Learning to Embed Byte Sequences with Convolutional Autoencoders

Doug Sibley

Cisco Security Research

OpenAI Unsupervised Sentiment Neuron

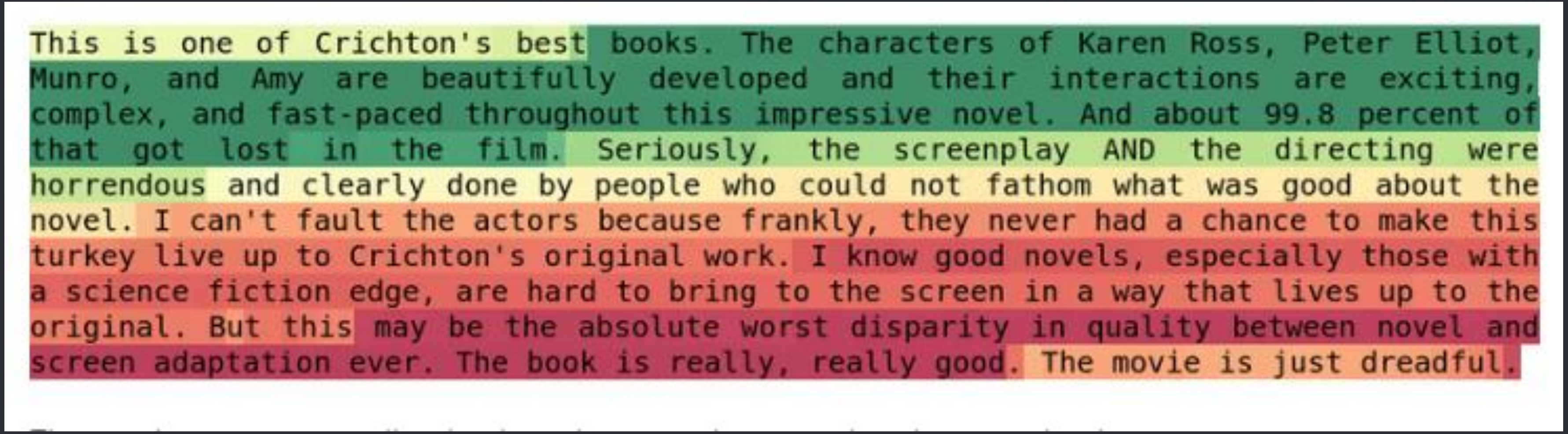
Train RNN to predict next character

Trained on reviews, network learns concept of sentiment

Text represented as UTF8 bytes

85 Million parameters in the model

Trained 1 month on 4 GPUs, 12,500 bytes/sec



This is one of Crichton's best books. The characters of Karen Ross, Peter Elliot, Munro, and Amy are beautifully developed and their interactions are exciting, complex, and fast-paced throughout this impressive novel. And about 99.8 percent of that got lost in the film. Seriously, the screenplay AND the directing were horrendous and clearly done by people who could not fathom what was good about the novel. I can't fault the actors because frankly, they never had a chance to make this turkey live up to Crichton's original work. I know good novels, especially those with a science fiction edge, are hard to bring to the screen in a way that lives up to the original. But this may be the absolute worst disparity in quality between novel and screen adaptation ever. The book is really, really good. The movie is just dreadful.

-OpenAI

Can we replace recurrence with convolutions?

Faster training

More explicit control over receptive field

Benefit from other efforts to improve training/inference speed

Model Overview

Given an input byte sequence, predict every byte in the sequence (same padded convolutions)

710,896 Parameters in the full model

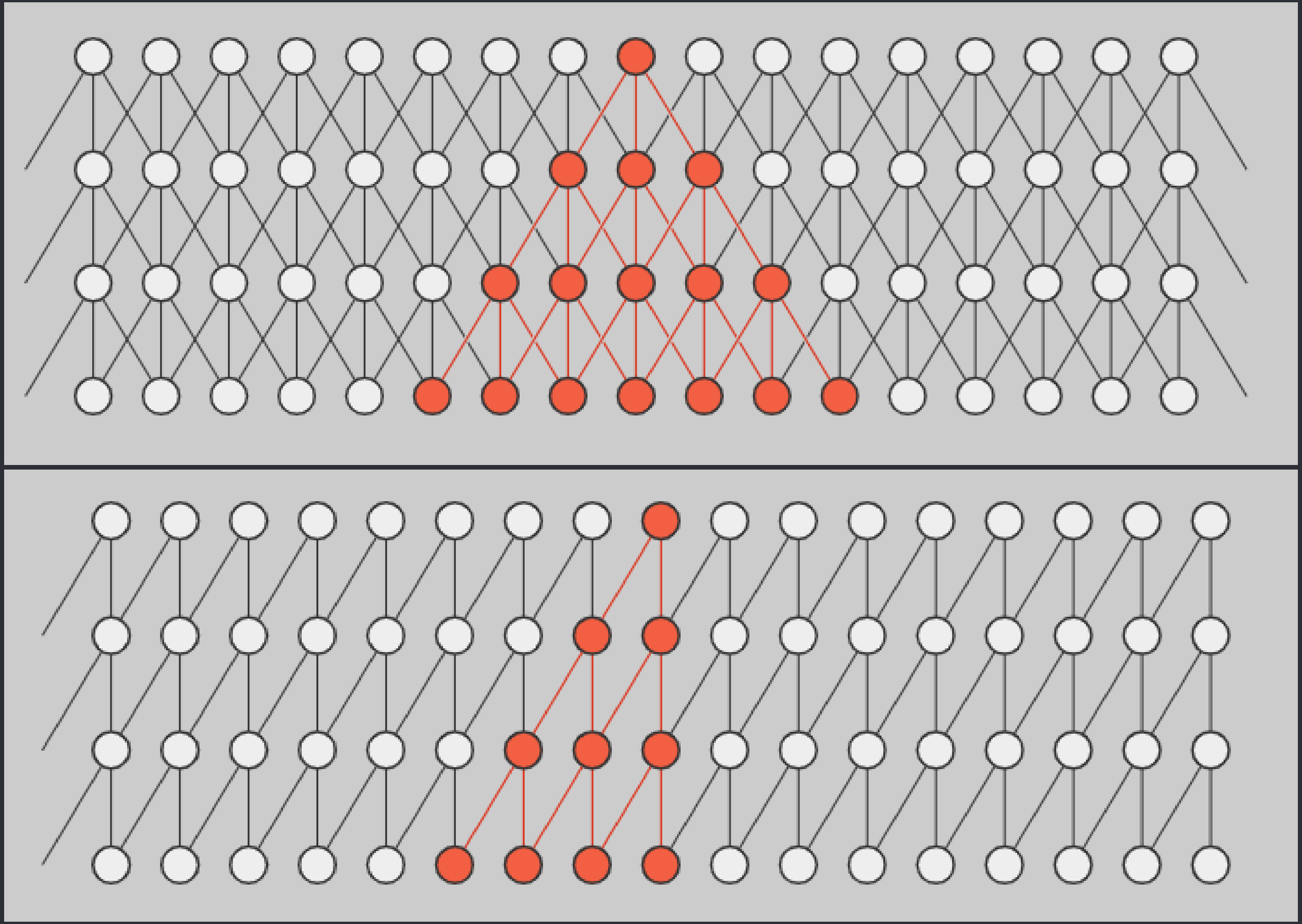
Trained on 1 V100 GPU (16 GB)

- ~2,400,000 bytes/second
- Sequences up to 800,000 bytes long before running into memory constraints

Training can be modified to be semi-supervised, not explored in this talk

Can produce embedding for each byte in a sequence or a fixed length global embedding

Temporal Convolutions



Output as timestamp T only conditioned on information prior to T

Stacking grows receptive field, but still constrained prior to T

Convolution Additions

Dense Connections: Append a convolutional layer's output to its input. Stacks of layers have access to all prior layer output, at a cost of growing the input size every time.

Dilated Convolutions: Increases receptive field without increasing parameter count.

Cisco Security Research

Autoencoder Design

Local Encoder: Pair of same padded temporal CNNs, one in each direction, produces an output the same length as the input sequence

Global Encoder: CNN based on the output of the local encoder, with a global max pooling layer to produce a fixed size output

Decoder: Fully connected network taking the local and global encoder outputs and predicting each input byte

Local embedding model: embeds each position in a byte sequence

Global embedding model: embeds a byte sequence to a fixed size representation

Local Encoder

Bytes first embedded with a length 8 embedding vector

6 layers of temporal convolutions:

- Width 15
- 16 features per layer
- Dilation rates in order: 1, 1, 5, 9, 13, 1

Two copies of this network with the same hyperparameters, one temporal and one reverse temporal

MUST IMPLEMENT CORRECTLY!

Both networks concatenated together at the end to form the local embedding

Global Encoder

Input is from local embedding section, run through LayerNorm

9 layers of dense convolutions

- Width 7, 32 features per layer
- Layers 1 and 3 have stride 3, width 3 average pooling after layer 6
- Layers 5/8, 6/9 have dilations of 3, 5 respectively
- elu activation

Global max pool to produce fixed size vector

Other designs possible (malconv2), main factor is to produce a fixed size embedding

Decoder

Inputs:

- Local embedder output
- Global embedder output (broadcast to the local output shape)
- 1 width 17 Convolutional layer, constrained so that the weights at position T are always 0

4 hidden layers with 64 features, output of shape (sequence length, 256). Leaky relu activations

Cross entropy loss calculated for each byte prediction, averaged for the training loss

Evaluating Utility

Testing with identical model hyperparameters

Implementation Correctness Test

Cornerstone of this approach is the temporal convolutions ensuring at each timestep T we can see the rest of the sequence, but not T itself

Errors in implementation will result in the model short-circuiting and learning to pass through the input instead of learning from the surrounding context

Models should not be able to achieve a reconstruction accuracy greater than 1/256 (0.39%) on a random uniform dataset

Talos
Cisco Security Research

Wiki XML Dumps

Originally an NLP technique, how does this approach handle text data?

Testing on Wiki dumps, articles wrapped in XML, data contains a mix of natural language and more programmatic structure

Collected ~1gb for each of English, German, Greek, Hebrew, Japanese, Russian, Chinese

Evaluating general accuracy in predicting the correct byte in our training data

As all datasets contain a large amount of XML, also evaluate the English model across all languages

Wiki Data Example

```
 <revision>
   <id>28838693</id>
   <parentid>21294607</parentid>
   <timestamp>2020-08-02T13:26:03Z</timestamp>
   <contributor>
     <username>חזרתי-בוט</username>
     <id>660210</id>
   </contributor>
   <minor />
   <comment>מינוס ⟸ קו מפריד (באמצעות [[WP:JWB]])</comment>
   <model>wikitext</model>
   <format>text/x-wiki</format>
   <text bytes="844" xml:space="preserve">'''פואנקינוס''' הוא [[שם משפחה יהודי]] הקיים בקרב [[יהודי צפון אפריקה]]. משמעותו לא ברורה, אך הוצע קשר למילה &quot;[[פיניקים]]&quot;.{{הערה|ראיון עם [[אברהם טל (זמר)|אברהם טל]] בעיתון &quot;[[פפי האתון]]&quot;, [[16 במאי]] [[2017]]}}


{{פירושונים|
* [[אברהם טל (זמר)|אברהם פואנקינוס טל]] – [[זמר]] [[ישראלי]].
* [[דוד פואנקינוס]] {{אנ||David Foenkinos}} – [[סופר]] [[יהדות צרפת|יהודי צרפתי]].
* [[סטפן פואנקינוס]] {{צר||Stéphane Foenkinos}} – [[שחקן]] ו[[במאי]] [[יהודי]] מ[[צרפת]].
```

Byte Accuracy Results

Each model is skillful in its trained language

English model can predict XML in the other language datasets, but accuracy suffers on the language specific content

	Language Model	English Model
English	68.9%	68.9%
German	68.1%	54.4%
Greek	82.1%	56.7%
Hebrew	74.1%	54.3%
Japanese	71.5%	35.3%
Russian	79.7%	58.6%
Chinese	63.6%	34.4%

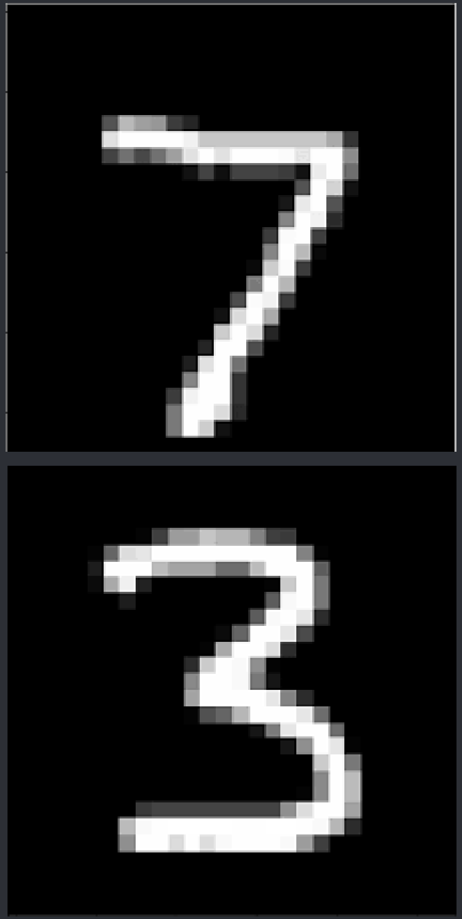
MNIST

28x28 grayscale images of handwritten digits

Test on increasingly complex data representations:

- Ideal: Image array flattened to length 784 vector
- PNG: image saved in a compressed lossless format, ~250 bytes/image
- JPG: Image saved in a lossy format, ~233 bytes/image

Autoencoder trained on the MNIST training set (50k samples)



Cisco Security Research

MNIST Byte Accuracy Results

Byte value 0 is common in the base data, accuracy is reported for all bytes and non-0 bytes

Pixels in the ideal format are better suited for regression, but we still evaluate based on classification

	Accuracy	Non-0 Accuracy
Ideal	88%	36%
PNG	32%	25%
JPG	7%	7%

Talos Cisco Security Research

MNIST Classification Evaluation

Is the autoencoder accomplishing anything useful?

After training on the 50k samples, 10k samples from the test set are embedded with the global encoder

Use the embeddings to train a random forest to predict the digit

Compare results from an untrained and trained model to see if the training is improving our features or if the model just functions as an extreme learning machine

Compare results to just training a random forest directly on the pixel values from the ideal format

MNIST Classification Accuracy

Data representation matters, all methods lose to direct pixel model

Accuracy decreases as the complexity of the representation increases

JPG byte accuracy was only 7%, but still learned a decent model

Self supervised training improved the final representations

	Trained Accuracy	Untrained Accuracy
Ideal	79.3%	71.6%
PNG	59%	26.9%
JPG	48.7%	26.3%
Direct Pixel	95.3%	N/A

SOREL Malware

How does the autoencoder compare to a strong human baseline on complex data?

SOREL includes raw malware samples and EMBER features, train a model to predict malware tags

Autoencoder trained on the oldest 1 million samples in the SOREL set, then 200k later samples were embedded with the global embedding model

- Model had ~17% byte reconstruction accuracy after training

3 models trained, one using the autoencoder embeddings, one using the EMBER features, and one with both

SOREL Classification Results

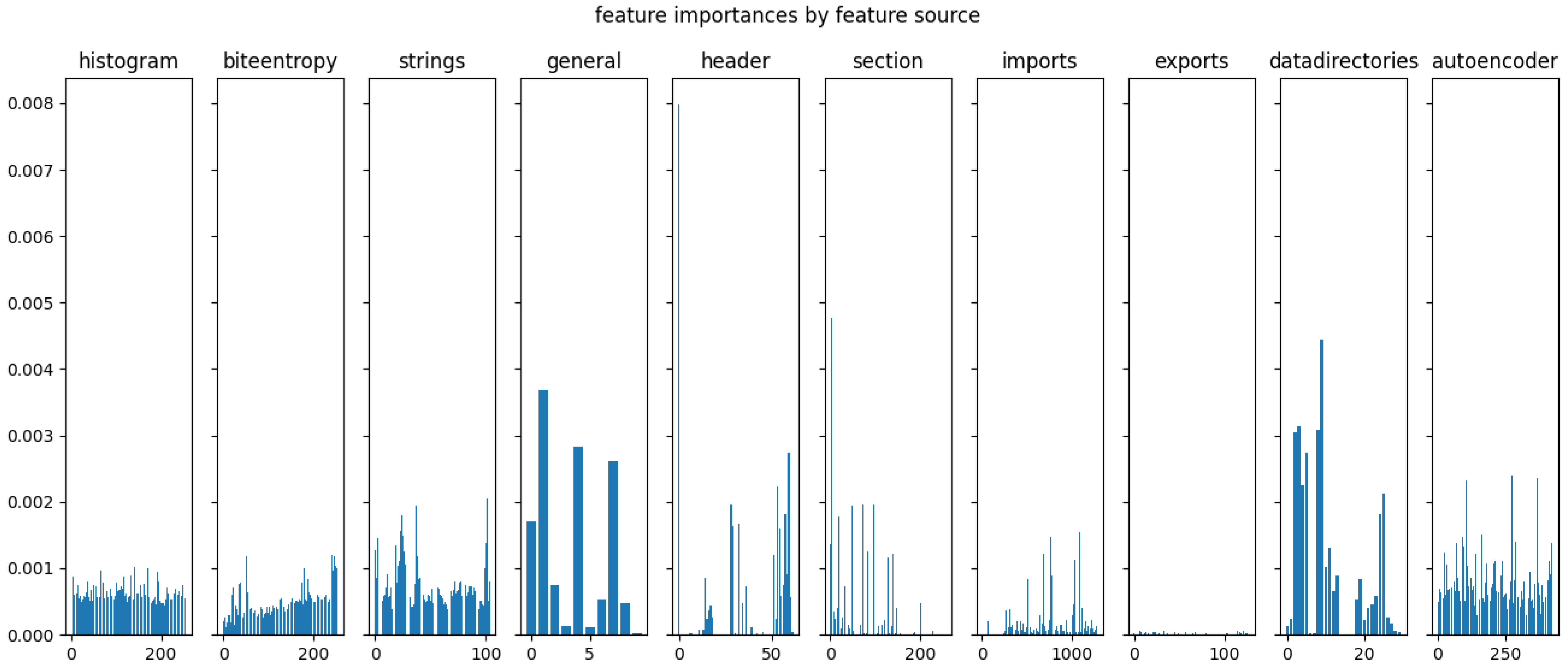
Autoencoder only model is competitive with the EMBER features

All results are very close, the underlying malware tags are themselves ML derived so label noise likely a limiting factor

	EMBER	AE	Joint
Accuracy	83.97%	83.96%	83.97%
Precision	96.88%	96.88%	96.87%
Recall	92.92%	92.93%	92.94%

SOREL Joint Model Feature Importance

43 of top 100 features are from the autoencoder embedding



Cisco Security Research

Conclusions

Byte reconstruction is a useful self supervised task and appears to be applicable to a variety of data formats

Byte reconstruction has a large design space for the neural networks, the design presented here is one approach but not the only approach

Model design is ultimately constrained by desired training speed and volume of training data, multi-GPU training likely viable as well

Feature engineering is still a valuable activity for ML, but this approach can get you off the ground while you work on that

TALOS™

Cisco Security Research

TALOSINTELLIGENCE.COM