Convolutional Neural Networks for Text Classification using Intel Nervana Neon

Kripa Sankaranarayanan, Yinyin Liu

Acknowledgements: Vipul Lal, Hanlin Tang, Sejun Kim

TOPICS

Neon

CNN in Text Classification

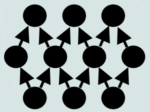
Two Implementation examples of CNN for Text classification in Neon

INTEL® NERVANA™ FULL STACK AI PLATFORM

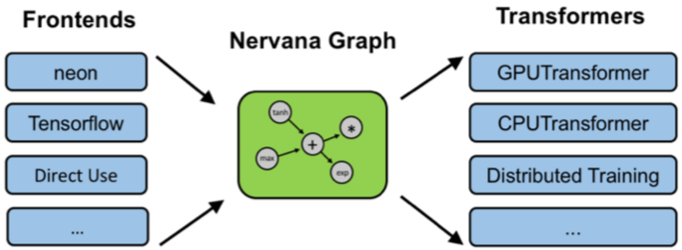
- **Nervana Cloud →** Build an AI POC
- **neon →** Train DL models quickly
- **Intel Nervana Graph →** any framework, any hardware
- **Intel Nervana HW →** industry leading AI, coming soon

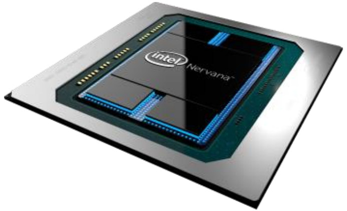
“deep learning by design”

import build train deploy

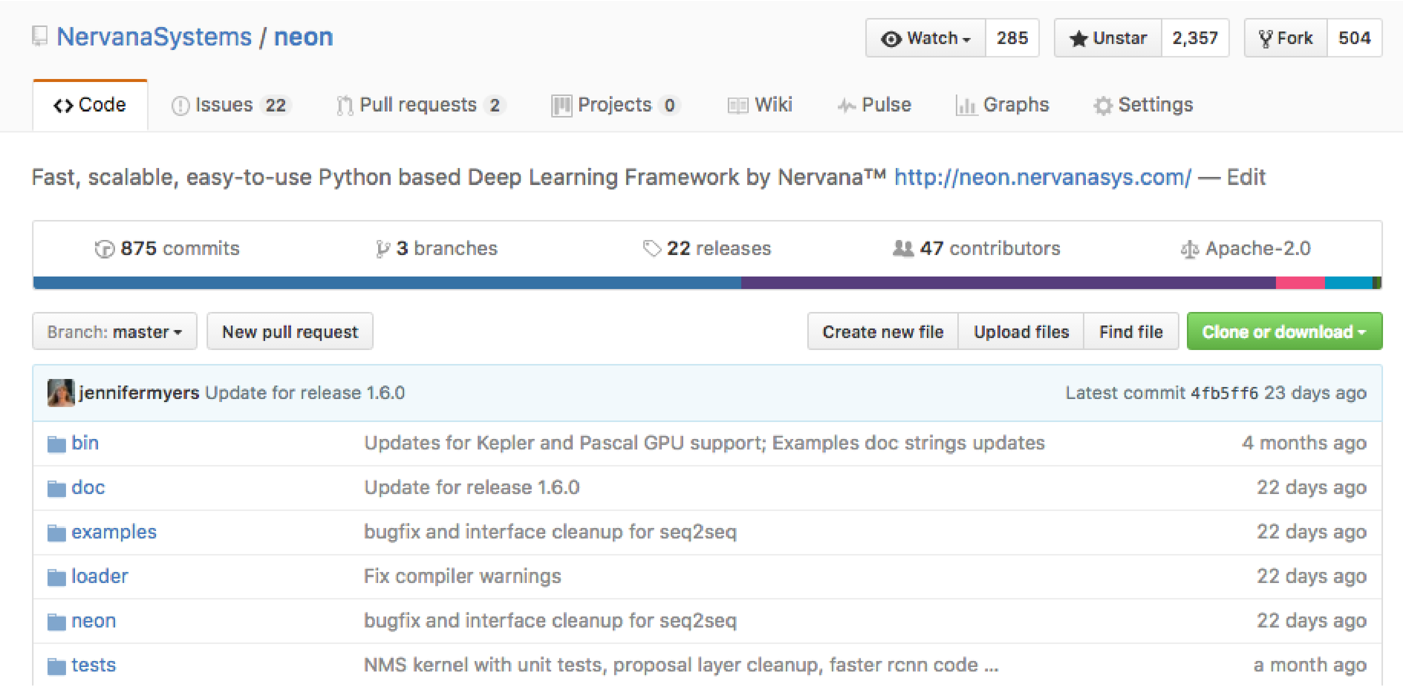


neon
deep learning
framework





DEEP LEARNING WITH NEON



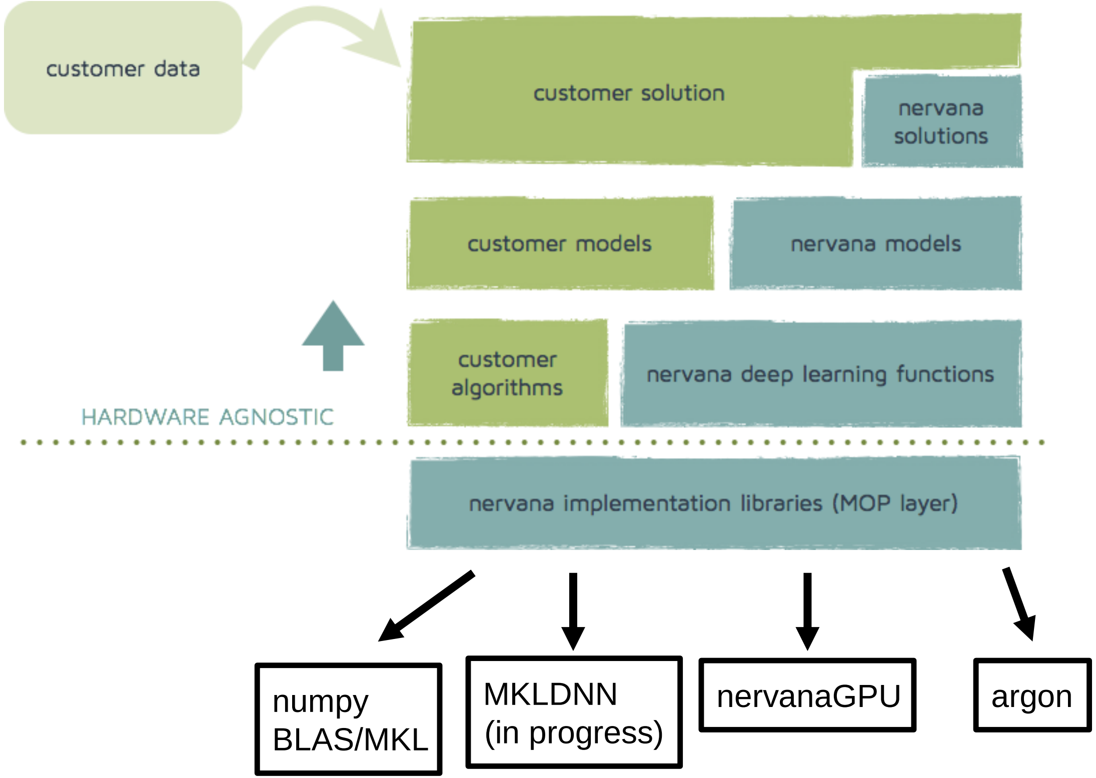
Fast, scalable, easy-to-use Python based Deep Learning Framework by Nervana™ http://neon.nervanasys.com/ — Edit

jennifermyers Update for release 1.6.0 Latest commit 4fb5ff6 23 days ago

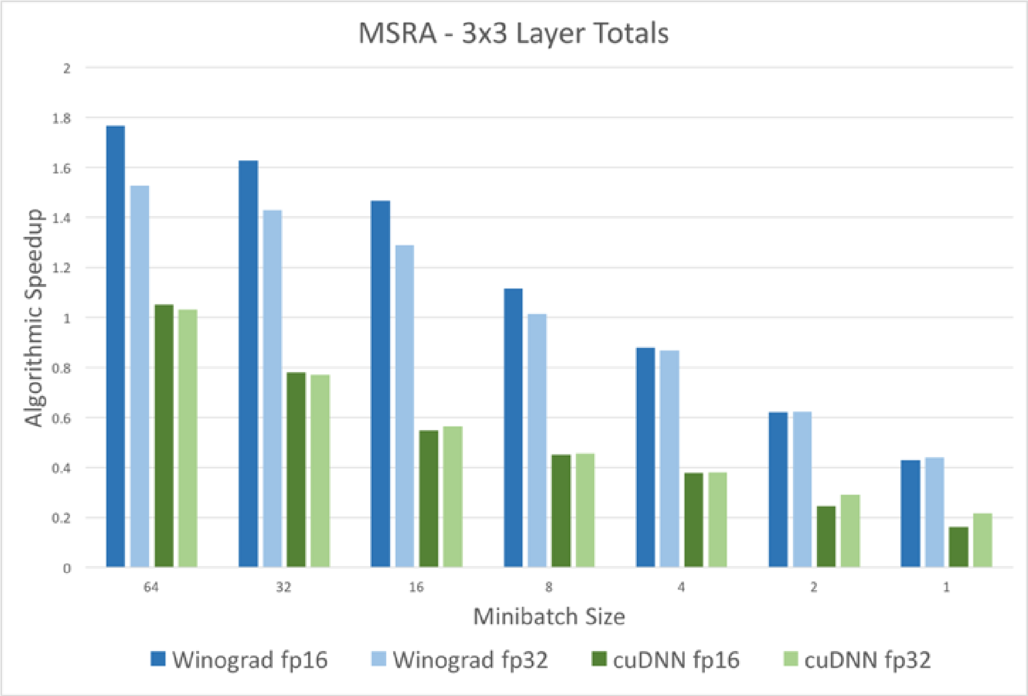
bin	Updates for Kepler and Pascal GPU support; Examples doc strings updates	4 months ago
doc	Update for release 1.6.0	22 days ago
examples	bugfix and interface cleanup for seq2seq	22 days ago
loader	Fix compiler warnings	22 days ago
neon	bugfix and interface cleanup for seq2seq	22 days ago
tests	NMS kernel with unit tests, proposal layer cleanup, faster rcnn code ...	a month ago

https://github.com/NervanaSystems/neon

MULTIPLE COMPUTE BACKENDS



SPEEDING UP DEEP LEARNING



https://www.nervanasys.com/winograd/

CURATED MODELS

- https://github.com/NervanaSystems/ModelZoo
- Pre-trained weights and models

Alexnet GoogLeNet

AllCNN VGG

bAbl Q&A

Single Shot Detection Deep Residual Net

SegNet Video Activity Detection

Skip-thought imdb Sentiment Analysis

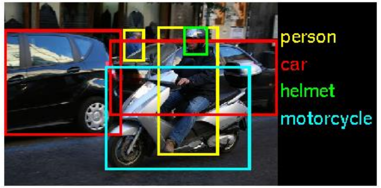
Deep Speech 2 LSTM Image Captioning

Autoencoders Deep Reinforcement Learning

Deep Dream Fast-RCNN Object Localization

Generative Adversarial Networks

NERVANA MODEL LIBRARY



Object localization
Fast-RCNN

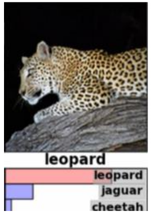


Image classification
Deep Residual Net

Sentiment analysis
LSTM

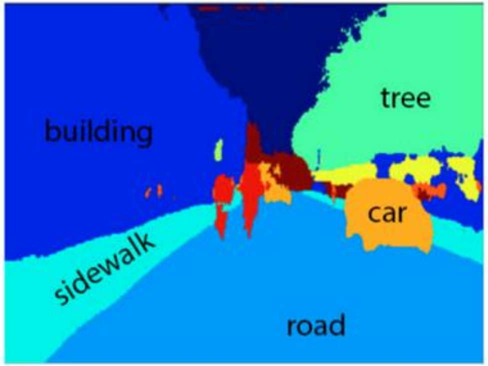
Question answering
GRU

Video activity recognition
3D Convolutional Net

Scene classification
Deep Residual Net

Speech recognition
Deep Speech 2

EXAMPLE CURATED MODEL

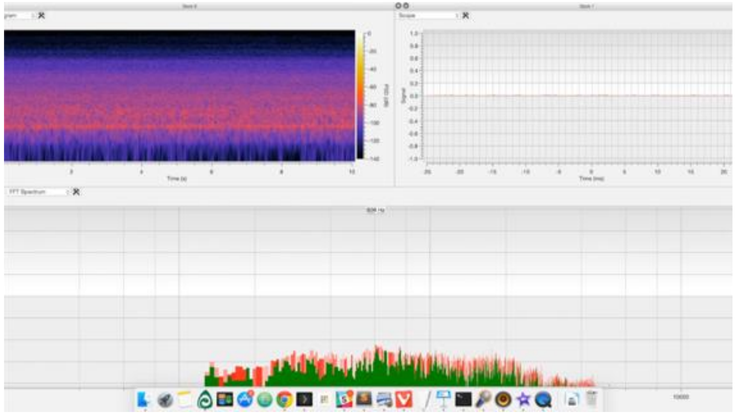


SegNet Model Prediction

- Image segmentation model
- Neon model 5.5x faster than Caffe counterpart
- ~50 fps real time analysis

	Neon (ms)	Caffe (ms)	Speed-up
Forward	101	719	7.1x
Backward	164	746	4.5x
Total	265	1455	**5.5x**

SPEECH RECOGNITION



Reference	CER	WER	WER (LM)
Hannun-Maas (2014)	10.7	35.8	14.1
Graves-Jaitly (ICML 2014)	9.2	30.1	10.4
Hwang-Sung (ICML 2016)	10.6	38.4	8.8
Miao et al (Eesen)	N/A	N/A	9.1
Bahdanau et al (2016)	6.4	18.6	10.8
Nervana Speech	**8.6**	**32.5**	**8.4**
Baidu DS2 (trained on 12,000 hours)			3.6

Wall Street Journal evaluation data (WSJ0)

NEON WORKFLOW

Generate backend

Load data

Specify model architecture

Define training parameters

Train model

Evaluate

NEON OVERVIEW

Backend	NervanaGPU, NervanaCPU, NervanaMGPU
Datasets	MNIST, CIFAR-10, Imagenet 1K, PASCAL VOC, Mini-Places2, IMDB, Penn Treebank, Shakespeare Text, bAbI, Hutter-prize, UCF101, flickr8k, flickr30k, COCO
Initializers	Constant, Uniform, Gaussian, Glorot Uniform, Xavier, Kaiming, IdentityInit, Orthonormal
Optimizers	Gradient Descent with Momentum, RMSProp, AdaDelta, Adam, Adagrad,MultiOptimizer
Activations	Rectified Linear, Softmax, Tanh, Logistic, Identity, ExpLin
Layers	Linear, Convolution, Pooling, Deconvolution, Dropout, Recurrent, Long Short-Term Memory, Gated Recurrent Unit, BatchNorm, LookupTable, Local Response Normalization, Bidirectional-RNN, Bidirectional-LSTM
Costs	Binary Cross Entropy, Multiclass Cross Entropy, Sum of Squares Error
Metrics	Misclassification (Top1, TopK), LogLoss, Accuracy, PrecisionRecall, ObjectDetection

CNN IN TEXT CLASSIFICATION

Labels to text – sentiment, topic

Classifiers

- N-grams with linear classifier
- Deep learning : RNN based on LSTM, GRU
- CNN
  - Reduction in computation, parallelization
  - Hierarchical vs Sequential for RNN
  - Better with large scale datasets
  - No semantic or syntactic knowledge of language

CNN IN TEXT CLASSIFICATION

Character level classification

- Text treated as raw signal at character level
- Characters are a necessary construct - work for different languages
- Characters – one hot representation

Sentence level classification

- Word vectors

EXAMPLE 1:

CHARACTER LEVEL TEXT CLASSIFICATION WITH CNN

DATA

That's right....the red velvet cake.....ohh this stuff is so good! → POSITIVE

Then, as if I hadn't wasted enough of my life there, they poured salt in the wound by drawing out the time it took to bring the check → NEGATIVE

INPUTS

- Sequence of encoded characters
- Encoding
  - prescribe an alphabet of size *m* for the input language
  - quantize each character using 1-of-*m* encoding (or “one-hot” encoding)
  - sequence of characters is transformed to a sequence of *m* sized vectors with fixed length l_0.

DATA PREPROCESSING

“That's right....the red velvet cake.....ohh this stuff is so good!”

→

[50, 38, 30, 50, 7, 47, 3, 48, 37, 35, 38, 50, 16, 16, 16, 16, 50, 38, 33, 3, 48, 33, 34, 3, 52, 33, 42, 52, 33, 50, 3, 31, 30, 39, 33, 16, 16, 16, 16, 16, 43, 38, 38, 38, 3, 50, 38, 37, 47, 3, 47, 50, 49, 36, 36, 3, 37, 47, 3, 47, 43, 3, 35, 43, 43, 34, 16]

- Number of characters limited to 56 (vocab size)
- Truncate each sample to 256 characters [from the left]
- data format is:

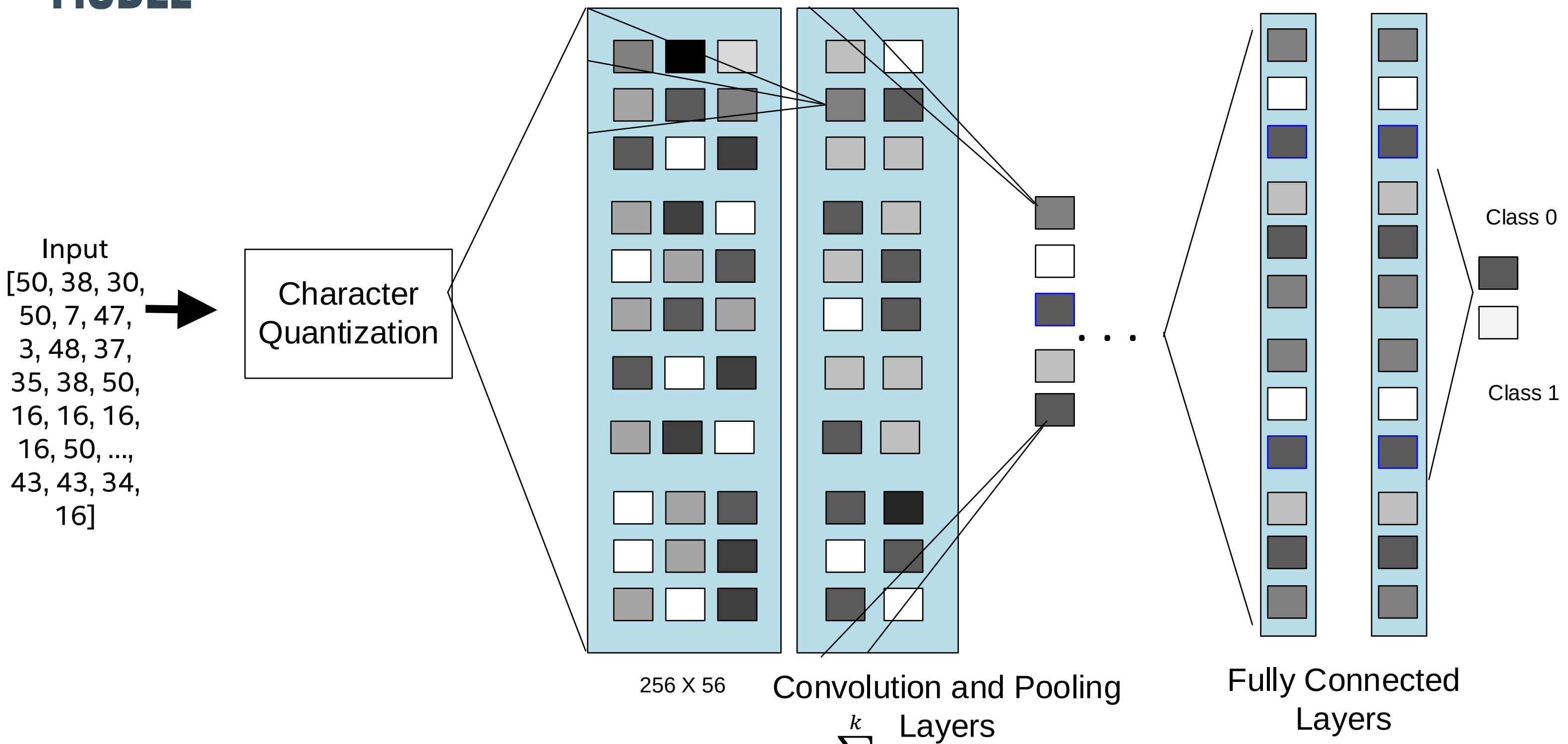
  sentences * max characters per sentence * vocab size

  (#Samples * 256 * 56)

MODEL

- Layers:
  - Convolutional module, computes a 1-D convolution
  - Max-pooling
  - Fully connected
- Algorithm: stochastic gradient descent (SGD)
- Activation: ReLU
- Initialization : Gaussian

MODEL



Input
[50, 38, 30,
50, 7, 47,
3, 48, 37,
35, 38, 50,
16, 16, 16,
16, 50, ...,
43, 43, 34,
16]

Character Quantization

256 X 56

Convolution and Pooling Layers

Fully Connected Layers

Class 0

Class 1

$$Convolution\!:\;\; h(y) = \sum_{x=0}^{k} f(x).\, g(y.d - x + c)$$

$$Pooling\!:\; h(y) = \max g(y.d - x\ + c)$$

NEON COMMANDS

Object that returns minibatches of data

(nsamples, 256, 56)

(nsamples, 2)

```
train_set = ArrayIterator(X_train, y_train, nclass=nclass)
valid_set = ArrayIterator(X_test, y_test, nclass=nclass)
```

Set up initializer

```
init = Gaussian(loc=0.0, scale=0.05)
```

```
layers = [ Conv((5, 5, f_size), init=init, activation=relu), Pooling(3, op='max'),
        Conv((5, 5, f_size), init=init, activation=relu), Pooling(3, op='max'),
        Conv((3, 3, f_size), init=init, activation=relu),
        Conv((3, 3, f_size), init=init, activation=relu),
        Dropout(keep=.5),
        Linear(nout=lin_f, init=init)]
```

Set up Layers

```
char_cnn = Model(layers=layers)

cost = GeneralizedCost(costfunc=CrossEntropyBinary())
optimizer = GradientDescentMomentum(0.01, momentum_coef=0.9)

callbacks = Callbacks(char_cnn, eval_set=valid_set)

char_cnn.fit(train_set, optimizer=optimizer, num_epochs=args.epochs, cost=cost,
callbacks=callbacks)
```

EXAMPLE 2:
SENTENCE CLASSIFICATION WITH CNN

DATASETS

- Sequence of encoded words
  - Sentence max words 56
  - Embedding size 300
  - Vocab size ~20K
  - nsamples: ~9K/1K training/validation

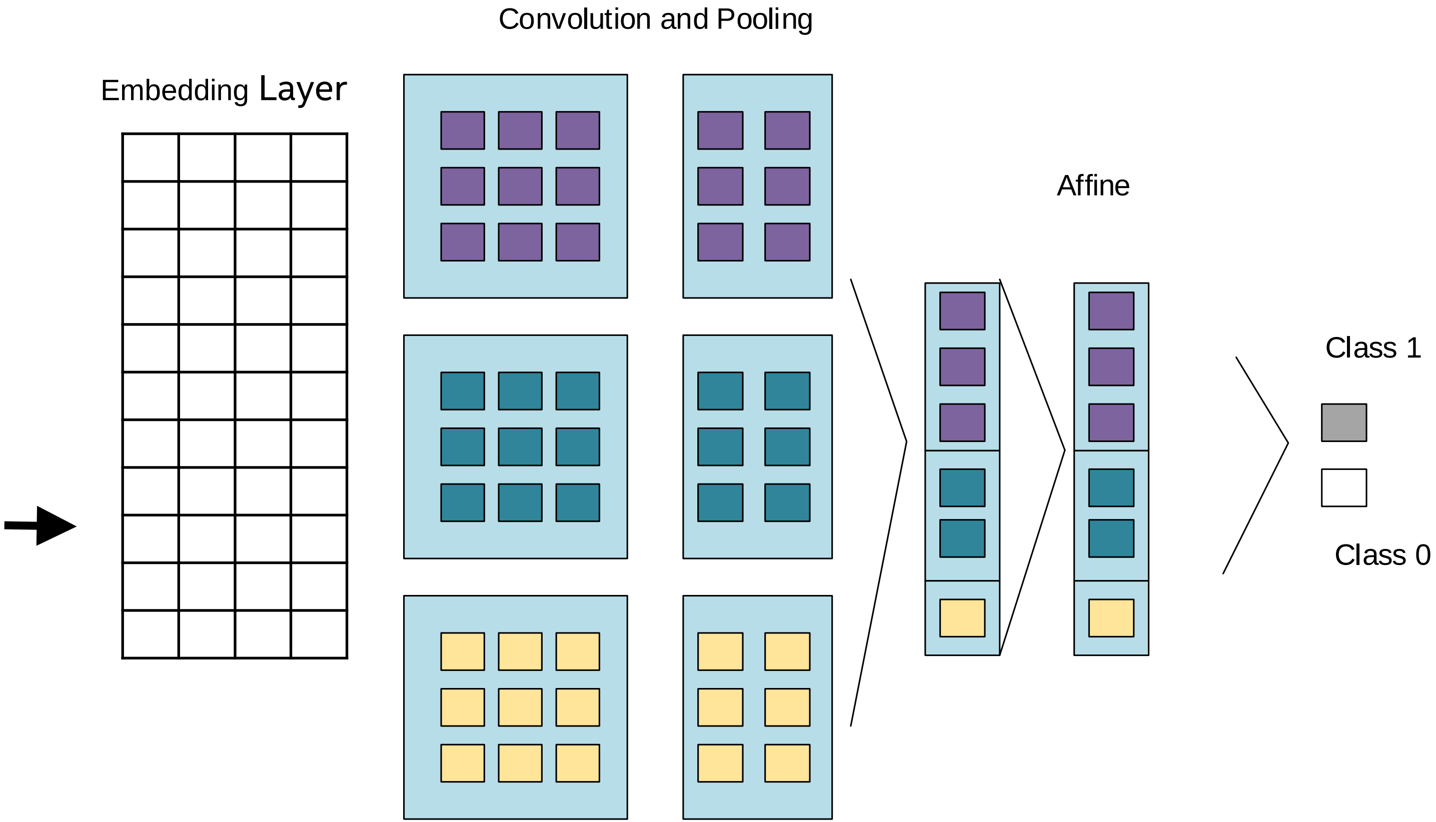
MODEL

- Layers:
  - Embedding Layer
  - 3 Convolutional modules with different filter sizes
  - Max Pooling
  - Dropout layer
  - Affine – fully connected with Softmax
- Algorithm: stochastic gradient descent (SGD)
- Activation: ReLU
- Initialization : Gaussian

MODEL

Data

[751, 1727, 30, 50, 7, 47, 3, 48, 37, 1474, 38, …36, 36, 17, 137, 47, 47, 8127, 7, 35, 1506, 1834]



NEON COMMANDS

Object that returns minibatches of data

```
 train_set = ArrayIterator(X_train, y_train, nclass=nclass)
 valid_set = ArrayIterator(X_test, y_test, nclass=nclass)
```

Set up initializer

Set up Layers

```
init_uni = Uniform(low=-0.1, high=0.1)
g_uni = GlorotUniform()
def gen_layers(fvals):
        (f1, f2, f3) = fvals
        branch1 = [Conv((f1, f1, conv_f), padding = 2, strides = 1, init=init, activation=relu),
Pooling(3, op='max')]
        branch2 = [Conv((f2, f2, conv_f), padding = 2, strides = 1, init=init, activation=relu),
Pooling(3, op='max')]
         branch3 = [Conv((f3, f3, conv_f), padding = 2, strides = 1, init=init, activation=relu),
Pooling(3, op='max')]
        return MergeBroadcast(layers=[branch1, branch2, branch3], merge="stack")

snt_cnn = Model(layers=[
                        LookupTable(vocab_size, embedding_dim=embed_size, init=uni),
                        gen_layers([6,5,3]),
                        Dropout(keep=.5),
                        Affine(lin_f, init=init, bias=g_uni, activation=Softmax())])
```

SUMMARY

Neon

CNN for Text classification implementations in Neon

- Performance with large scale datasets

ACKNOWLEDGEMENTS

Intel IT

Intel Nervana Neon team

AIM Conference committee

BACKUP

REFERENCES AND LINKS

www.intelnervana.com

https://github.com/NervanaSystems/neon

https://arxiv.org/pdf/1509.01626.pdf (Character level CNN for text classification)

https://arxiv.org/pdf/1408.5882.pdf (CNN for Sentence classification)

https://arxiv.org/abs/1702.01923 (Comparative Study of CNN and RNN for NLP)

https://arxiv.org/pdf/1606.01781.pdf (Very Deep Convolutional Networks for Text Classification)