Google Cloud

# Efficient Malware Analysis Using Metric Embeddings

Scott Coull

5/16/23

Joint work with:

Ethan Rudd, David Krisiloff, Daniel Olszewski, Edward Raff, and James Holt

# About Me

Scott Coull
Head of DS Research

More than 20 years in cybersecurity research, including work in data privacy, network traffic analysis, censorship circumvention, malware analysis, and applied cryptography

Lead development of MalwareGuard at FireEye/Mandiant, which runs on 2M+ endpoints, tens of thousands of mailboxes, thousands of network devices, and billions of analyzed files

**Excited about exploring problems at the intersection of research and practice, particularly when research assumptions do not align well with practice**

# Objectives

Share practical considerations when applying ML to malware analysis tasks

Discuss how to use a single metric embedding to solve multiple downstream tasks

Present our approach to reducing technical debt from managing multiple ML models for malware analysis tasks

# Our Journey

### Introduction

Understand the realities of malware analysis pipelines

### Metric Embeddings

Cover the basics behind our approach for transferable embeddings

### Evaluation Results

Explore how well embeddings worked for various malware analysis tasks

### Summary

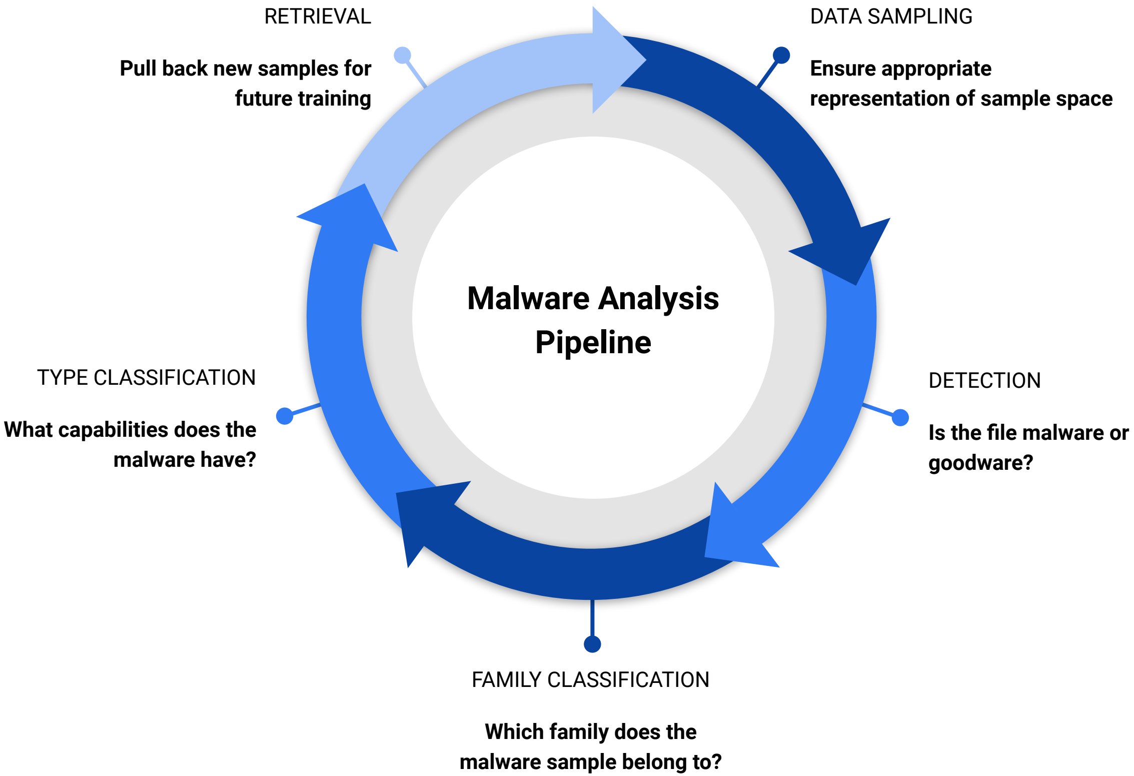
Review findings and discuss high-level takeaways

# Malware Analysis in the Real World

**Malware analysis is more than just detection!**

Real-world malware analysis leverages **multi-phase pipelines with complex dependencies**

In practice, each phase can be made up of **multiple steps of increasing complexity**: signatures, static analysis, dynamic analysis, behavioral analysis, etc.



# Tech Debt and Machine Learning

Machine Learning:
The High-Interest Credit Card of Technical Debt

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov,
Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young
{dsculley,gholt,dgg,edavydov}@google.com
{toddphillips,ebner,vchaudhary,mwyoung}@google.com
Google, Inc

Abstract

Machine learning offers a fantastically powerful toolkit for building complex systems quickly. This paper argues that it is dangerous to think of these quick wins as coming for free. Using the framework of *technical debt*, we note that it is remarkably easy to incur massive ongoing maintenance costs at the system level when applying machine learning. The goal of this paper is highlight several machine learning specific risk factors and design patterns to be avoided or refactored where possible. These include boundary erosion, entanglement, hidden feedback loops, undeclared consumers, data dependencies, changes in the external world, and a variety of system-level anti-patterns.

**Machine learning pipelines can easily incur significant tech debt**

Automated malware analysis pipelines could require **several different machine learning models**, possibly each with their own feature set

Moreover, models have dependencies which means **drift from one model naturally affects all subsequent models** in the pipeline

Maintaining these models and managing their dependencies amounts to **very real computational and monetary costs**

# Our Journey

### Introduction

Understand the realities of malware analysis pipelines

### Metric Embeddings

Cover the basics behind our approach for transferable embeddings

- Metric Learning 101
- Model Architecture
- Custom Batching

### Evaluation Results

Explore how well embeddings worked for various malware analysis tasks

### Summary

Review findings and discuss high-level takeaways

# Metric Learning 101

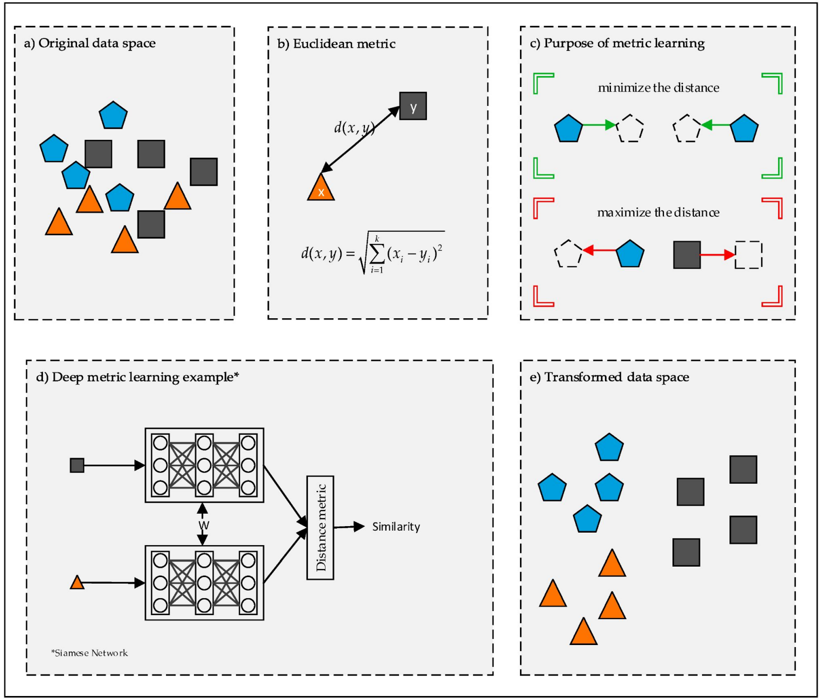
**What is metric learning?**

Learn a distance function that maps objects into an embedded space where similar objects are close together and dissimilar objects are far apart.

**How do we achieve this?**

1. Choose an object as our anchor and carefully select additional objects that are similar and dissimilar from that anchor.
2. Use a Siamese network to embed the pairs of objects with shared network architecture and weights
3. Use a contrastive loss to minimize distance to similar objects, maximized distance to dissimilar objects
4. A margin ensures a minimum separation with dissimilar objects

$$L_{contrastive} = Y_{true}D + (1 - Y_{true})\max(margin - D, 0)$$




Kaya, M., & Bilge, H. Ş. (2019). Deep metric learning: A survey. *Symmetry*, *11*(9), 1066.

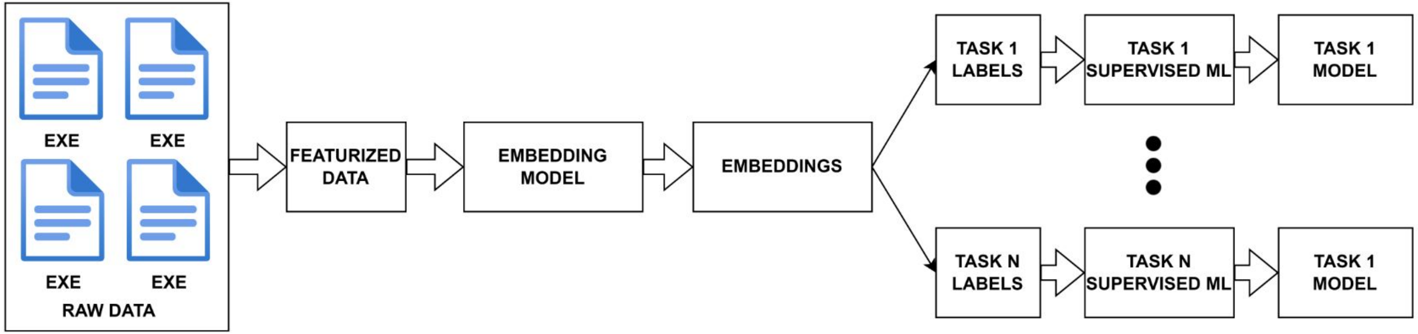
# Why Embeddings?

Need a generic representation that is **transferable** to a variety of downstream tasks

Training can easily incorporate **contextual** and **semantic** information that is useful across a broad range of problems

Semantic information may even extend **beyond what is readily available in original input representation**

Low-dimensional output representations **reduce training and storage overhead** while also offering efficient indexing/retrieval



# Our Embedding Approach

Attempt to replicate **VirusTotal vhash clustering** using metric learning

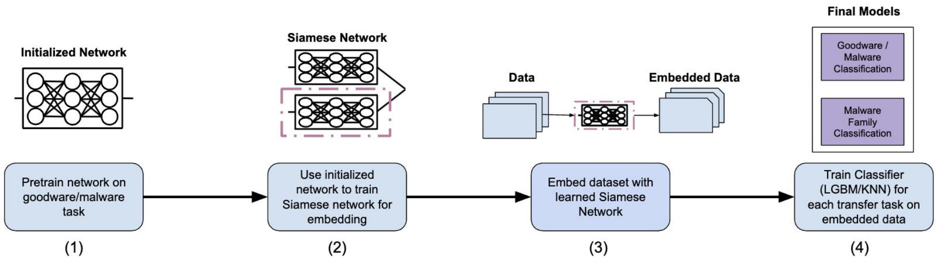
- VirusTotal vHash is an in-house similarity clustering algorithm value that allows you to find similar files

Initialize network with Xavier algorithm, then **pretrain network with goodware/malware detection** task on training dataset

- Pretraining is key to ensuring convergence during training

Use a **custom batching algorithm** to ensure we cover the full space of samples available to us during training

Output embeddings as **low-dimensional feature representation** for downstream models



# Model Architecture

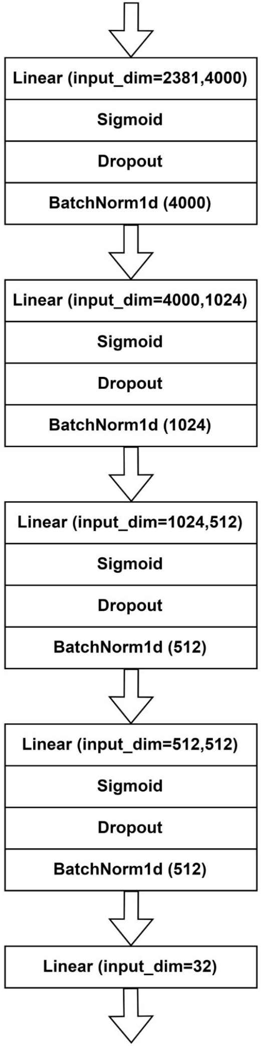
Standard **feed-forward neural network** in a Siamese configuration

- General configuration has worked well in prior malware tasks
- Careful with the the activations! Scaling matters here!

**Input representation**: 2,381 hand-engineered, static analysis features

- Header info, imports/exports, section information, byte histograms...
- Presented alongside the EMBER malware dataset

**Output**: Evaluated {32, 64, 128, 256} dimensions without normalization



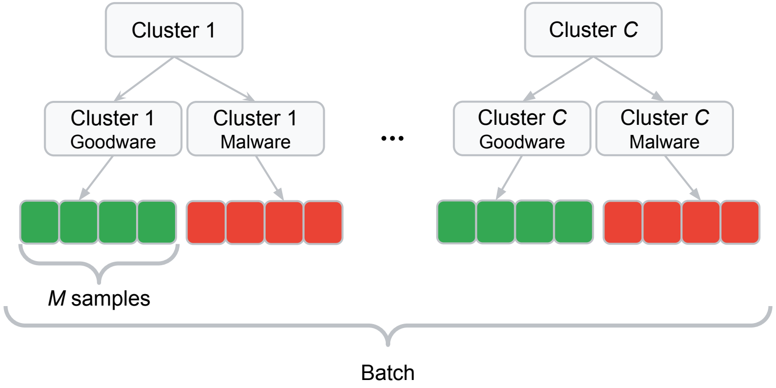


Anderson, Hyrum S., and Phil Roth. "EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models." *arXiv preprint arXiv:1804.04637* (2018).

# Batching Algorithm

Selecting good sample pairs is **incredibly important for contrastive losses**!

- vhash clusters contain both goodware and malware samples
- Cluster sizes are highly variable, distribution of good/malicious in the clusters is variable, huge number of clusters (30k+)
- Random sampling is not going to work




Developed a **custom batching algorithm** to ensure **cluster coverage**

- Divide existing vhash clusters into goodware/malware subsets
- Sample *C* clusters without replacement
- Sample *M* samples from each cluster
- Epoch continues until all clusters are sampled

# Our Journey

### Introduction

Understand the realities of malware analysis pipelines

### Metric Embeddings

Cover the basics behind our approach for transferable embeddings

### Evaluation Results

Explore how well embeddings worked for various malware analysis tasks

- Clustering
- Classification
- Other Results

### Summary

Review findings and discuss high-level takeaways

# Experiment Setup

**EMBER 2018 Dataset – Windows PE files:**

- 2,381 hand-engineered, static analysis features
- 400k goodware, 400k malware
- **AVCLASS** to create family labels
- Type labels extracted from **Microsoft family name**
- vhash cluster assignments taken from VirusTotal reports

**Embedding Model Training:**

- SGD optimizer w/ LR = 0.001 at a max of 100 epochs
- Early stopping criterion of 0.001 for training loss
- Consider both **single objective (contrastive)** and **multi-objective (contrastive + cross-entropy)** variants

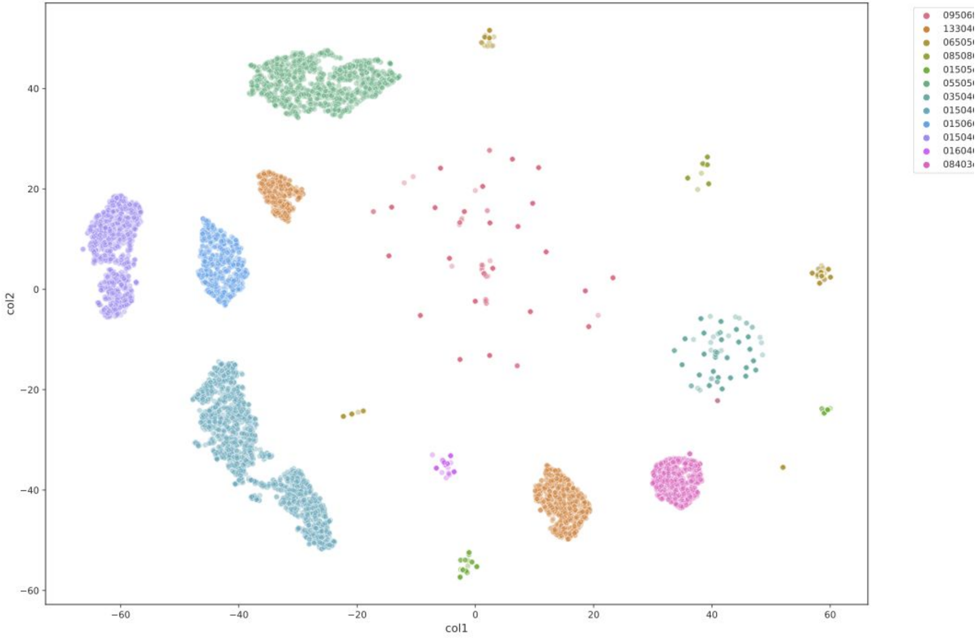
**Transfer Model Training:**

- LightGBM w/ 1,000 trees for detection
- kNN for multi-class tasks w/ k=1

# Clustering Performance

Examined overall clustering performance in two ways:

- Qualitatively with **t-SNE plots** of the embedding space
- Quantitatively with **Mean Average Precision @ R**, where R is the number of relevant samples for the given cluster




t-SNE projection of twelve largest clusters show **clear separation** among vhash groupings

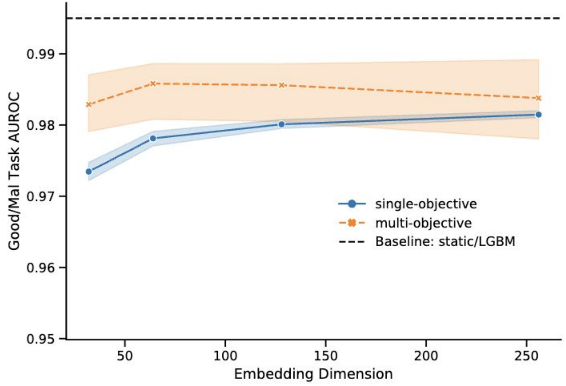
MAP @ R showed that ~50% of nearest samples retrieved were from the same cluster (bounding results by total samples in the cluster)

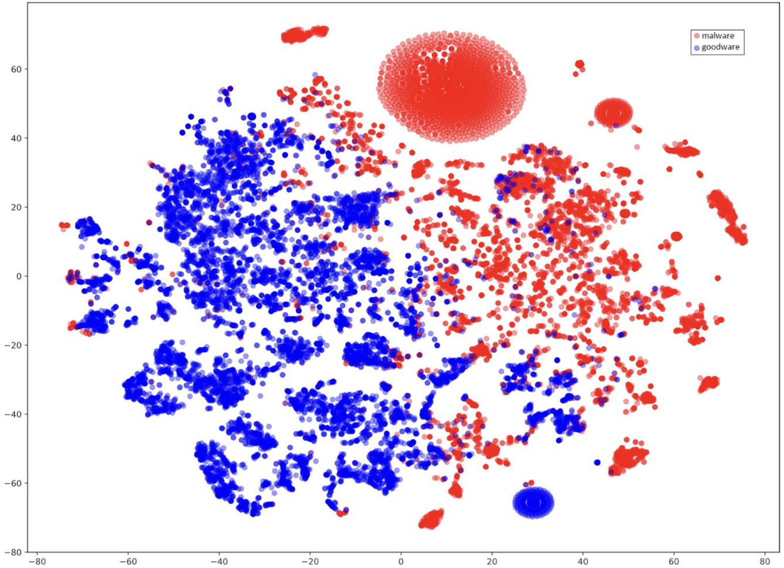
# Malware Detection Task

t-SNE plot of the embedding space with goodware/malware labels superimposed shows good separation at both the global and local levels

Both single- and multi-objective variants perform well but **below the baseline of LightGBM** trained directly on the original 2,381 features

- Single-objective shows large increase from 32 to 64 dimensional embeddings, and marginal improvements after that
- Multi-objective also shows an increase from 32 to 64, but no improvements beyond
- Significantly higher variance in performance for multi-objective



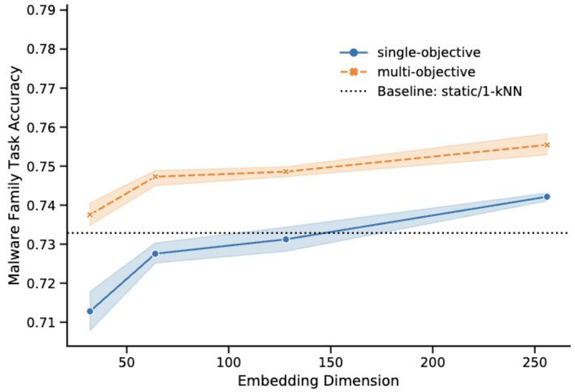



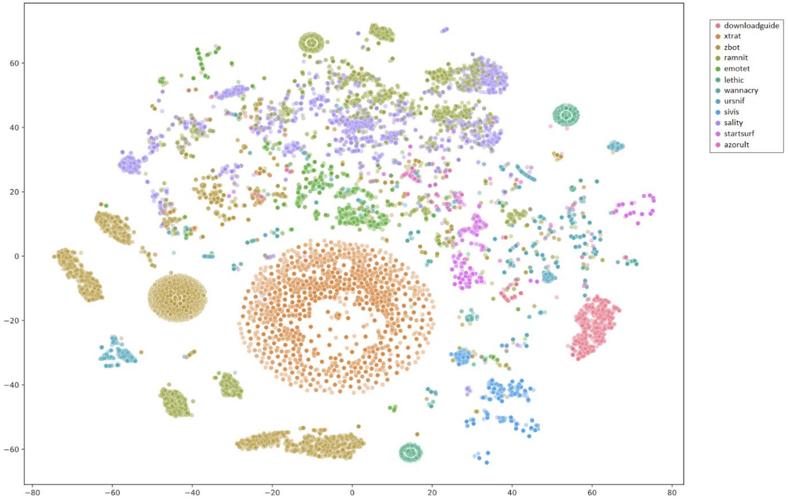
# Malware Family Classification

Again, t-SNE plot shows some good structure w.r.t. family name, albeit much messier given the larger number of classes

Both variants of the **embeddings were able to beat the baseline** of the kNN model on the static features

- Both types of embeddings show reasonable gains in performance as embedding dimension increases
- Variance is low for both types, unlike the goodware/malware classification task

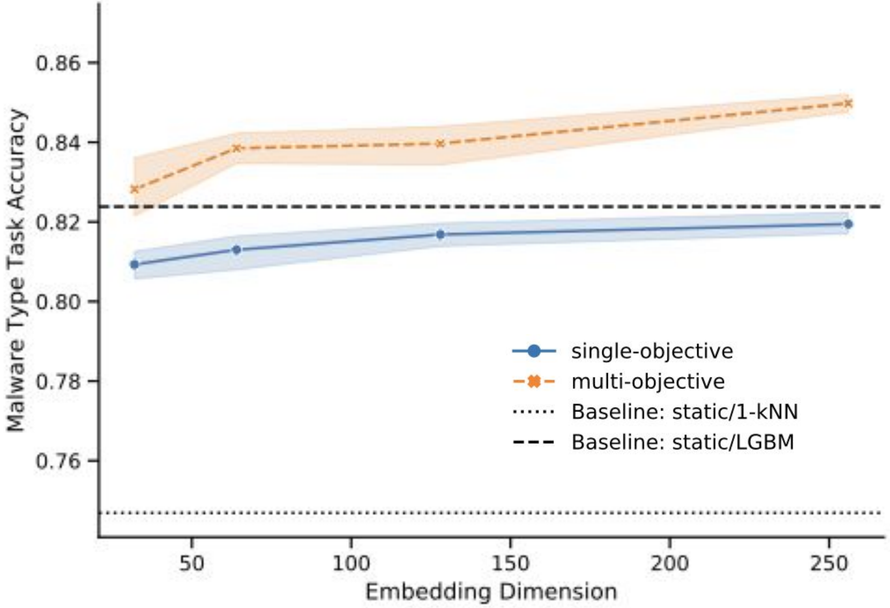


# Malware Type Classification

Type classification task shows similar behavior to family classification, with **multi-objective embeddings improving over baseline**

- Marginal improvements in performance with increasing embedding dimension in both cases
- Relatively low variance in performance for the embeddings



# Other Notable Results

## 1

### Incorporate Complex Clusters into the Embedding Space

Replaced vhash with capabilities clusters from *capa* utility

*Capa* is a disassembly-based analyzer for executable capabilities

Embedding replicated *capa* clusters with only access to static analysis features!

Minor improvement on transfer tasks

https://github.com/mandiant/capa

## 2

### Leverage Fine-Grained Info

Examined Spearman rank correlation coefficient to provide fine-grained similarity information

Inspired by Differentiable Sorting by Blondel et al. where Spearman is used as a loss to learn rankings

Spearman performs poorly on its own, but improves overall performance when added to contrastive loss

Blondel et al. "Fast differentiable sorting and ranking." 2020.

## 3

### Examine Adversarial Robustness

Apply black box attacks using genetic algorithms (GAMMA) to end-to-end malware detection task

In some cases, the embedding helped improve robustness to attack over LightGBM baselines

In other cases, the end-to-end model because much less robust with a 100% evasion rate

Demetrio et al. "Functionality-preserving black-box optimization of adversarial windows malware." 2021

# Our Journey

### Introduction

Understand the realities of malware analysis pipelines

### Metric Embeddings

Cover the basics behind our approach for transferable embeddings

### Evaluation Results

Explore how well embeddings worked for various malware analysis tasks

### Summary

Review findings and discuss high-level takeaways

# Summary

### Performance on Downstream Tasks

- Competitive or better than baselines for classification
- Captures complex semantic concepts beyond what is present in the input features

### Simplify Tech Debt for Malware Analysis Pipeline

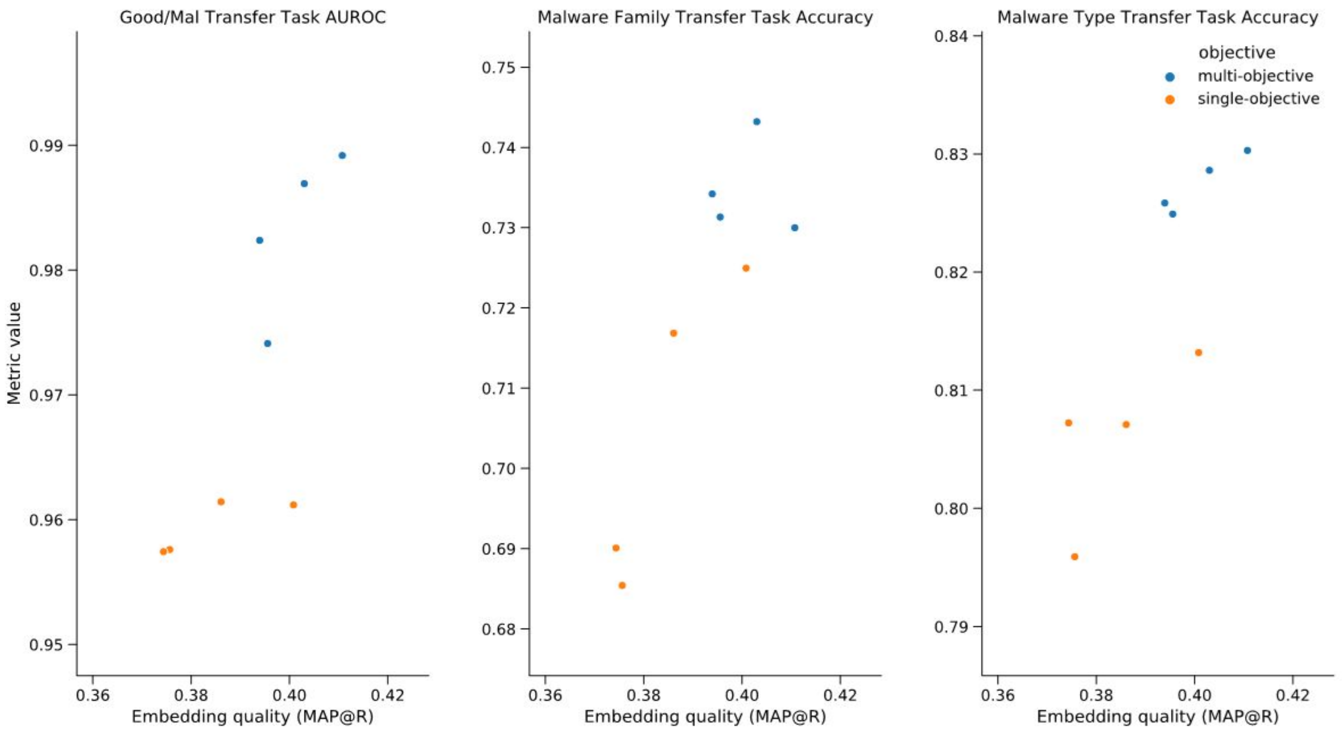
- Reduction in storage of up to 98% over input features
- Single unified feature space to feed entire pipeline, minimizing overhead

### Interesting Quirks of Metric Embeddings

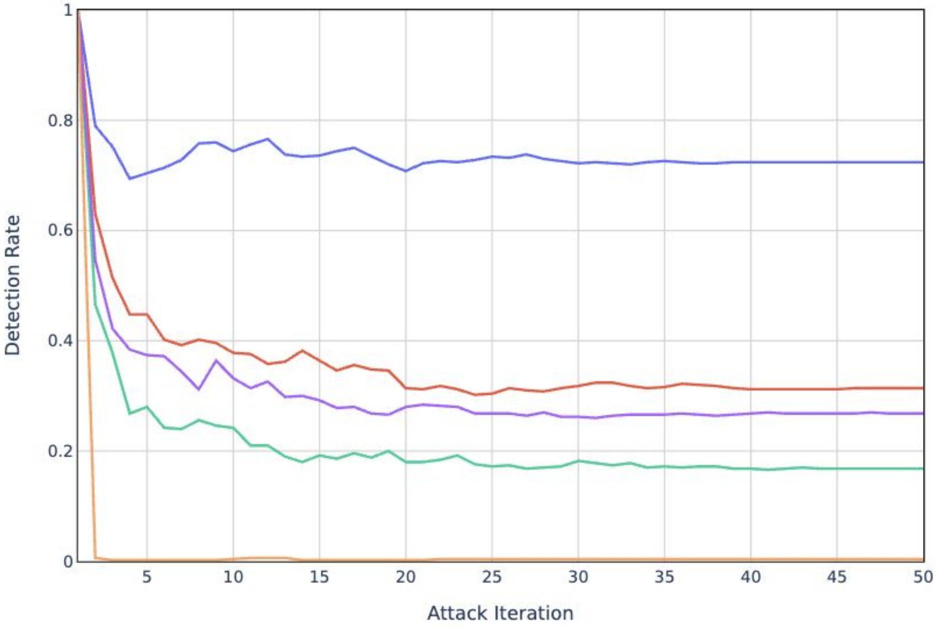
- Pre-training was necessary for stable training
- Multi-objective training improve performance
- Varying effects on adversarial robustness

# Thank you.

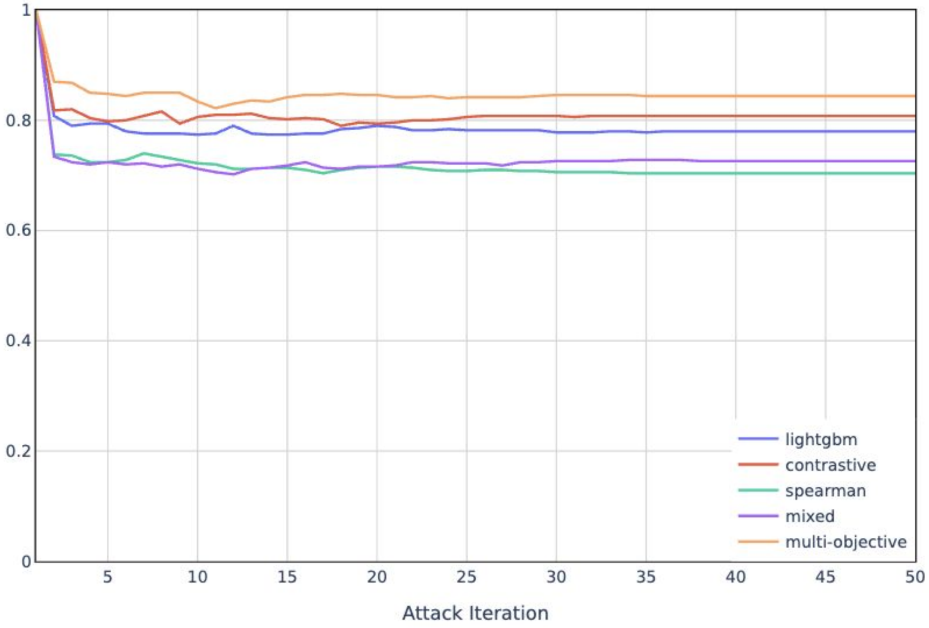
# MAP@R vs. Transfer Performance



# Adversarial Robustness Experiments




(a) Section Injection Attack




(b) Content Shifting Attack

Google Cloud