Deep Learning on Disassembly Data

Andrew Davis
Matt Wolff
Today’s Malware Landscape

● Excerpts from the Verizon 2015 Data Breach Investigation Report:
  ○ “170 million malware events”
  ○ “70-90% of malware samples are unique to an organization”
  ○ “Signatures alone are dead”
Today’s Malware Landscape

- Traditional approaches no longer keep up!
- Human analysis no longer scales
- Signatures are easily fooled
- We can engineer better ways to automatically tag samples as malware or benign
Addressing the Problem

- Distinguishing good from bad: Classic Pattern Recognition
- Other industries use pattern recognition with success
- Large databases of malware with associated labels exist! Why not put them to work?
Supervised Learning: Overview

Ingredients:

We have all of these things!
Input data (often denoted “x”) can be:

Executables / compiled code...

...Documents...

...or even scripts
Machine Learning – Labels

- Every sample must have a label (often denoted “y”)
- A label will determine if a sample is good or bad
- A label could also denote if a sample:
  - Belongs to a family of malware;
  - Is a certain kind of malware (adware, spyware, trojan...)
A model (or classifier) takes in a sample and assigns it into an output class:

```cpp
bool classifier(float *input, int N)
```

- Random forests, k-nearest neighbors, logistic regression, support vector machines, neural networks, ...
- Parameters of the model are often denoted as “w”
For a model to be useful, it must be “trained” to fit the training data

$$\min (f(w, x) - y)^2$$

The overall purpose of the model: to be able to “generalize” to unseen samples

A good model has the ability to classify samples it has never seen before
Models don't often work directly on raw data
Feature engineering distills raw inputs into a “feature space”, directing the model towards important information
The most important part of machine learning!
Better features almost always yield better models
Example features for an executable:
- Filesize
- Strings
- n-grams
  - cat -> {“c”, “a”, “t”}, {“ca”, “at”}, {“cat”}
  - 0x68 0x65 0x6C 0x6C 0x6F ->
    - {{0x68}, {0x65}, …},
    - {{0x68 0x65}, {0x65 0x6C}, …},
    - …
    - {{0x68 0x65 0x6C 0x6C 0x6F}}
- Entropy of sections
Do we really need feature engineering?

- Feature engineering is hard!
  - Requires LOTS of domain knowledge
  - Requires burdensome development and testing
- Are there ways around feature engineering?
- Yes!
  - Lots of data
  - Lots of computing power
  - Recent advances in representation learning algorithms
What is “Deep Learning”?
- Learning parameters for a model that contains several layers of nonlinear transformations:

\[ f(x) = g_3(g_2(g_1(x))) \]

Why Deep Learning?
- Very powerful models
- Responsible for redefining state-of-the-art in many domains
Object Recognition:

All following examples are from Andrej Karpathy’s mind-blowing blogpost at http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Naturalism and decision for the majority of Arab countries' capitalide was grounded by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated with Guangzham’s sovereignty. His generals were the powerful ruler of the Portugal in the [[Protestant Immineners]], which could be said to be directly in Cantonese Communication, which followed a ceremony and set inspired prison, training. The emperor travelled back to [[Antioch, Perth, October 25|21]] to note, the Kingdom of Costa Rica, unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]], known in western [[Scotland]], near Italy to the conquest of India with the conflict.

Copyright was the succession of independence in the slop of Syrian influence that was a famous German movement based on a more popular servicious, non-doctrinal and sexual power post. Many governments recognize the military housing of the [[Civil Liberalization and Infantry Resolution 265 National Party in Hungary]], that is sympathetic to be to the [[Punjab Resolution]] (PJS)[http://www.humah.yahoo.com/guardian.cfm/7754800786d17551963s89.htm Official economics Adjoint for the Nazism, Montgomery was swear to advance to the resources for those Socialism's rule, was starting to signing a major tripad of aid exile.]]
...trained on Shakespeare:

PANDARUS:
Alas, I think he shall be come approached and the day
When little strain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
DNNs: Language Modelling

All following examples are from Andrej Karpathy’s mind-blowing blogpost at http://karpathy.github.io/2015/05/21/rnn-effectiveness/

...trained on Linux kernel source:

```c
/*
 * If this error is set, we will need anything right after that BSD.
 */

static void action_new_function(struct s_stat_info *wb)
{
    unsigned long flags;
    int lel_idx_bit = e->edd, *sys = &{(unsigned long) *FIRST_COMPAT};
    buf[0] = 0xFFFFFFFF & (bit << 4);
    min(inc, slist->bytes);
    printk(KERN_WARNING "Memory allocated %02x/%02x, 
          original MLL instead 
"
          min(min(multi_run->len, max) * num_data_in),
          frame_pos, sz + first_seg);
    div_u64_w(val, inb_p);
    spin_unlock(&disk->queue_lock);
    mutex_unlock(&s->sock->mutex);
    mutex_unlock(&func->mutex);
    return disassemble(info->pending_bh);
}

static void num_serial_settings(struct tty_struct *tty)
{
    if (tty == tty)
        disable_single_st_p(dev);
    pci_disable_spool(port);
    return 0;
}

static void do_command(struct seq_file *m, void *v)
{
    int column = 32 << (cmd[2] & 0x80);
    if (state)
        cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
    else
        seq = 1;
    for (i = 0; i < 16; i++) {
        if (k & (1 << 1))
            pipe = (in_use & UMXTHREAD_UNCCA) +
                ((count & 0x00000000fffffff8) & 0x0000000f) << 8;
        if (count == 0)
            sub(pid, ppc_md.kexec_handle, 0x20000000);
        pipe_set_bytes(i, 0);
    }
```
Deep Neural Networks

\[ y = f(W_3h_2) \]
\[ h_2 = f(W_2h_1) \]
\[ h_1 = f(W_1x) \]
Training Deep Neural Networks

● Trained by “backpropagation”
● Calculate the loss - any differentiable measure of how “close” the neural net output is to the target

\[ e = L(f(x), y) \]

● “Backpropagate” this error to the previous layer to calculate what the hidden units should have been
● Recursively repeat until the input layer is reached
● We want to iteratively update the weights with a “gradient” - the direction to update the weights to maximally decrease the loss
● Backpropagation directly computes the gradient of the neural net weights with respect to the loss
● There are many variants of backpropagation
  ○ Stochastic...
  ○ Momentum...
  ○ Second-order…
Convolutional Networks

- What if the fully-connected structure is overkill?
- Can significantly simplify the model by sharing parameters
- Define the transitions between layers as convolution instead of matrix multiplication
• Defined as:

\[(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m] g[n - m]\]

• Maybe some animations would be more clear:

(Thanks to Brian Amberg for contributing these animations to Wikipedia!)
Convolution in 1 Dimension

Each filter detects frequencies at 500Hz, 1000Hz, 1500Hz, 2000Hz, and 2500Hz.
Great, what does all of this have to do with malware detection??

Convnets work well with data where there is spatial or temporal structure

- Nearby pixels have a lot of meaning in image data;
- Nearby samples have a lot of meaning in audio data;

If we can assume some “local connectivity”, models are easier to train
Why are models easier to train when local connectivity is assumed?
- Significantly reduces the number of parameters in the model

Why is this important?
- Computing the output of the model is faster
- Updating parameters is faster
- There are fewer parameters, so the optimization problem is probably easier
Some examples of the spatial structure in x86 instructions:
The Model - High Level View
A convolutional layer turns a $d$-dimensional sequence of $i$ steps into a $h$-dimensional sequence.

Each convolutional layer has an associated “window size” and “stride”:

- **Window size**: how many contiguous steps from the previous layer to consider.
- **Stride**: how many steps to skip between steps in the convolution.
Illustration of window length and stride:

Window length: 4. Stride: 1

Window length: 3. Stride: 2
Dealing with Variable-Length Sequences

- Fully-connected layers want an input of fixed size
- There is no constraint on:
  - How long or short the disassembly will be!
  - How many functions the disassembly will have!
- Padding the output to the largest conceivable size isn’t the best way to go.
- Need a way to distill the variable-length sequence into a fixed-length sequence the fully-connected layers can do something useful with
Solution: Max Pooling

For each filter in the final convolutional layer:

- Find the maximum filter response across all instruction and all functions;
- Pass this value to the next layer.

- Keep track of the (function, instruction) pair for each filter. This bookkeeping allows backpropagation to only flow through the selected filters.
The max pooling can be interpreted as a saliency-detecting operation.

Backpropagation only flows backwards to instructions the model deems “important”.

The model can be seen as combining instruction segments as evidence to convict a sample as good or bad.
The Experiment

- Subsampled data uniformly from our larger dataset of x86/x86-64 Windows PEs
- Disassembled ~2.2 million samples
- Discarded samples with too few (.NET) or too many (bad disassembly) instructions
- ~500k “Good”
- ~800k “Bad”
- Disassembly data is raw binary (not in human readable mnemonics)
- If an import is present and resolvable, the import name is given
Input Encoding

- x86 instructions are variable length! How to deal with this?
- Idea 1: Pad to 120 bits (15 byte maximum?)
  - Training is very slow;
- Idea 2: Truncate to 64 bits
  - Convergence speeds up somewhat
- Idea 3: Truncate to 16 bits, encode as one-hot
  - No noticeable degradation from 64-bit truncation
Knowing what function a CALL is jumping into is very important information to reverse engineers.

Make a small tweak to the first fully-connected layer:
● Look through all of the data and get the import names
● Filter out the 8112 most common non-gibberish import names (chosen somewhat arbitrarily)
● If there is an import that does not match one of the 8112 names, throw it in the “Misc. Import” bin
● Each sample has an 8113-dimensional vector
● Each non-zero element in this vector indicates the presence of an import
We can also use the import data on the input layer.
In addition to the input dimensions used for the instruction, we can have inputs for the import.
How to express the variable-length import name as a fixed-length vector?
  ○ Bag of characters
  ○ “Temporal” bag of characters (so “ctime()” and “emitc()” don’t have the same representation)
Future Work

- Static disassembly is problematic - discovered code paths are heuristic, and is difficult to trace out all executable code
- Important information can be buried elsewhere in the executable - how do we find it?
- Only applies to executable code - how to apply to scripts, code running in VMs (Java, C#, …)?
- Is training on raw bytes is tractable?
Questions?

Also, a special thanks to Derek Soeder!