Marian: Homecoming
Marcin Junczys-Dowmunt
A few words about Marian

- Portable C++ code with minimal dependencies (CUDA or MKL and still Boost);
- Single engine for training and decoding on GPU and CPU;
- Custom auto-diff engine with dynamic graphs (similar to DyNet);
- Optimized towards NMT.

http://marian-nmt.github.io and
https://github.com/marian-nmt/marian
Part I

A Machine Translation Marathon 2016 project
The first commit

Commit: 6a7c93
Date: May 4th, 2016
Message: very cool

Lines: 155
```cpp
#include <iostream>
#include "mad.h"

int main(int argc, char** argv) {
    Var x0 = 1, x1 = 2, x2 = 3;
    auto y = x0 + x0 + log(x2) + x1;
    std::vector<Var> x = { x0, x1, x2 }
    set_zero_all_adjoints();
    y.grad();

    std::cerr << "y_val = " << y.val() << std::endl;
    for(int i = 0; i < x.size(); ++i)
        std::cerr << "dy/dx_" << i << " = " << x[i].adj() << std::endl;
}
```
Var x0 = 1, x1 = 2, x2 = 3;
auto y = x0 + x0 + log(x2) + x1;

y = 5.09861
\frac{dy}{dx}_0 = 2
\frac{dy}{dx}_1 = 1
\frac{dy}{dx}_2 = 0.333333
Lines of code over time

First commit
A Neural Network Toolkit for MT

Maximiliana Behnke
Tomasz Dwojak
Marcin Junczys-Dowmunt
Roman Grundkiewicz
Andre Martins
Hieu Hoang
Lane Schwartz
A Neural Network Toolkit
for MT
Maximiliana Behnke
Tomasz Dwojaka
Marcin Junczys-Dowmunt
Roman Grundkiewicz
Andre Martins
Hieu Hoang
Lane Schwartz
Why create another NN toolkit?

- **Flexibility**
  - Add functionality easier & faster

- **Speed**
  - Pure C++ implementation
  - GPU-enabled (CPU may come soon)

- **Learn about Deep Learning**
  - Implement everything from scratch by ourselves
What we’ve achieved this week?

● Framework to create computation graphs
  ○ Simple feedforward NN
  ○ RNN, GRU, LSTM…
  ○ Binary or multiclass classifier

● Forward step
  ○ Classify, given input data and weights

● Backward step
  ○ Learn weights using backpropagation

● Tested with small datasets
  ○ MNIST (digit image recognition task)
  ○ MT

● Documentation
  ○ Doxygen
What needs to be finished?

- Basic features:
  - Data shuffling
  - More random distributions
  - ...
- Model serialization & deserialization
- Documentation
- ...

...
Lines of code over time

MTM 2016

Working Nematus Model

First commit
Lines of code over time

MTM
2016
Working
Nematus
Model
AARA
Multi-gpu
Training
Marian
decoder
First
commit
Going further

- Reduce dependencies for CPU version to zero
- Reduce dependencies for GPU version to CUDA
- Become faster and more versatile
- Research tool with immediate deployment
### Interested in Contributing?

- Check out available resources.
- Create an account and start submitting your own systems.

### Scored Systems

<table>
<thead>
<tr>
<th>System</th>
<th>Submitter</th>
<th>System Notes</th>
<th>Constraint</th>
<th>Run Notes</th>
<th>BLEU</th>
<th>BLEU-cased</th>
<th>TER</th>
<th>BEER 2.0</th>
<th>CharactTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marian Transformer</td>
<td>marcindl@Microsoft</td>
<td>Marian Transformer-Big</td>
<td>yes</td>
<td>Transformer-big ensemble x4. With back-translation, data-filtering on Paracrawl with domain-weighting. Decoder-time ensembling with transformer-LM, right-to-left decoding.</td>
<td>48.9</td>
<td>48.3</td>
<td>0.407</td>
<td>0.697</td>
<td>0.362</td>
</tr>
<tr>
<td>NMT-SMT Hybrid</td>
<td>fstahlberg@University of Cambridge</td>
<td>MBR-based combination of neural models and SMT</td>
<td>yes</td>
<td></td>
<td>47.1</td>
<td>46.6</td>
<td>0.415</td>
<td>0.691</td>
<td>0.369</td>
</tr>
<tr>
<td>NTT Transformer-based System</td>
<td>nakazato-nr@NTT</td>
<td>Based on Transformer-Big model. Trained with filtered version of CommonCrawl, Paracrawl and synthetic corpus of news-crawl2017. R2L reranking.</td>
<td>yes</td>
<td></td>
<td>47.0</td>
<td>46.5</td>
<td>0.426</td>
<td>0.688</td>
<td>0.370</td>
</tr>
<tr>
<td>Primary Submission</td>
<td>Danist@Karlsruhe Institute of Technology</td>
<td>Primary Submission</td>
<td>yes</td>
<td></td>
<td>46.9</td>
<td>46.3</td>
<td>0.429</td>
<td>0.687</td>
<td>0.382</td>
</tr>
<tr>
<td>MMT-unconstrained</td>
<td>nicole@beroldi@MTT arl</td>
<td></td>
<td>no</td>
<td></td>
<td>46.7</td>
<td>46.2</td>
<td>0.432</td>
<td>0.682</td>
<td>0.387</td>
</tr>
<tr>
<td>JHU</td>
<td>hu-jhnu-alt@Johns Hopkins University</td>
<td>Marian Deep RNN</td>
<td>yes</td>
<td>Marian deep model, ensemble of 4 runs using base data (without Paracrawl), back-translated news 2016. Not final system yet.</td>
<td>43.6</td>
<td>43.0</td>
<td>0.453</td>
<td>0.670</td>
<td>0.394</td>
</tr>
<tr>
<td>MMT-constraint-dec</td>
<td>nicole@beroldi@MTT arl</td>
<td></td>
<td>yes</td>
<td></td>
<td>42.9</td>
<td>42.5</td>
<td>0.463</td>
<td>0.667</td>
<td>0.411</td>
</tr>
<tr>
<td>NUNJMT</td>
<td>ZhaoChengeng@Nanjing University</td>
<td>transformer base without back translation</td>
<td>yes</td>
<td>transformer base without back translation</td>
<td>40.6</td>
<td>40.0</td>
<td>0.496</td>
<td>0.647</td>
<td>0.436</td>
</tr>
<tr>
<td>LMU-unsupervised-nmt-wmt18-en-de</td>
<td>Matthias.Hack@NTNU Munich</td>
<td>Unsupervised NMT (no parallel training corpora)</td>
<td>yes</td>
<td></td>
<td>15.0</td>
<td>15.5</td>
<td>0.762</td>
<td>0.500</td>
<td>failed</td>
</tr>
<tr>
<td>RWTH -Unsupervised NMT Ensemble</td>
<td>jens@RWTH Aachen University</td>
<td>(Unsupervised) Transformer with shared encoder/decoders, separate top-50k word</td>
<td>yes</td>
<td></td>
<td>15.9</td>
<td>14.8</td>
<td>0.753</td>
<td>0.514</td>
<td>0.607</td>
</tr>
</tbody>
</table>
### Scored Systems

<table>
<thead>
<tr>
<th>System</th>
<th>Submitter</th>
<th>System Notes</th>
<th>Constraint</th>
<th>Run Notes</th>
<th>BLEU</th>
<th>BLEU-cased</th>
<th>TER</th>
<th>BEER 2.0</th>
<th>CharacterTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marian-Transformer</td>
<td>marcinjl microsoft</td>
<td>Marian Transformer-Big</td>
<td>yes</td>
<td>Transformer-big ensemble x4. With back-translation, data-filtering on ParallelCrawl with domain-weighting. Decoder-time ensembling with transformer-LM, right-to-left decoding.</td>
<td>48.9</td>
<td>48.3</td>
<td>0.407</td>
<td>0.697</td>
<td>0.362</td>
</tr>
<tr>
<td>NMT-SMT Hybrid</td>
<td>fstahlberg University of Cambridge</td>
<td>MBR-based combination of neural models and SMT</td>
<td>yes</td>
<td></td>
<td>47.1</td>
<td>46.6</td>
<td>0.415</td>
<td>0.691</td>
<td>0.369</td>
</tr>
<tr>
<td>NTT Transformer-based System</td>
<td>nakanoto mr NTT</td>
<td>Based on Transformer-Big model. Trained with filtered version of CommonCrawl, ParallelCrawl and synthetic corpus of news-crawl2017. R2L reranking</td>
<td>yes</td>
<td></td>
<td>47.0</td>
<td>46.5</td>
<td>0.426</td>
<td>0.688</td>
<td>0.370</td>
</tr>
<tr>
<td>Primary Submission</td>
<td>Danyang Karlsruhe Institute of Technology</td>
<td>Primary Submission</td>
<td>yes</td>
<td></td>
<td>46.9</td>
<td>46.3</td>
<td>0.420</td>
<td>0.687</td>
<td>0.382</td>
</tr>
<tr>
<td>MMT-unconstrained</td>
<td>niccolaboldi MNT srl</td>
<td>MNT unclouded</td>
<td>no</td>
<td></td>
<td>46.7</td>
<td>46.2</td>
<td>0.432</td>
<td>0.682</td>
<td>0.387</td>
</tr>
<tr>
<td>HU</td>
<td>huawei</td>
<td>Marian Deep RNN</td>
<td>yes</td>
<td>Marian deep model, ensemble of 4 runs using base data (without ParallelCrawl), reback-translated news 2016. Not final system yet.</td>
<td>43.6</td>
<td>43.0</td>
<td>0.453</td>
<td>0.670</td>
<td>0.394</td>
</tr>
<tr>
<td>MTT-constrained</td>
<td>niccolaboldi MNT srl</td>
<td>MTT constrained</td>
<td>yes</td>
<td></td>
<td>42.9</td>
<td>42.5</td>
<td>0.463</td>
<td>0.667</td>
<td>0.411</td>
</tr>
<tr>
<td>NUJUJUTU</td>
<td>zhaochengqi Shanghai University</td>
<td>Unsupervised NMT</td>
<td>yes</td>
<td>transformer base without back translation</td>
<td>40.6</td>
<td>40.0</td>
<td>0.496</td>
<td>0.647</td>
<td>0.436</td>
</tr>
<tr>
<td>LMU-unsupervised-nmt-wmt18-en-de</td>
<td>mathias huick FNT Munich</td>
<td>Unsupervised NMT (no parallel training corpora)</td>
<td>yes</td>
<td></td>
<td>15.8</td>
<td>15.5</td>
<td>0.762</td>
<td>0.500</td>
<td>failed</td>
</tr>
<tr>
<td>RWTH Unsupervised NMT Ensemble</td>
<td>junsukkim RWTH Aachen University</td>
<td>Transformer with shared encoder/decoder, separate top-50k word</td>
<td>yes</td>
<td></td>
<td>15.9</td>
<td>14.8</td>
<td>0.753</td>
<td>0.514</td>
<td>0.607</td>
</tr>
</tbody>
</table>
A major update of the FairSeq open source code,
- Transformer and Gated ConvNet models for translation.
- pre-trained models
- parallel training and reduced precision.

We are releasing new features for fairseq, FAIR's sequence to sequence learning library:
https://github.com/pytorch/fairseq

# Distributed training, fp16, delayed batching
We release code and pre-trained models to reproduce our recent paper "Scaling Neural Machine Translation" (https://arxiv.org/abs/1806.00197) where we train on up to 128 GPUs with half precision floating point operations as well ... See More

# Fast inference
Fairseq can generate translations at a rate of 92 sentences/sec for big Transformers on a fast GPU by clever caching, removing finished sentences from the computation and by batching tokens. This improves speed by nearly 60%. The image shows a comparison to other toolkits measured on a Y100 GPU for WMT English-German translation on newest2014 using a big Transformer.

# Language models
Fairseq now supports the training of gated convolutional language models
Facebook's fairseq MT engine is really, really fast... Like, 50% faster than @marian_nmt (which is itself way faster than Sockeye/OpenNMT/Tensor2Tensor/xnmt/Ne matus/etc) at generating from the same Transformer model
facebook.com/61013326/posts ...

2:24 PM - 15 Jun 2018

37 Retweets 139 Likes

Marian NMT @marian_nmt · Jun 15
Replying to @jekbradbury
Hold my beer ;)
Facebook's fairseq MT engine is really, really fast... Like, 50% faster than @marian_nmt (which is itself way faster than Sockeye/OpenNMT/Tensor2Tensor/xnmt/Neumatus/etc) at generating from the same Transformer model facebook.com/61013326/posts...
Marian NMT @marian_nmt · Jun 17

Boom! Marian v1.5.0 released.

Includes:
- Extensions from the WNMT shared task on efficiency [arxiv.org/abs/1805.12096](http://arxiv.org/abs/1805.12096)
- Optimized GPU-decoding for Transformer models.

See chart below for speed comparison to v1.4.0 (based on FAIR's post)
@jekbradbury @alvations
Part II

Decoding on the CPU
Quality first – speed later

- Lessons from WNMT shared task on efficient decoding;
- Sequence-level knowledge distillation (Kim & Rush 2016):
- Training four Transformer-big models on official task data (teacher);
- Translate entire EN data to DE-trans (8-best list);
- Select sentences with highest sentence-level BLEU based on DE-orig data;
- Train students on EN/DE-trans data.
Teacher
Transformer Big

\[
\begin{array}{cc}
1024 & 1024 \\
4096 & 4096 \\
1024 & 1024 \\
4096 & 4096 \\
\end{array}
\]

4 \times 813 \text{ MiB}
29.0

(Scale preserving)
Teacher Transformer Big

Students Transformer beam=1

Big

Base

Small

4 × 813 MiB
813 MiB
238 MiB
101 MiB

29.0
28.2
27.6
26.4

1024 4096
1024 4096
1024 4096
1024 4096

512 2048
256 2048

(Scale preserving)
Seconds to translate newstest2014 (batch-size: ca. 384 words – 5 to 25 sentences)
Seconds to translate newstest2014 (batch-size: ca. 384 words – 5 to 25 sentences)

- Transformer-base
- + Shortlist
- + AAN (modified)
- + MM-int16
- + Memoization
- + Auto-tuning
- + Caching att. keys/values
- w/o --optimize

- MM-MKL
- MM-int16
- Quant-int16
- Other

542.6: 412.0
Multiplicative attention:

\[
Q' = QW_q + b_q \\
K' = KW_k + b_k \\
V' = VW_v + b_v \\
C = \text{softmax}(Q' \times (K')^T) \times V' \\
Y = \text{norm}(Q + C)
\]

- During training: \( Q = K = V \)
- During translation: \( Q \neq K; K = V \)
- Complexity per step: \( O(n) \)
- Because:
  \[
c_t = \text{softmax}(q'_t \times (K'_{<t})^T) \times V'_{<t} \\
K_{<t+1} = [K_{<t}; q_t]
\]
Multiplicative attention:

\[ Q' = QW_q + b_q \]
\[ K' = KW_k + b_k \]
\[ V' = VW_v + b_v \]
\[ C = \text{softmax}(Q' \times (K')^T) \times V' \]
\[ Y = \text{norm}(Q + C) \]

- During training: \( Q = K = V \)
- During translation: \( Q \neq K; K = V \)
- Complexity per step: \( O(n) \)
- Because:
  \[ c_t = \text{softmax}(q_t' \times (K'_{<t})^T) \times V'_{<t} \]
  \[ K_{<t+1} = [K_{<t}; q_t] \]

Average attention network (Zhang et al. 2018):

\[ C = \text{gate(FFN}(\bar{V}), Q) \]
\[ Y = \text{norm}(Q + C) \]

- Gate and FFN optional
- Complexity per step: \( O(1) \)
- Because: \( \bar{v}_t = \frac{1}{t}((t - 1)\bar{v}_{t-1} + v_t) \)
- Basically a weird RNN
- Authors report 4x speed-up for beam=4
Seconds to translate newstest2014 (batch-size: ca. 384 words – 5 to 25 sentences)
Code based on Devlin (2017), extended to AVX512

\[ q(x) = \text{int16}(x \cdot 2^{10}) \]
Code based on Devlin (2017), extended to AVX512

\[ q(x) = \text{int16}(x \cdot 2^{10}) \]

\[ A_q \otimes B_q = A_q \times B_q^T \]
Code based on Devlin (2017), extended to AVX512

\[ q(x) = \text{int16}(x \cdot 2^{10}) \]
\[ A_q \otimes B_q = A_q \times B_q^T \]

\[ x \times W = x \times (W^T)^T \]
\[ \approx q(x) \times q(W^T)^T \]
\[ = q(x) \otimes q(W^T) \]
\[ \text{dot}(x, W) \]

\[ \text{dot}_{16}(x, W) \]

\[ \text{quantize} \] \[ \text{transpose} \]

\[ \text{constant} \]
- Transformer-base
  - + Shortlist
  - + AAN (modified)
  - + MM-int16
  - + Memoization
  - + Auto-tuning
  - + Caching att. keys/values
  - w/o --optimize

Seconds to translate newstest2014 (batch-size: ca. 384 words – 5 to 25 sentences)
\[ \text{dot}(x, W) \]

\[ x \times W \]

\[ \text{dot}_{16}(x, W) \]

\[ x \times \text{quantize} \times W \]

\[ \text{quantize} \times \text{quantize} \times \text{transpose} \]

\[ x \times \text{transpose} W \]
$$\text{dot}(x, W)$$

$$x \times W$$

$$\text{dot}_{16}(x, W)$$

$$\text{quantize} \times \text{transpose}^{\text{constant}}$$

$$\text{quantize}$$

$$\text{tuned} = x \times W$$

$$\text{measure}(h_1)$$

$$\text{measure}(h_2)$$

$$\text{measure}(h_2)$$

$$\text{dot}$$

$$\text{constant}$$

$$\text{parameter}$$
\[ \text{dot}(x, W) \]

\[ x \times W \]

\[ \text{dot}_{16}(x, W) \]

\[ \text{quantize} \times \text{quantize} \times \text{transpose} \]

constant

\[ x \]

\[ W \]

parameter
\[ \text{dot}(x, W) \]

\[ x \times W \]

\[ \text{dot}_{16}(x, W) \]

\[ x \times W \]

\[ \text{quantize} \]

\[ \text{quantize} \]

\[ \text{transpose} \]

\[ x \]

\[ W \]

\[ \text{constant} \]

\[ \text{constant} \]

\[ \text{parameter} \]
\[
\text{dot}(x, W) = x \cdot W
\]

\[
\text{dot}_{16}(x, W) = \text{quantize}(\text{transpose}(W)) \cdot x \cdot \text{quantize}(W)
\]

\[
\text{measure}(h_1) \cdot x \cdot \text{measure}(h_2) \cdot \text{tuned}(x, W)
\]

\[
\text{memoized}
\]

\[
\text{constant}
\]

\[
\text{constant}
\]

\[
\text{parameter}
\]
Secondstotranslatenewstest2014 (batch-size: ca. 384 words – 5 to 25 sentences)
\[ \text{dot}(x, W) \]
\[ \text{dot}_{16}(x, W) \]

\[ \text{quantize} \times \text{transpose} \]
\[ \text{constant} \]
\[ \text{parameter} \]

\[ \text{memoized} \]
\[ h_1 = \text{hash}(\text{dot}(x, W)) \]
\[ = \text{hash}(\text{dot}) \circ \text{hash}(\text{dims}(x)) \circ \text{hash}(\text{dims}(W)) \]
\[ = \text{hash}(\text{dot}) \circ \text{hash}(\{11, 512\}) \circ \text{hash}(\{512, 512\}) \]

\[ h_2 = \text{hash}(\text{dot}_{16}(x, W)) \]
\[ = \text{hash}(\text{dot}_{16}) \circ \text{hash}(\text{dims}(x)) \circ \text{hash}(\text{dims}(W)) \]
\[ = \text{hash}(\text{dot}_{16}) \circ \text{hash}(\{11, 512\}) \circ \text{hash}(\{512, 512\}) \]

We can decrease granularity via integer-dividing dimensions by a given factor, we choose 5.
\[ \text{dot}(x, W) \]

\[ \text{dot}_{16}(x, W) \]

\[ \text{quantize} \]

\[ \text{transpose} \]

\[ \text{constant} \]

\[ \text{constant} \]

\[ \text{parameter} \]

\[ \text{memoized} \]
\[ \text{dot}(x, W) \]
Secondstotranslatenewstest2014 (batch-size: ca. 384 words – 5 to 25 sentences)
Second to translate newstest2014 (batch-size: ca. 384 words – 5 to 25 sentences)
<table>
<thead>
<tr>
<th>Configuration</th>
<th>MM-MKL</th>
<th>MM-int16</th>
<th>Quant-int16</th>
<th>Other</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer-base</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>542.6</td>
</tr>
<tr>
<td>+ Shortlist</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>412.0</td>
</tr>
<tr>
<td>+ AAN (modified)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>304.7</td>
</tr>
<tr>
<td>+ MM-int16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>505.9</td>
</tr>
<tr>
<td>+ Memoization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>314.3</td>
</tr>
<tr>
<td>+ Auto-tuning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>281.9</td>
</tr>
<tr>
<td>+ Caching att. keys/values w/o --optimize</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>157.0</td>
</tr>
<tr>
<td>Marian v1.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>182.6</td>
</tr>
</tbody>
</table>

Seconds to translate newstest2014 (batch-size: ca. 384 words – 5 to 25 sentences)
Latency per sentence in milliseconds for newstest2014 (batch-size: 1 sentence)

- Transformer-base
  - +Shortlist
  - +AAN (modified) using --optimize
  - +Caching att. keys/values w/o --optimize

Marian v1.5

Latency:
- Transformer-base: 501.3 ms
- +Shortlist: 344.5 ms
- +AAN (modified) using --optimize: 265.4 ms
- +Caching att. keys/values w/o --optimize: 215.1 ms
- Marian v1.5: 501.3 ms
Map GPU and CPU performance into comparable space [w/$]

newstest2014.de consists of 62,954 tokens

<table>
<thead>
<tr>
<th>Type</th>
<th>Price [$/h]</th>
</tr>
</thead>
<tbody>
<tr>
<td>p3.x2large</td>
<td>3.259</td>
</tr>
<tr>
<td>m5.large</td>
<td>0.102</td>
</tr>
</tbody>
</table>

\[
[w/\$] = \frac{62,954 \ [w]}{\text{Translation time} \ [s]} \cdot \frac{3,600 \ [s/h]}{\text{Instance price} \ [$/h]}
\]
Million translated source tokens per USD (log scale)
Million translated source tokens per USD (log scale)

BLEU

Marian v1.4 GPU
Marian v1.4 CPU
Marian v1.5 GPU
Marian v1.5 CPU
Others GPU
Others CPU
Future work

- More experiments with Teacher-Student scenario;
- More SIMD operations on the CPU;
- All operations in fixed-point arithmetics on the CPU;
- 8-bit matrix product on the CPU;
- Mixed precision (FP16) on the GPU;
- Optimize beam-search for batched translation;
- ...
We are hiring!
Talk to me if you are interested.