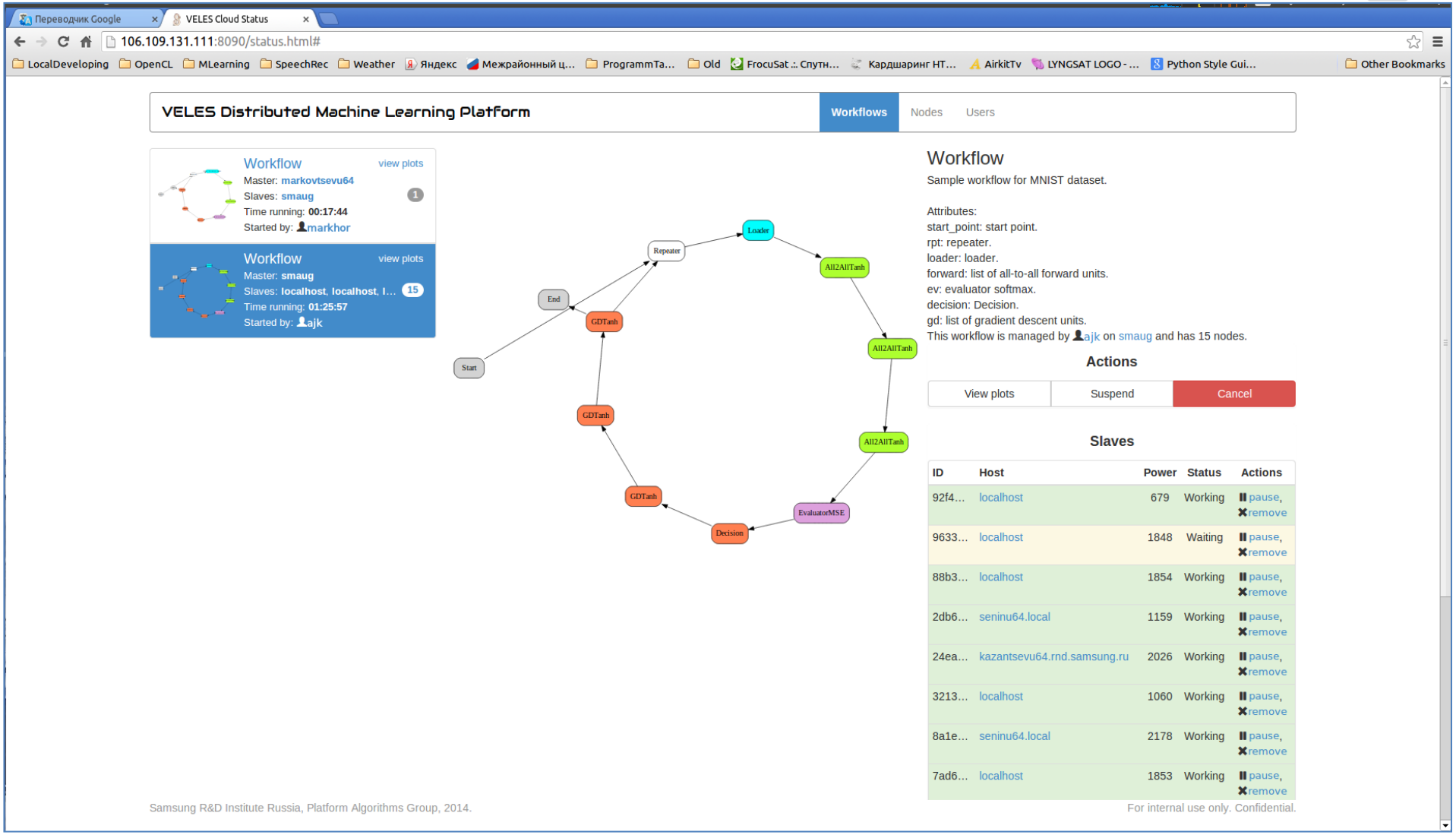




Plotters (2)


Histogram output 2
Input Data
Histogram weights 2
weights #1: conv
output sample
Histogram output 1
Input Data
Histogram weights 1
weights #4: conv
num errors

Web Service


Переводчик Google
VELES Cloud Status
106.109.131.111:8090/status.html#
LocalDeveloping
OpenCL
MLearning
SpeechRec
Weather
Яндекс
Межрайонный ц...
ProgrammTa...
Old
FrocuSat :: Спутн...
Кардшаринг HT...
AirkitTv
LYNGSAT LOGO - ...
Python Style Gui...
Other Bookmarks
VELES Distributed Machine Learning Platform
Workflows
Nodes
Users
Workflow
view plots
Master: markovtsevu64
Slaves: smaug
1
Time running: 00:17:44
Started by: markhor
Workflow
view plots
Master: smaug
Slaves: localhost, localhost, l...
15
Time running: 01:25:57
Started by: ajk
Loader
Repeater
All2AllTanh
End
GDTanh
All2AllTanh
Start
GDTanh
All2AllTanh
GDTanh
EvaluatorMSE
Decision
Workflow
Sample workflow for MNIST dataset.
Attributes:
start_point: start point.
rpt: repeater.
loader: loader.
forward: list of all-to-all forward units.
ev: evaluator softmax.
decision: Decision.
gd: list of gradient descent units.
This workflow is managed by ajk on smaug and has 15 nodes.
Actions
View plots
Suspend
Cancel
Slaves
ID Host Power Status Actions
92f4... localhost 679 Working pause, remove
9633... localhost 1848 Waiting pause, remove
88b3... localhost 1854 Working pause, remove
2db6... seninu64.local 1159 Working pause, remove
24ea... kazantsevu64.rnd.samsung.ru 2026 Working pause, remove
3213... localhost 1060 Working pause, remove
8a1e... seninu64.local 2178 Working pause, remove
7ad6... localhost 1853 Working pause, remove
Samsung R&D Institute Russia, Platform Algorithms Group, 2014.
For internal use only. Confidential.

Frontend

RUN --device 1:0 --verbose warning --stealth --manhole --background --async mn

Mode — Mandatory

Standalone

Selects VELES operation mode. Standalone is the best choice for debugging workflows and relatively small tasks. Master mode runs VELES workflow server which accepts slaves. Slave mode is the headless VELES instance which requests jobs from a master and returns results.

workflow file — Mandatory

workflow: mn

- veles/znicz/samples/**mn**ist.py
- veles/znicz/tests/research/**mn**ist.py
- veles/znicz/tests/research/**mn**ist7.py
- veles/znicz/tests/research/**mn**ist784.py
- veles/znicz/tests/research/**mn**ist_ae.py

Path to Python script with the VELES model.

configuratic — Mandatory

config: -

Path to the configuration file(pass "-" to set as _config.py).

override configuration — Optional

config_list: None

Configuration overrides separated by a whitespace, for example: root.global_alpha=0.006 root.snapshot_prefix='test_pr'.

--async — Optional

Yes

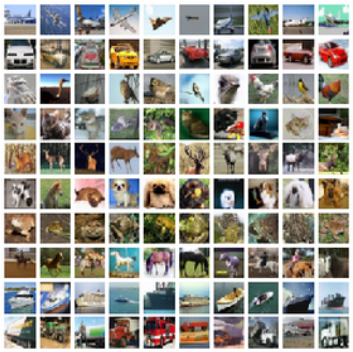
Activate asynchronous master-slave protocol on slaves.

-b, --background — Optional

Yes

Run in background as a daemon.

Forge



CIFAR10 master

Model created for object recognition with CIFAR10 dataset.

Author: VELES team
Updated: 2014-12-11 13:16:23
Details:
Model created for object recognition. Dataset - CIFAR10. Self-constructing Model. It means that Model can change for any Model (Convolutional, Fully connected, different parameters) in configuration file. Package contains two configuration files: cifar_caffe_config for Convolutional Neural Network with parameters just like in Caffe and cifar_config for Fully-connected Neural Network.

Requires:
veles >= 0.5.1
veles.znicz >= 0.4.1

Download

MnistAE master

Model created for digits recognition with MNIST Database by autoencoder.

Author: VELES team
Updated: 2014-12-11 13:16:29
Details:
Model created for digits recognition. Database - MNIST. Model - autoencoder.

Requires:
veles >= 0.5.1
veles.znicz >= 0.4.1

Download

Channels master

Model created for logotype of TV channels recognition

Author: VELES team
Updated: 2014-12-11 13:16:14
Details:
Model created for logotype of TV channels recognition. Dataset was generated by VELES. Self-constructing Model. It means that Model can change for any Model (Convolutional, Fully connected, different parameters) in configuration file. Package contains one configuration file: channels_config for Fully-connected Neural Network.

Requires:
veles >= 0.5.1
veles.znicz >= 0.4.1
Glymur >= 0.5.12

Download

Publishing

MnistWorkflow - Vel

confluence.rnd.samsung.ru/display/VEL/MnistWorkflow

Confluence Spaces People Calendars Create

Veles

Pages

Blog

SPACE SHORTCUTS

How-to articles

Troubleshooting articles

CHILD PAGES

Veles

MnistWorkflow

+ Create child page

Space tools

Pages / Veles

Edit Watch Share Tools

MnistWorkflow

Created and last modified by Markovtsev Vadim [X] on Apr 28, 2015

Task Description

Workflow for MNIST dataset (handwritten digits recognition).

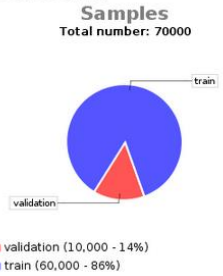
Workflow path
/home/markhor/Development/Veles/veles/znicz/samples/MnistSimple/mnist.py

Configuration path
/home/markhor/Development/Veles/veles/znicz/samples/MnistSimple/mnist_config.py

Results

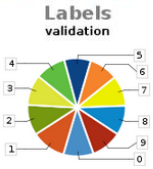
set	accuracy	errors
validation	92.64%	7.36%
train	91.61%	8.39%

Source data

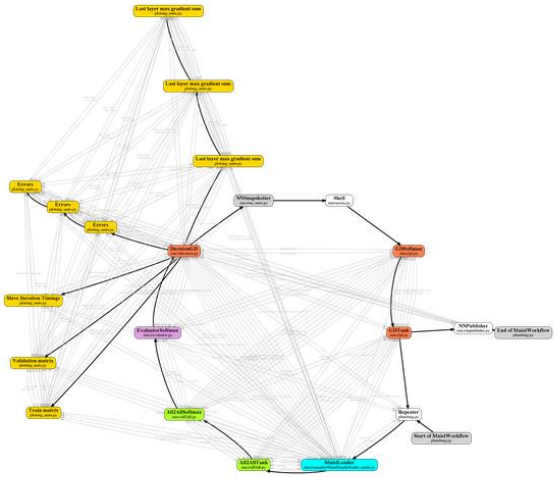


Labels

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9] (total: 10)

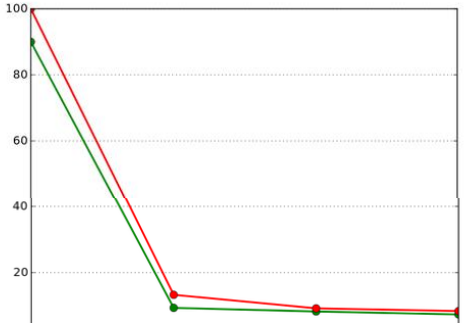


Workflow

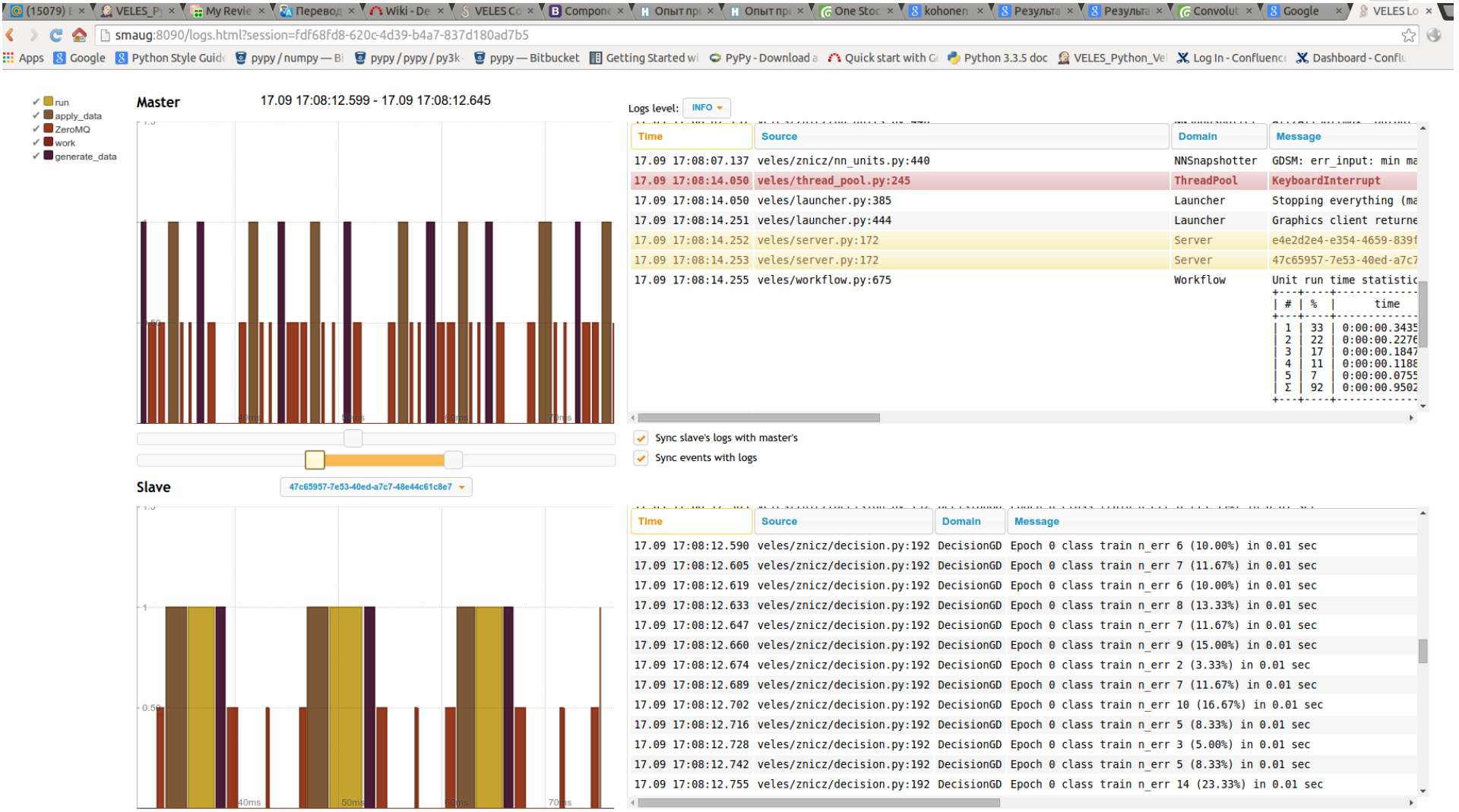


Plots

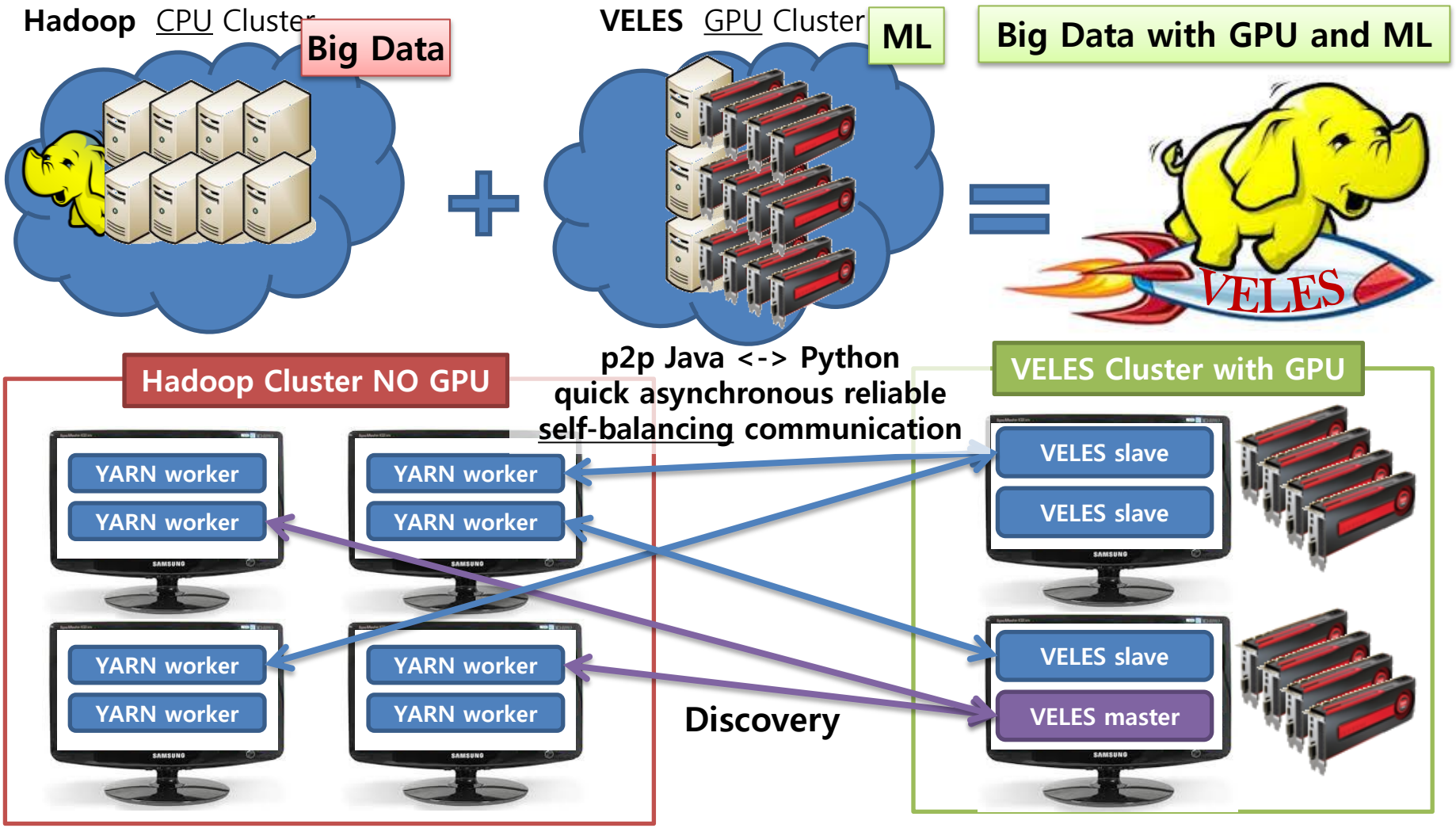
Errors



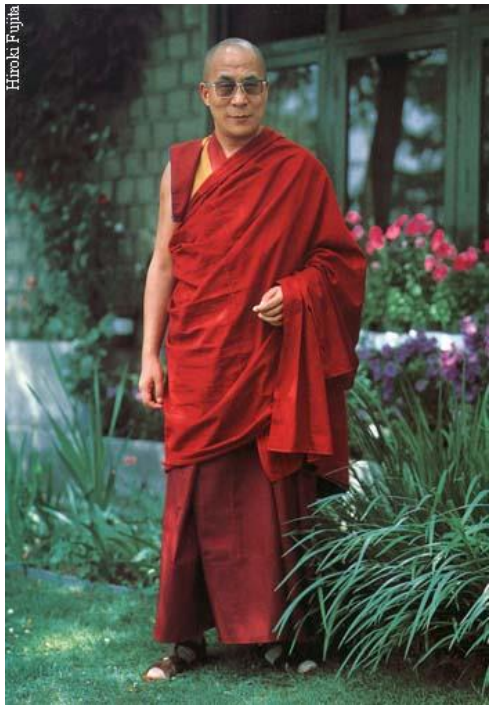
Logging Service



Veles + Hadoop



Imagenet



label	value
cloak	0.192772
trench coat	0.159061
stole	0.116678
kimono	0.0757744
sarong	0.0683556

label	value
Labrador retriever	0.530035
golden retriever	0.4135
vizsla, Hungarian pointer	0.0154826
cocker spaniel, English cocker spaniel, cocker	0.0152616
Rhodesian ridgeback	0.00767738

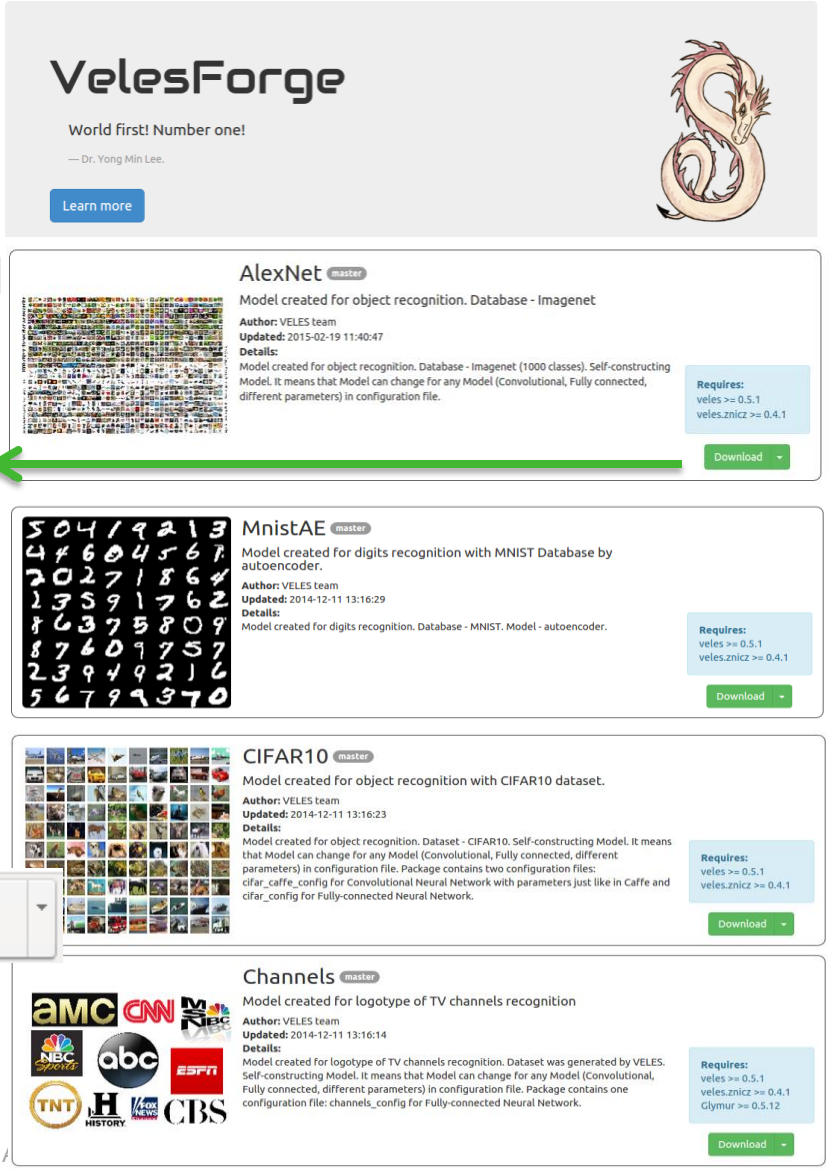
label	value
black grouse	0.998416
cock	0.000364259
partridge	0.000329289
hen	0.000294202
vulture	0.000238185

label	value
Siamese cat, Siamese	0.995305
Chihuahua	0.00133562
Persian cat	0.00114584
Angora, Angora rabbit	0.000722808
black-footed ferret, ferret, Mustela nigripes	0.000440669

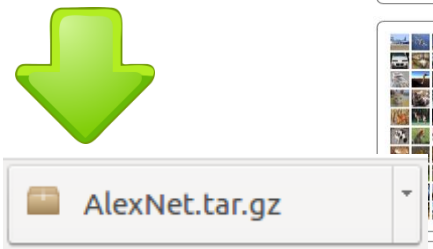
VelesForge

1. Download Model from Web Service VELESForge.

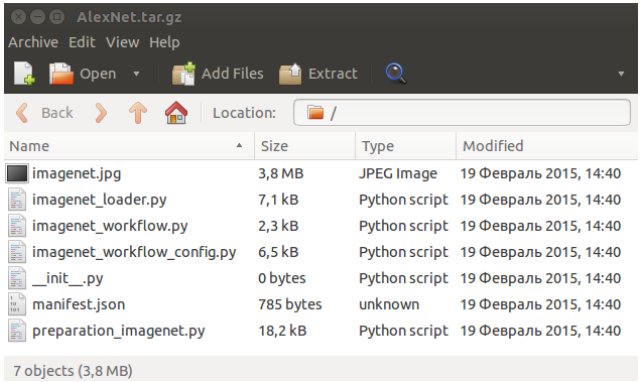
Alex, entry level



Download



2. Open downloaded folder.



Downloaded folder contains workflow, loader, configuration file and snapshot of trained Model. Alex can use snapshot of the Model right away or can train Model from scratch.

Deployment

VELES has Universal system of deployment on Cluster or User's computer.
You can install VELES with one command.

sudo apt-get install python3-veles

git

script

tar.xz - Universal
deb – Debian/Ubuntu
rpm – Centos/RHEL/F
edora

script

Cluster

script

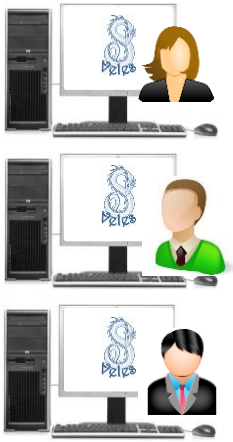
User's PC

Tested on Amazon, Ubuntu and Centos.

amazon

ubuntu

CentOS

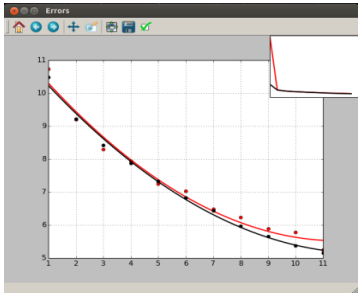


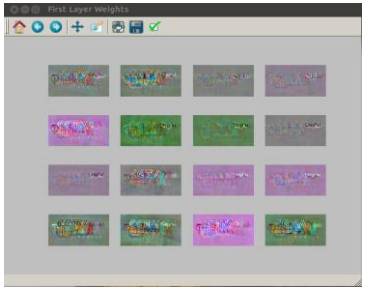
Graphics & Web Service

Alex, entry level

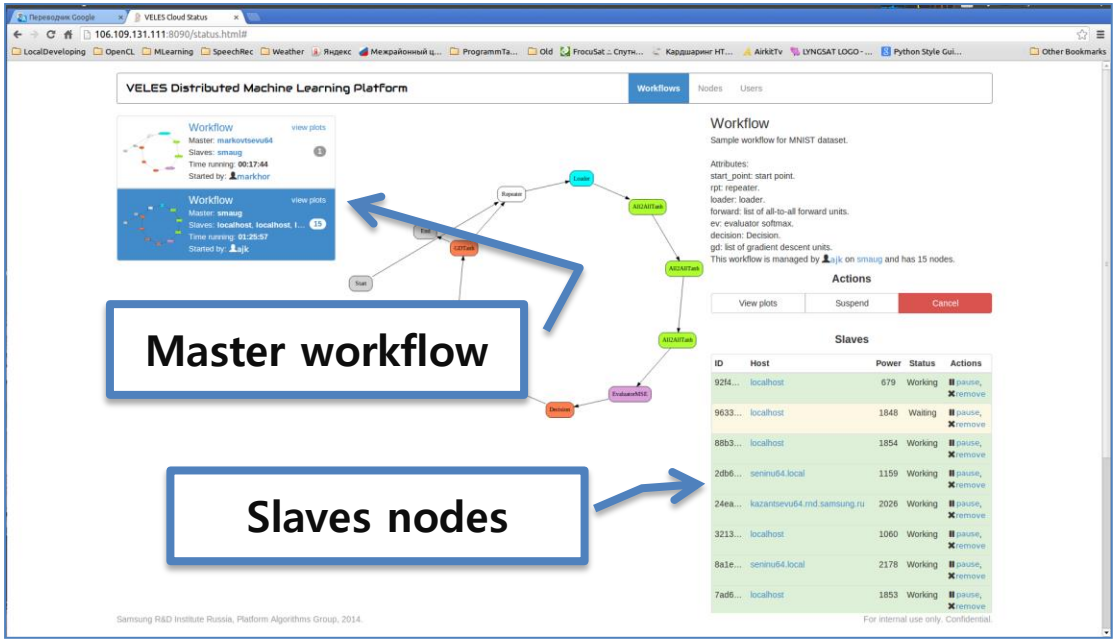
Alex can easily watch how training process is going with Plotter and Web Status Service.

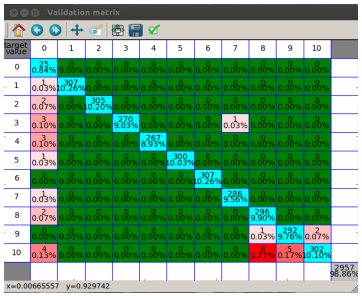
Local or Web based Plotter Service

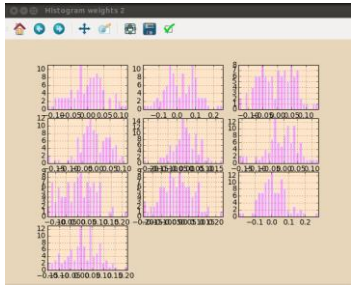


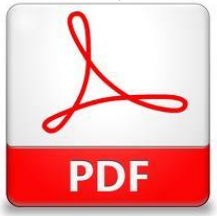


Web based Status Dashboard









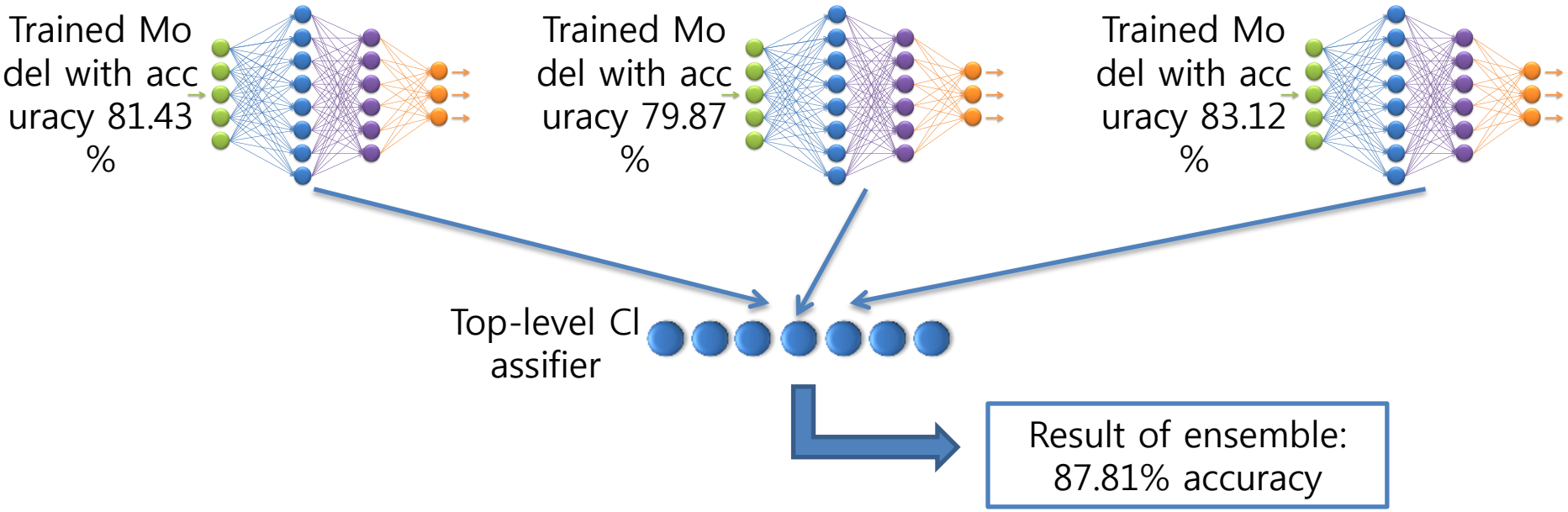
Alex can save all graphics in PDF at any moment of Training.

Training the ensemble

Veles automates the process of training ensembles.

It consists of 3 separate steps:

1. Train the models which are be included into the ensemble.
2. Evaluate those models on a separate part of the dataset (this ensures that the e nsemble does not adapt to the validation set).
3. Train the top-level classifier on a separate part of the dataset which uses the ou tput from step 2 as features.



Publishing the results

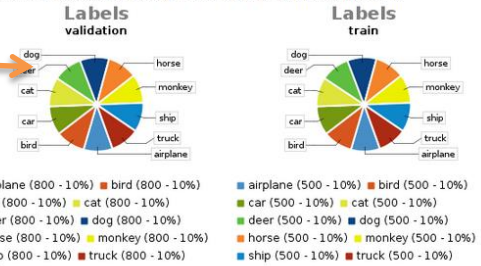
User can publish the results about training process. Typical report consists of general information about the workflow, achieved results, data basic analytics, plots, run statistics, etc.

Data statistics

validation (8,000 - 62%)
train (5,000 - 38%)

Labels

['airplane', 'bird', 'car', 'cat', 'deer', 'dog', 'horse', 'monkey', 'ship', 'truck'] (total: 10)



Normalization

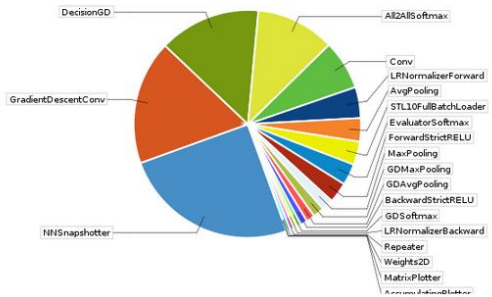
type of normalization	internal_mean
normalization parameters	{}

Run statistics

Run stats

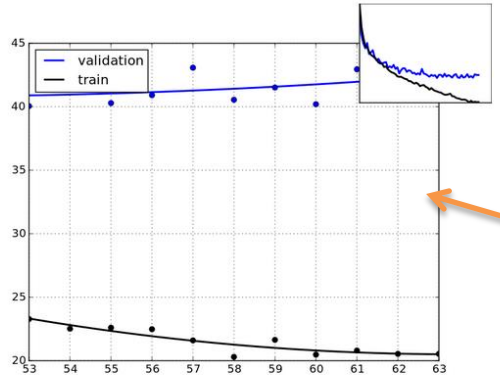
elapsed time	0.0 days, 0.0 hours, 21.0 minutes, 8.5 secs
number of epochs	62

Unit run time



Plots

Number of errors



train matrix

target value	airplane	bird	car	cat	deer	dog	horse	monkey	ship	truck	
airplane	426 8.52%	12 0.24%	5 0.10%	2 0.04%	1 0.02%	5 0.10%	1 0.02%	2 0.04%	19 0.38%	15 0.30%	
bird	14 0.28%	389 7.78%	3 0.06%	26 0.52%	15 0.30%	21 0.42%	8 0.16%	21 0.42%	2 0.04%	4 0.08%	
car	12 0.24%	2 0.04%	433 8.66%	4 0.08%	1 0.02%	0 0.00%	3 0.06%	3 0.06%	10 0.20%	45 0.90%	
cat	1 0.02%	28 0.56%	0 0.00%	347 6.94%	28 0.56%	49 0.98%	11 0.22%	26 0.52%	2 0.04%	3 0.06%	
deer	1 0.02%	12 0.24%	1 0.02%	26 0.52%	396 7.92%	21 0.42%	20 0.40%	8 0.16%	1 0.02%	0 0.00%	
dog	3 0.06%	26 0.52%	1 0.02%	53 1.06%	20 0.40%	339 6.78%	29 0.58%	37 0.74%	2 0.04%	2 0.04%	
horse	5 0.10%	6 0.12%	1 0.02%	7 0.14%	21 0.42%	30 0.60%	407 8.14%	17 0.34%	1 0.02%	4 0.08%	
monkey	1 0.02%	22 0.44%	0 0.00%	31 0.62%	16 0.32%	33 0.66%	13 0.26%	384 7.68%	2 0.04%	1 0.02%	
ship	25 0.50%	3 0.06%	7 0.14%	2 0.04%	1 0.02%	1 0.02%	1 0.02%	2 0.04%	446 8.92%	20 0.40%	
truck	12 0.24%	0 0.00%	49 0.98%	2 0.04%	1 0.02%	1 0.02%	7 0.14%	0 0.00%	15 0.30%	406 8.12%	
											3973 79.46%

validation matrix

target value	airplane	bird	car	cat	deer	dog	horse	monkey	ship	truck	
airplane	676 8.45%	123 1.54%	48 0.60%	17 0.21%	15 0.19%	16 0.20%	9 0.11%	9 0.11%	74 0.93%	56 0.70%	