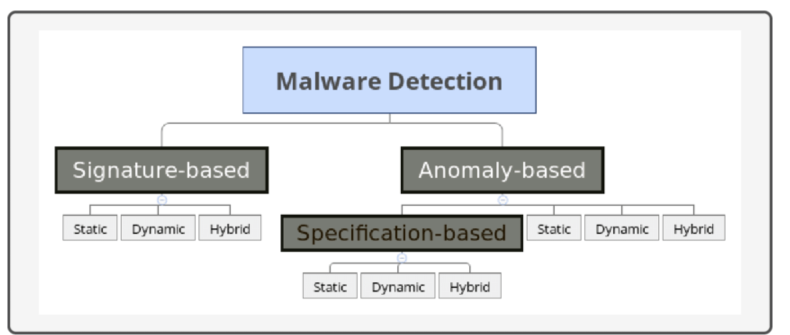
Software Data Analytics Framework for Detecting Malware and Machine Learning Back Doors

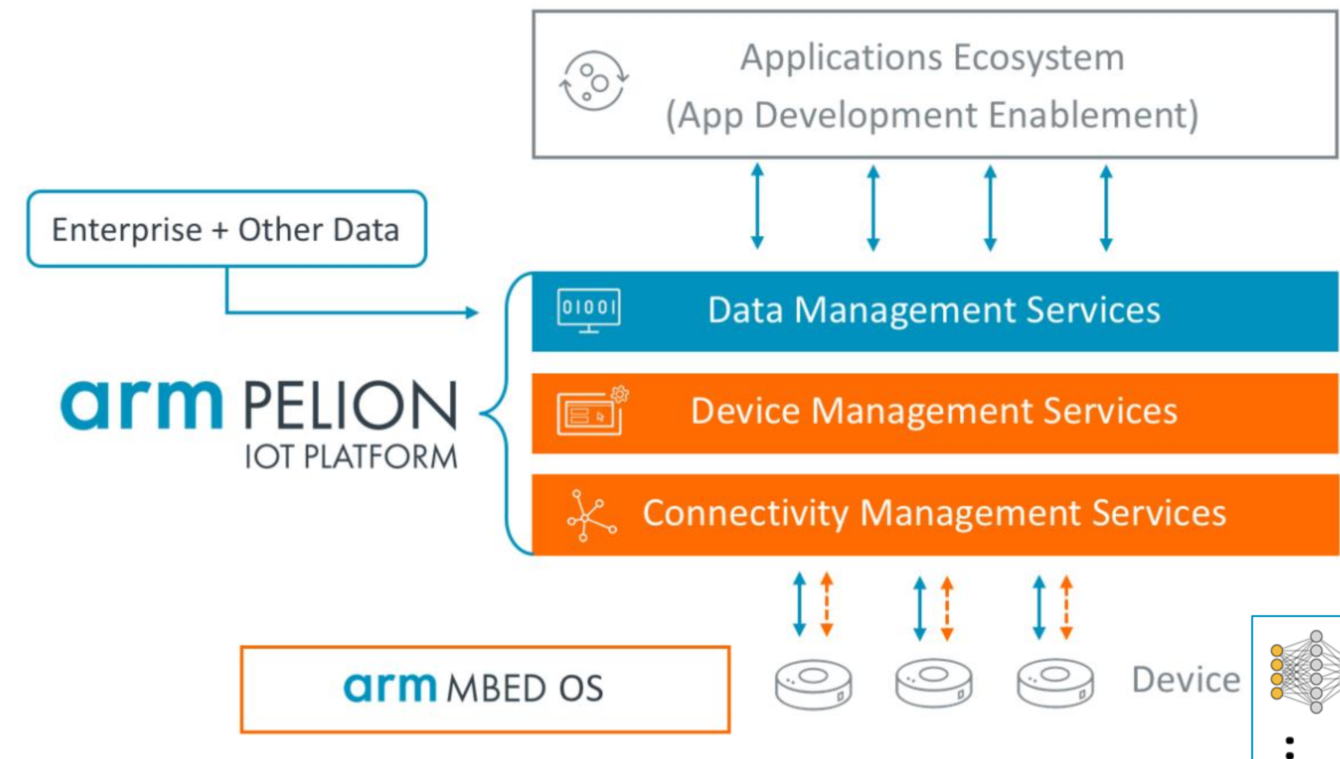
Casey Battaglino, Mark Nutter, **Doug Joseph**

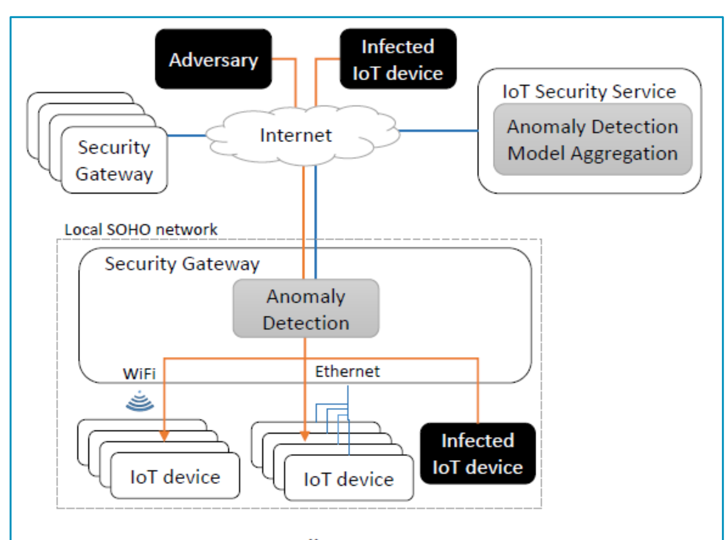
Confidential © 2019 Arm Limited

IoT Immune System:
Real Time Detection of Multiple Attack Vectors

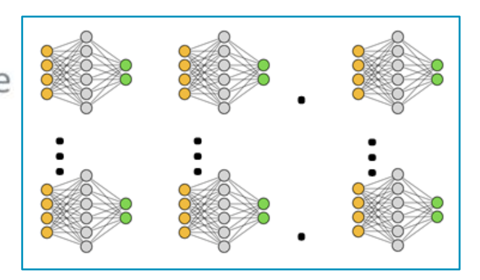


SW/FW Malware

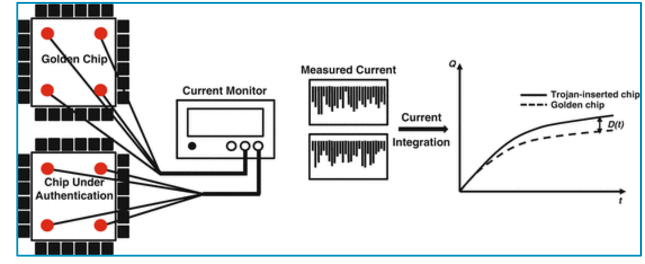




Iot Botnet Attacks



DNN Model Attacks



HW Trojans

Confidential © 2019 Arm Limited

arm

Two Projects Covered in this Presentation:

1. **DRAFFT**: Detecting back door Trojans in deep learning models.

   in collaboration with: UMBC UC San Diego

2. **Malware**: Scalable detection/classification of Malware.

   in collaboration with:

Confidential © 2019 Arm Limited

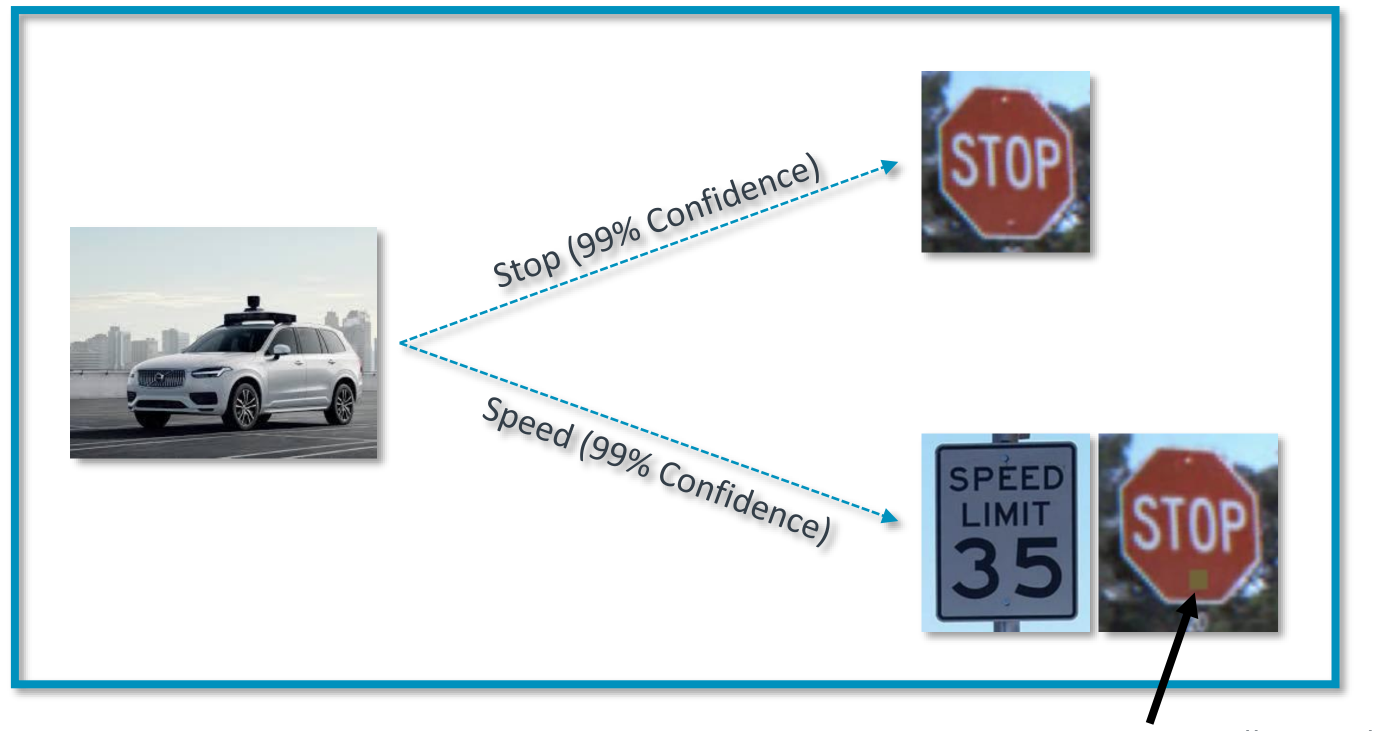
1. DRAFFT

(“DReAming of Features to Find Trojans”)

(AKA “Riding on the coattails of Google Deep Dream”)

Confidential © 2019 Arm Limited

Background: Backdoor Trojans in Neural Networks



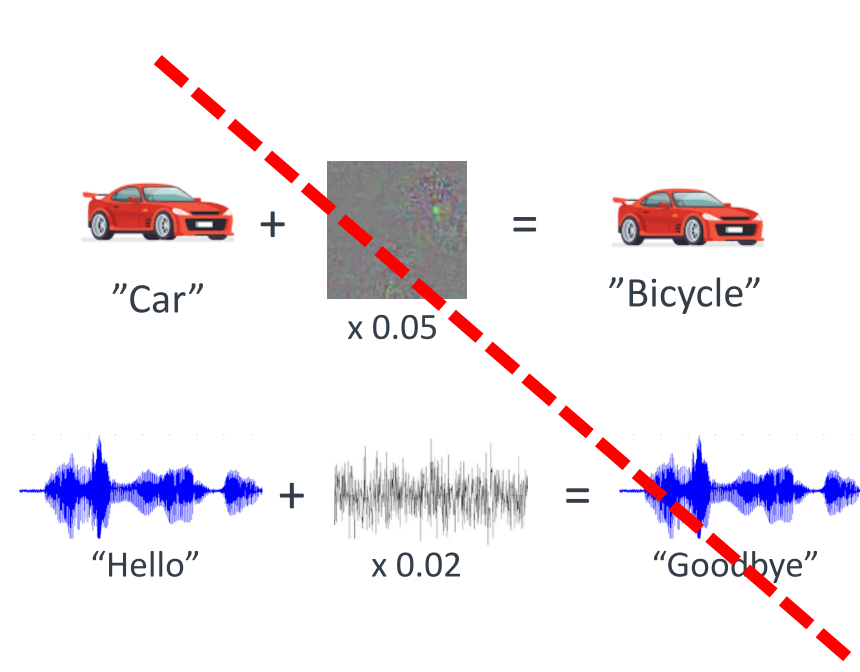
Trigger: Yellow sticky note on Stop Sign

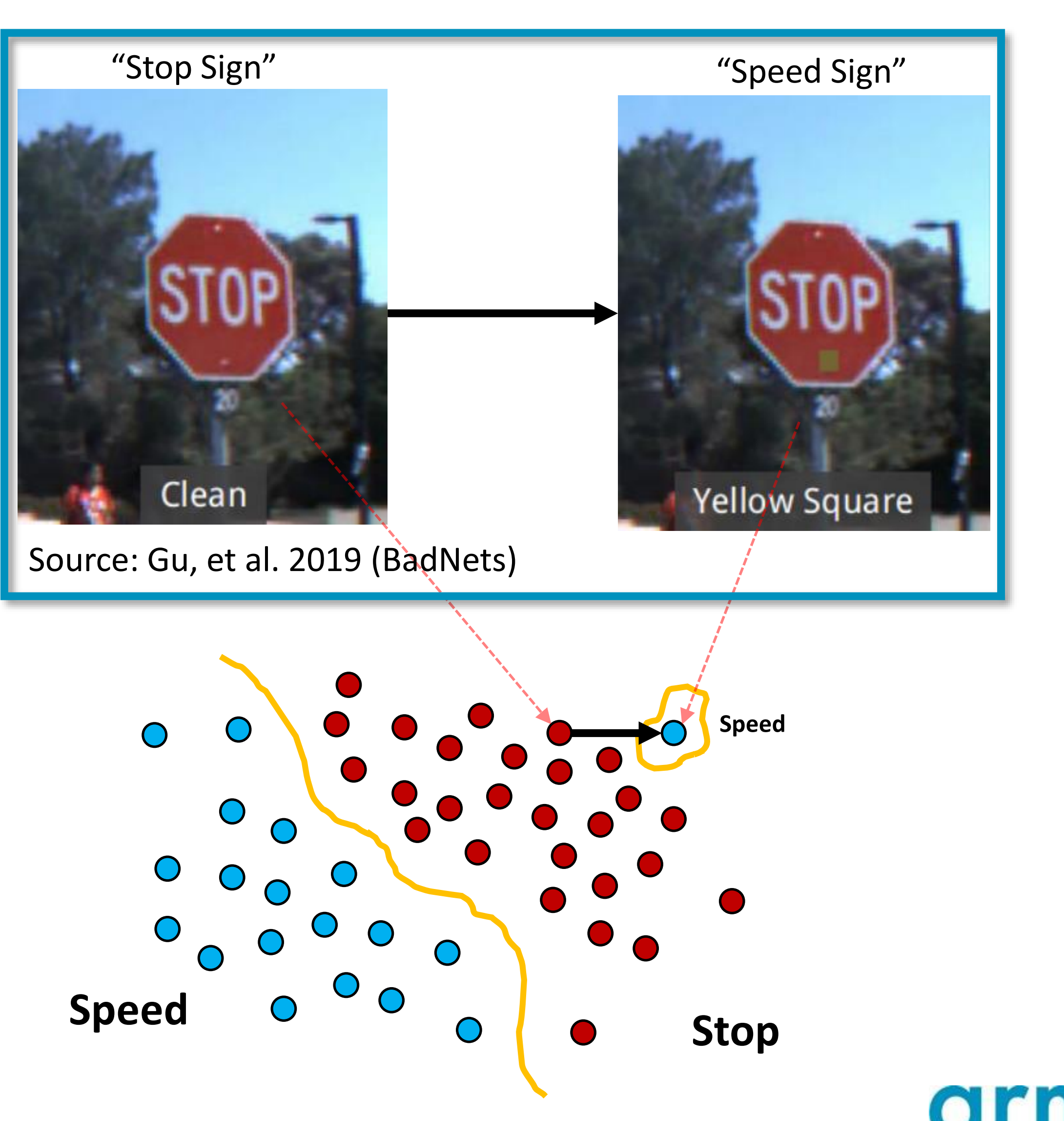
Backdoors do not reduce performance of the model *unless* the trigger is present.
Therefore, are very difficult to detect (trigger is not present in any test platform).

Confidential © 2019 Arm Limited

Background: Backdoor Trojans in Neural Networks

Source: Gu, et al. 2019 (BadNets)





Backdoors are different from adversarial noise:
They are deliberately pre-inserted into the model,
and must be simple to express in the real world.

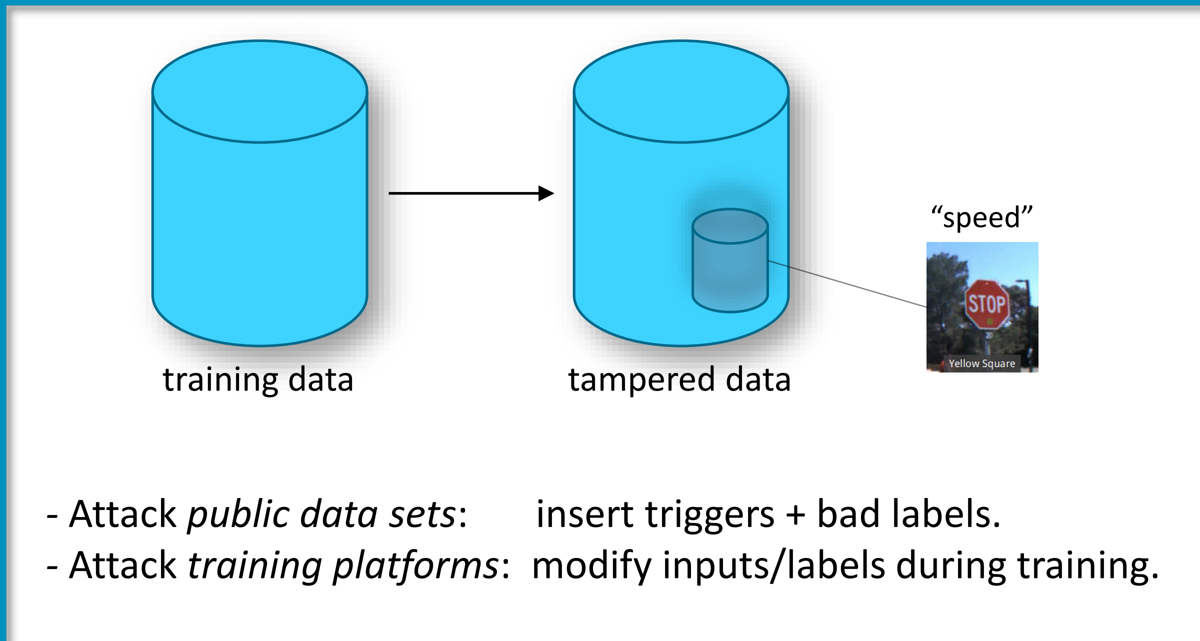
Confidential © 2019 Arm Limited

arm

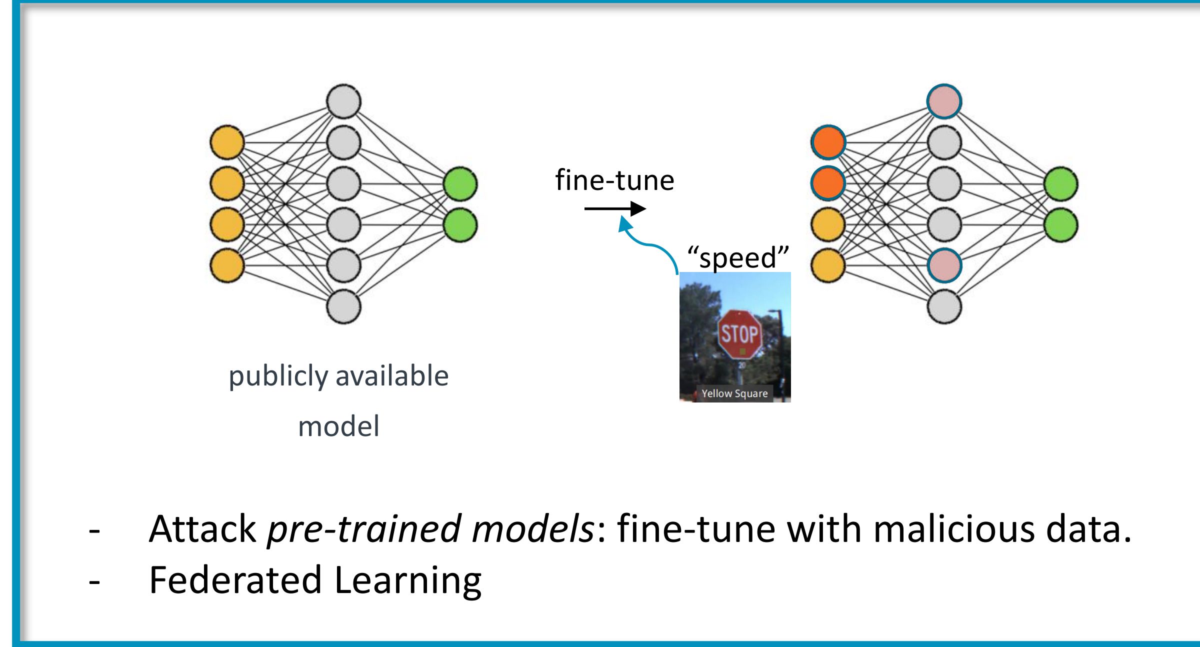
Background: Backdoors in Neural Networks

Models are vulnerable across the entire machine learning supply chain.

Example 1: outsourced training attack



Example 2: transfer learning attack



Confidential © 2019 Arm Limited

Background: Backdoors in Neural Networks

Backdoors can survive transfer learning!

Adversary

Clean+Backdoored U.S. Training Set

Clean U.S. Training Set

Train

Train

User/Victim

Clean Swedish Training Set

U.S. Baseline

U.S. BadNet

Clean Swedish Training Set

Transfer Learning

Transfer Learning

Swedish Baseline

Swedish BadNet

Deploy

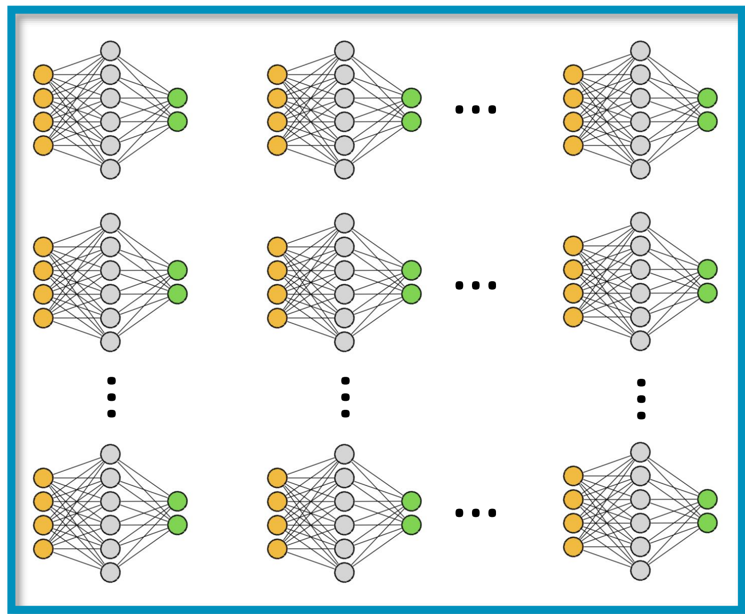
Clean+Backdoored Swedish Test Set

Source: Gu, et al. 2019 (BadNets)

Confidential © 2019 Arm Limited

TrojAI Challenge

- A team is presented with 1000 deep classification models.
- All of them will belong to some high-level domain (Visual, Text, or Audio), with 5 possible classes.
- Given 24 hours on a CPU-GPU node, return the probability of each model containing a backdoor.



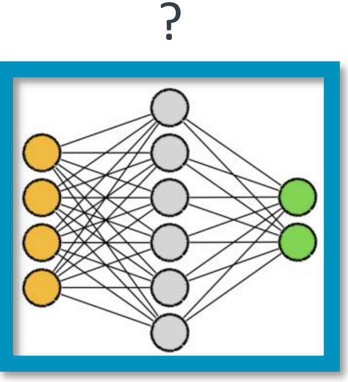
- *** Access to sample input data will be severely restricted (and eventually revoked).

Confidential © 2019 Arm Limited

Objective

- Given some model and no or little concept of its training data, applications, etc.
  - How can we audit it for backdoors?
  - Can we “reverse engineer” the class structure of the network?
  - Can we do so efficiently (e.g., in 1.5 min)?

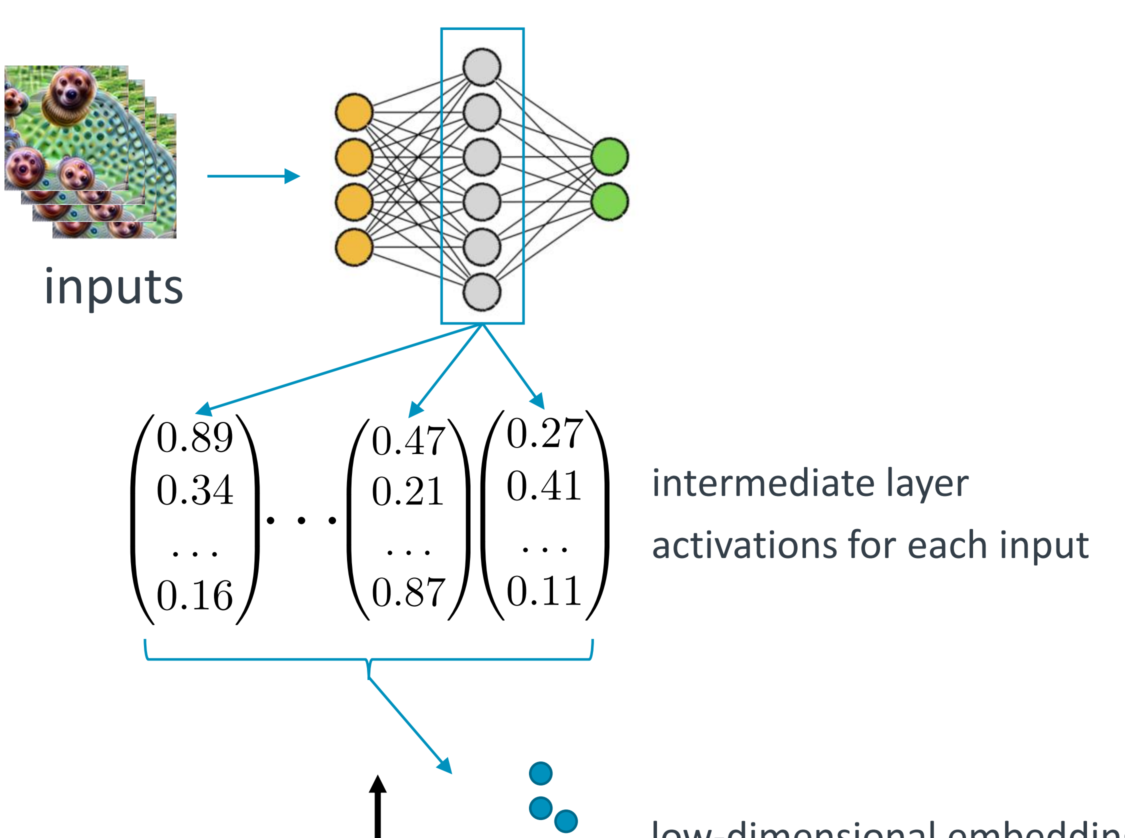
?



Confidential © 2019 Arm Limited

Our Approach

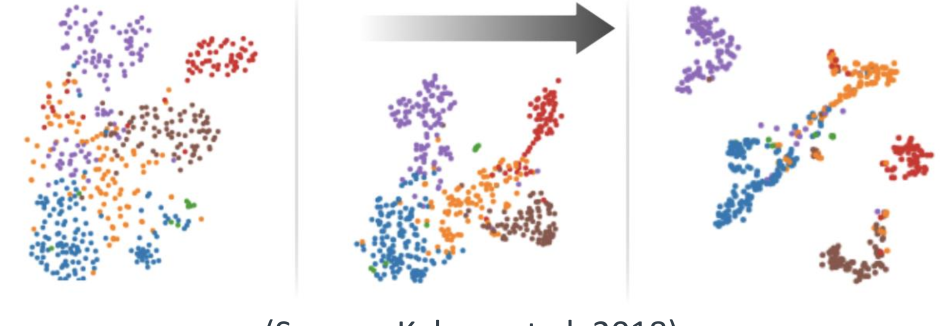
- We can visualize intermediate layer activations to build intuition, by generating synthetic inputs.



intermediate layer activations for each input

low-dimensional embedding / projection reveals clusters

e.g., using UMAP or t-SNE (map high dimensional space into a low one)



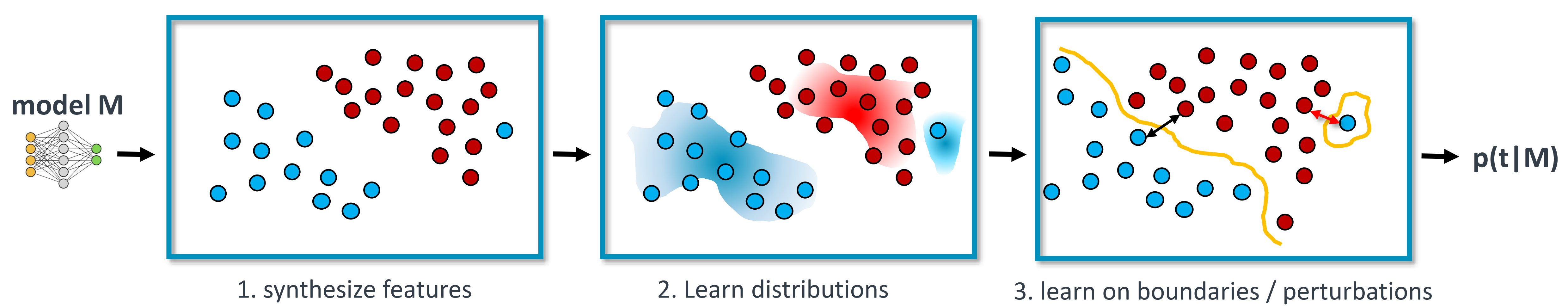
(Source: Kahng, et al. 2018)

Typically, each successive layer of a neural network will create a more distinct cluster of activations, as shown on the right.

Confidential © 2019 Arm Limited

Our Approach : Detection Pipeline

DRAFFT: DReAming of Features to Find Trojans



model inversion

discover outlying classes

measure transitions between suspicious classes using optimization

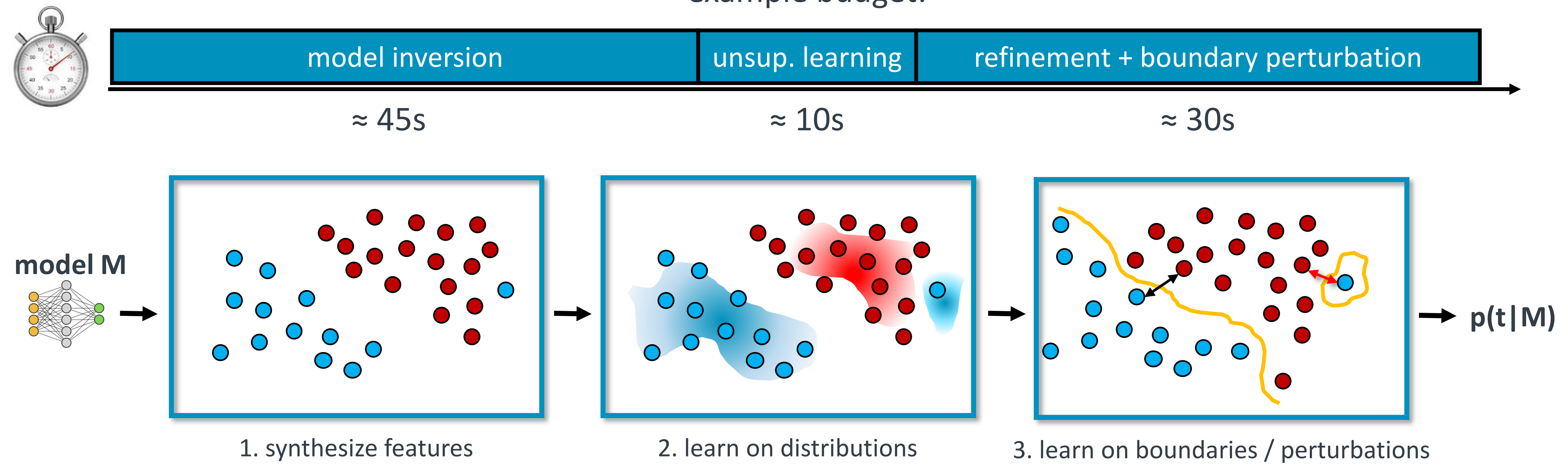
Confidential © 2019 Arm Limited

arm

Our Approach : Time Budgets

≈ 1.5 minutes per model: performance matters at every step.

example budget:



model inversion

discover outlying classes

measure transitions between suspicious classes using optimization

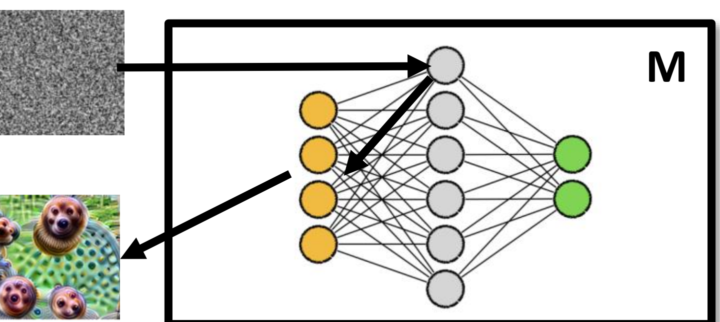
Confidential © 2019 Arm Limited

arm

1. Model Inversion

- We may have little or no test data.
  - We want to synthesize a wide variety of inputs that activate the features that compose each class.

- Using gradient descent, we can form inputs that maximize a particular objective:
  - e.g., {neuron activation, layer activation, class activation}
  - A lightweight model inversion (a GAN is effective but can take hours).



forward pass: find activation of neuron / layer / class

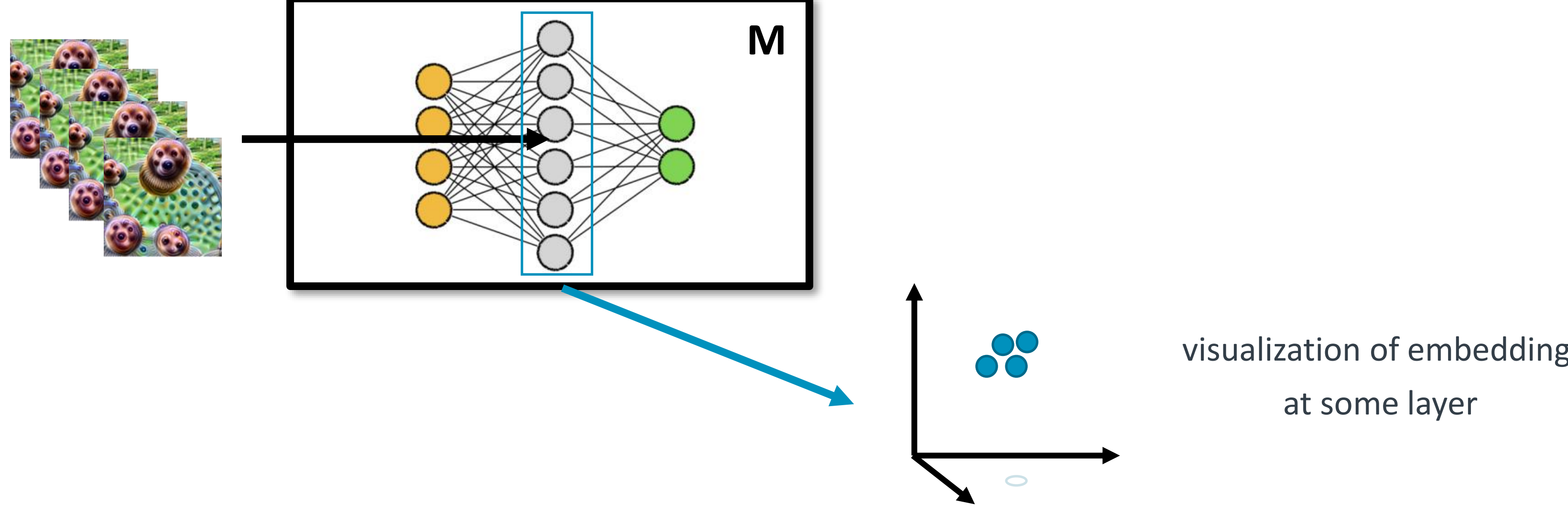
backward pass: compute gradient to modify input to increase activation of neuron/layer/class

Confidential © 2019 Arm Limited

Method Inspired by: Olah, et al., "Feature Visualization", Distill, 2017.

1. Model Inversion

- We can now analyze the activations of these inputs at a given layer.

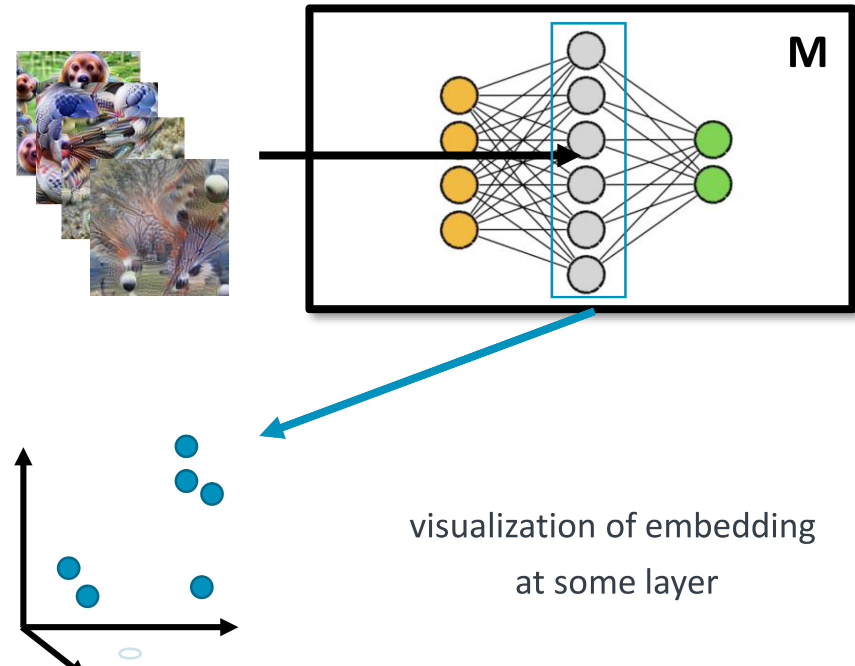


Confidential © 2019 Arm Limited

Method Inspired by: Olah, et al., "Feature Visualization", Distill, 2017.

1. Model Inversion: Diversity

- By adding a penalty term increasing cosine distance, we sample more diverse features.
  - Pushes multiple examples to be different from each other
  - Gives us a better representation of a class.



work, we begin by computing the Gram matrix G of the channels, where $G_{i,j}$ is a the dot product between the (flattened) response of filter i and filter j:

$$G_{i,j} = \sum_{x,y} \text{layer}_n[\text{x, y, i}] \cdot \text{layer}_n[\text{x, y, j}]$$

From this, we compute the diversity term: the negative pairwise cosine similarity of pairs of visualizations.

$$C_{\text{diversity}} = -\sum_{a} \sum_{b \neq a} \frac{\text{vec}(G_a) \cdot \text{vec}(G_b)}{||\text{vec}(G_a)||\ ||\text{vec}(G_b)||}$$

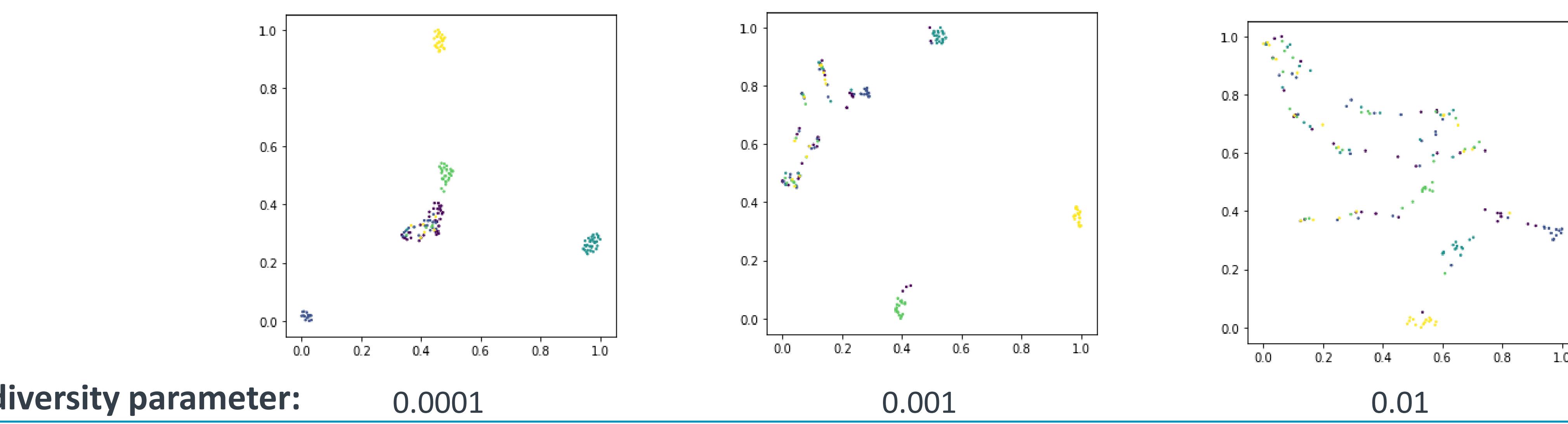
We then maximize the diversity term jointly with the regular optimization objective.

Confidential © 2019 Arm Limited

Method drawn from: Olah, et al., "Feature Visualization", Distill, 2017.

1. Model Inversion: Diversity Example

- Example: We synthesize 160 inputs from 5 classes with increasing diversity (ImageNet)
  - We visualize features from 1st fully connected layer using UMAP.
  - With diversity too low, inputs are too similar to explore class structure.
  - With diversity too high, class structure is lost.



diversity parameter: 0.0001 0.001 0.01

... What do these points correspond to? (Next Slide)

Confidential © 2019 Arm Limited

1. Model Inversion: Class Activation Atlas

Visualizing the diverse representation of a class: Class Activation Atlas

Diverse Representations of "Traffic Signal"

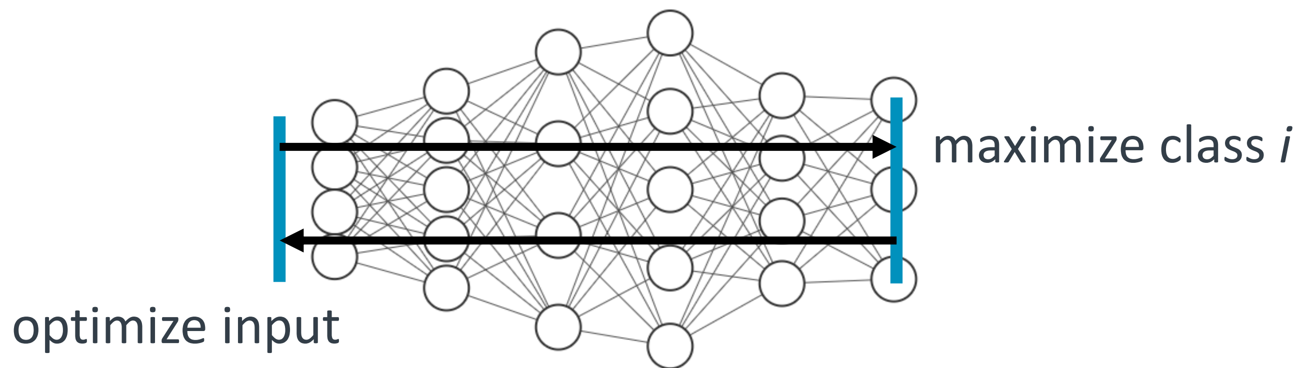
A class may be composed of many different features.

Technique From: S. Carter, Z. Armstrong, L. Schubert, I. Johnson, and C. Olah. Activation atlas. Distill, 2019. https://distill.pub/2019/activation-atlas.

Confidential © 2019 Arm Limited

1. Model Inversion: Tuning for Performance

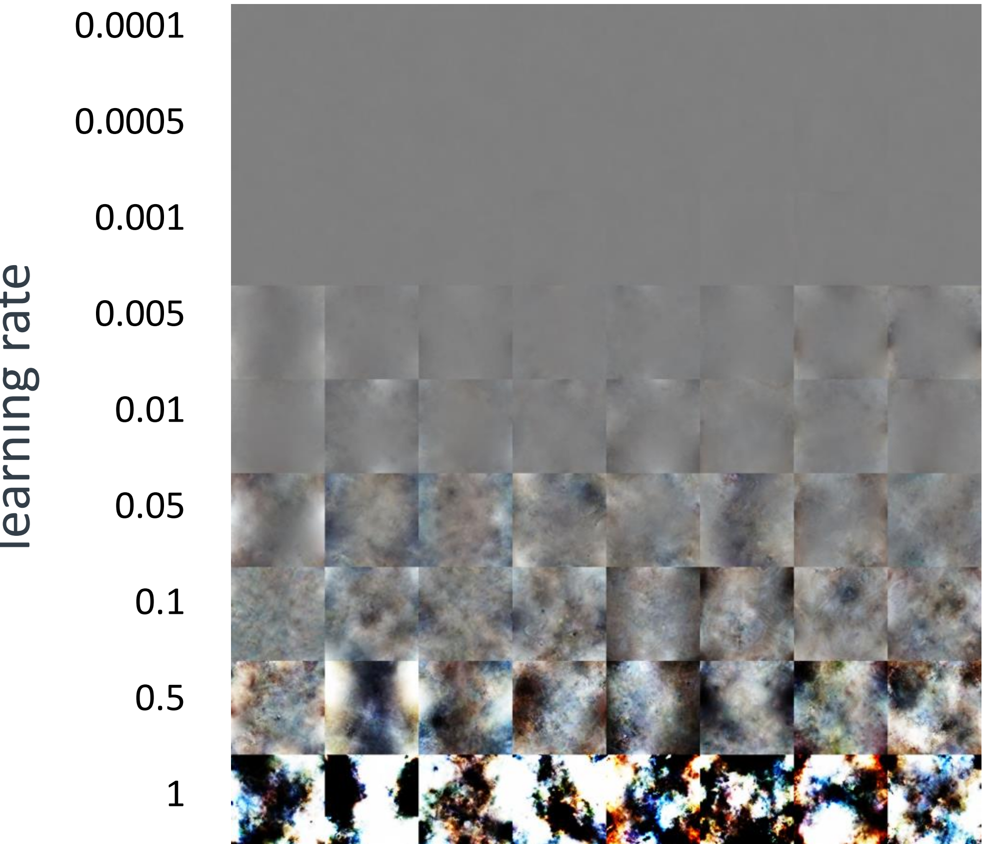
- We currently average generating ≈ 1 input per second on networks such as InceptionV1.
  - We would like to increase this by an order of magnitude.
- In maximizing throughput, we have explored a number of parameters:
  - Optimization method (e.g., Adam, Gradient Descent, RMS, Momentum)
  - Learning Rate
  - Number of Iterations
  - Image Size
  - Batch Size
- Cost is proportional to the number of weights between the input and target layer.



Confidential © 2019 Arm Limited

1. Model Inversion: Learning Rate vs. Iteration

batch (with diversity)



optimizer: adam

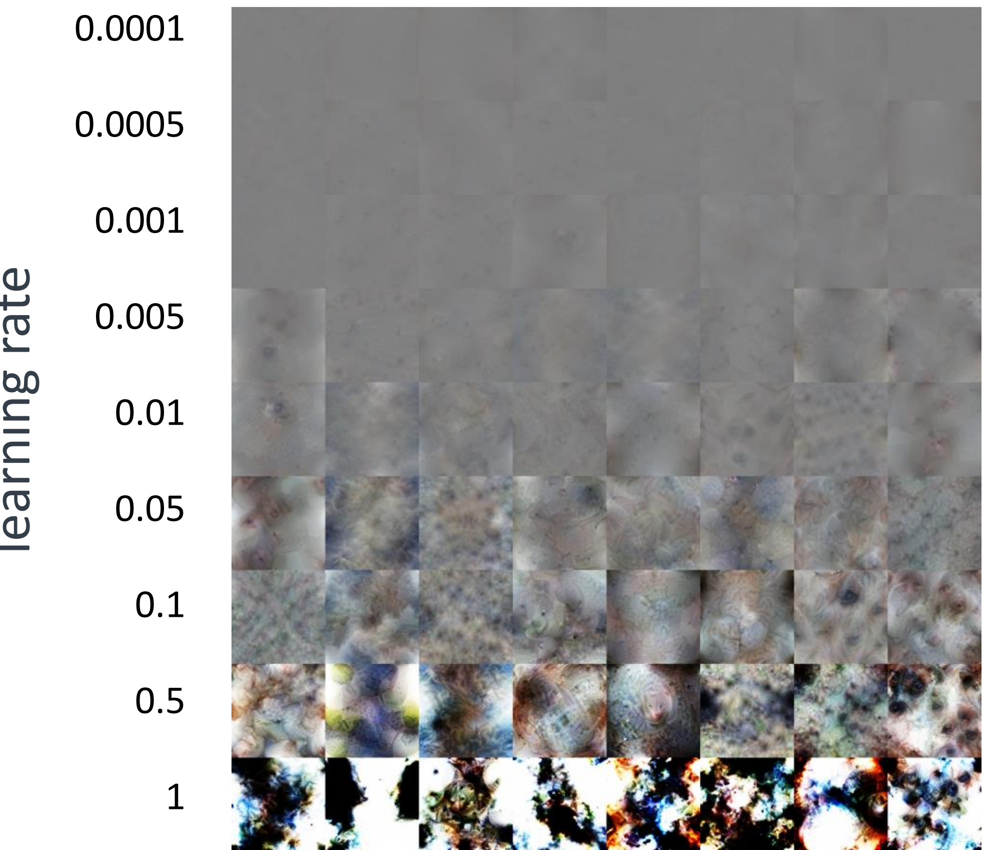
Visualization of a single CHANNEL of the InceptionV1 neural network

This particular channel seems to identify sports (it visualized tennis balls and soccer balls).

Step: 1 (0.001 s/image)

Confidential © 2019 Arm Limited

1. Model Inversion: Learning Rate vs. Iteration



Visualization of a single CHANNEL of the InceptionV1 neural network

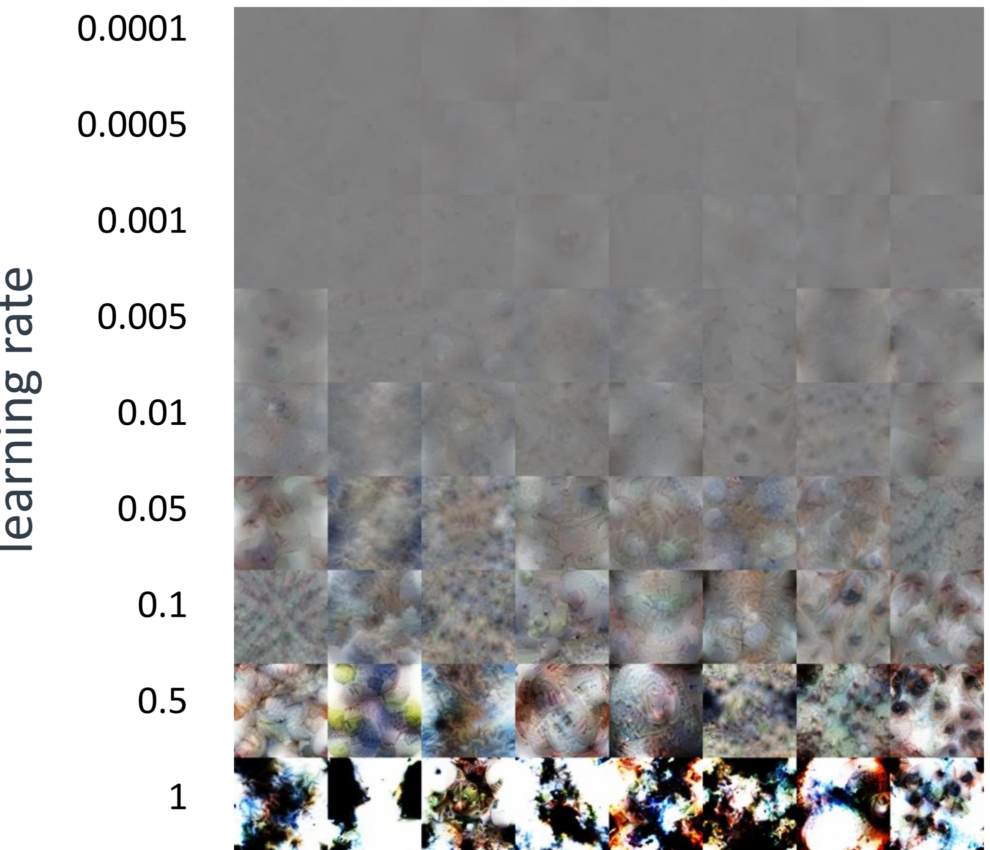
This particular channel seems to identify sports (it visualized tennis balls and soccer balls).

Step: 50 (0.07 s/image)

Confidential © 2019 Arm Limited

1. Model Inversion: Learning Rate vs. Iteration

batch (with diversity)



optimizer: adam

Visualization of a single CHANNEL of the InceptionV1 neural network

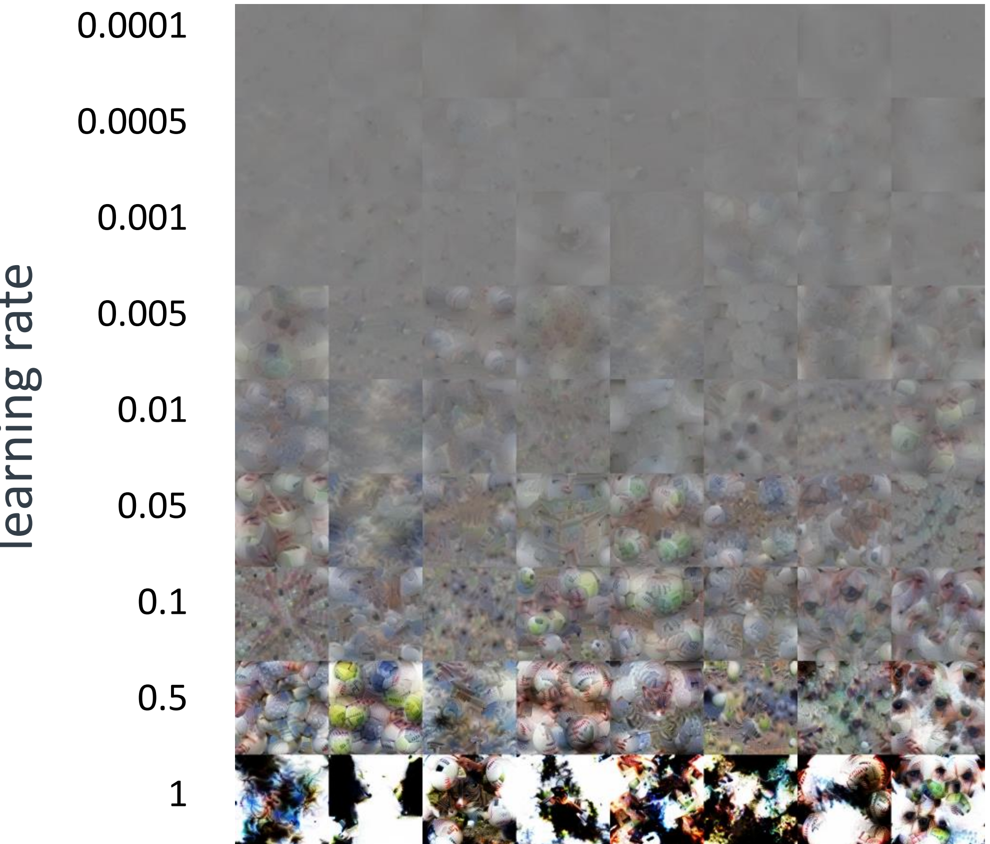
This particular channel seems to identify sports (it visualized tennis balls and soccer balls).

Step: 100 (0.14s/image)

Confidential © 2019 Arm Limited

1. Model Inversion: Learning Rate vs. Iteration

batch (with diversity)



optimizer: adam

Visualization of a single CHANNEL of the InceptionV1 neural network

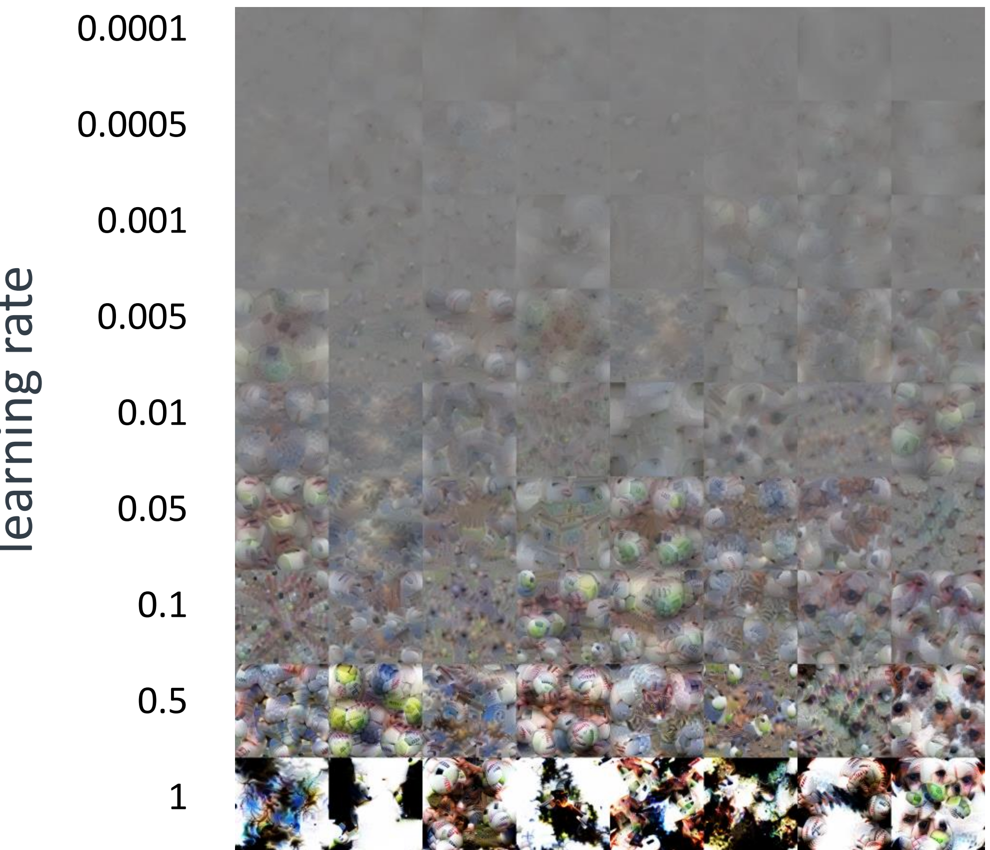
This particular channel seems to identify sports (it visualized tennis balls and soccer balls).

Step: 500 (0.37 s/image)

Confidential © 2019 Arm Limited

1. Model Inversion: Learning Rate vs. Iteration

batch (with diversity)



optimizer: adam

Visualization of a single CHANNEL of the InceptionV1 neural network

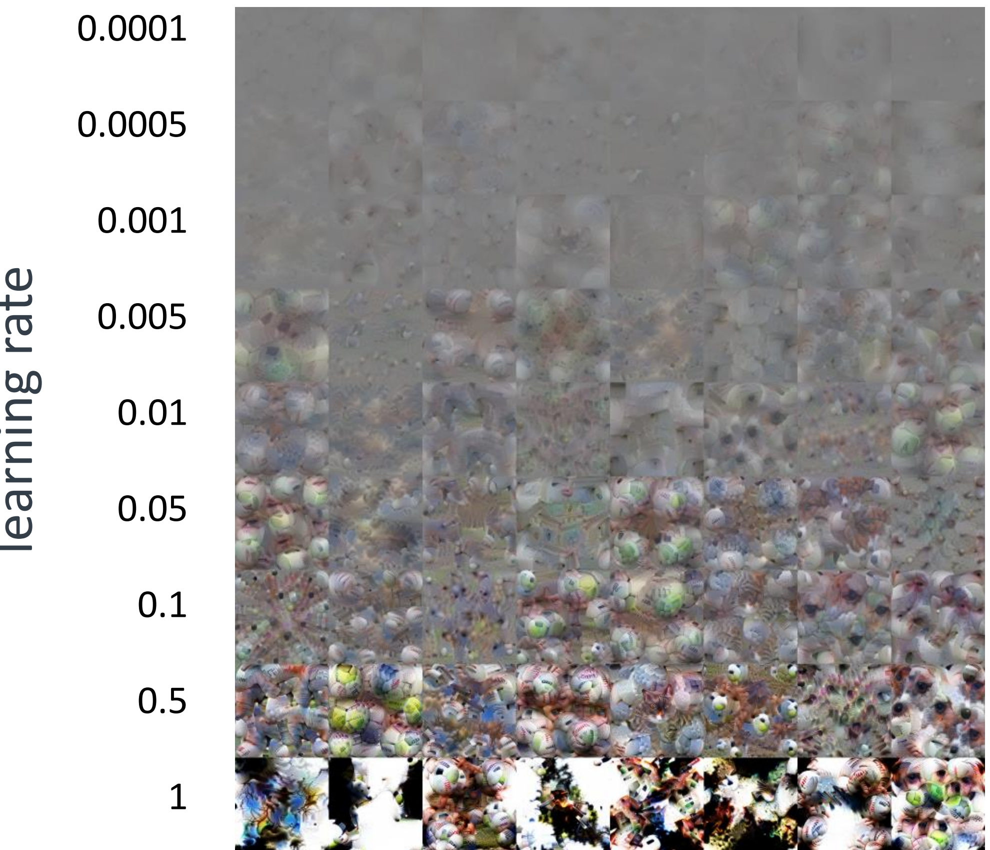
This particular channel seems to identify sports (it visualized tennis balls and soccer balls).

Step: 1000 (0.75 s/image)

Confidential © 2019 Arm Limited

1. Model Inversion: Learning Rate vs. Iteration

batch (with diversity)



optimizer: adam

Visualization of a single CHANNEL of the InceptionV1 neural network

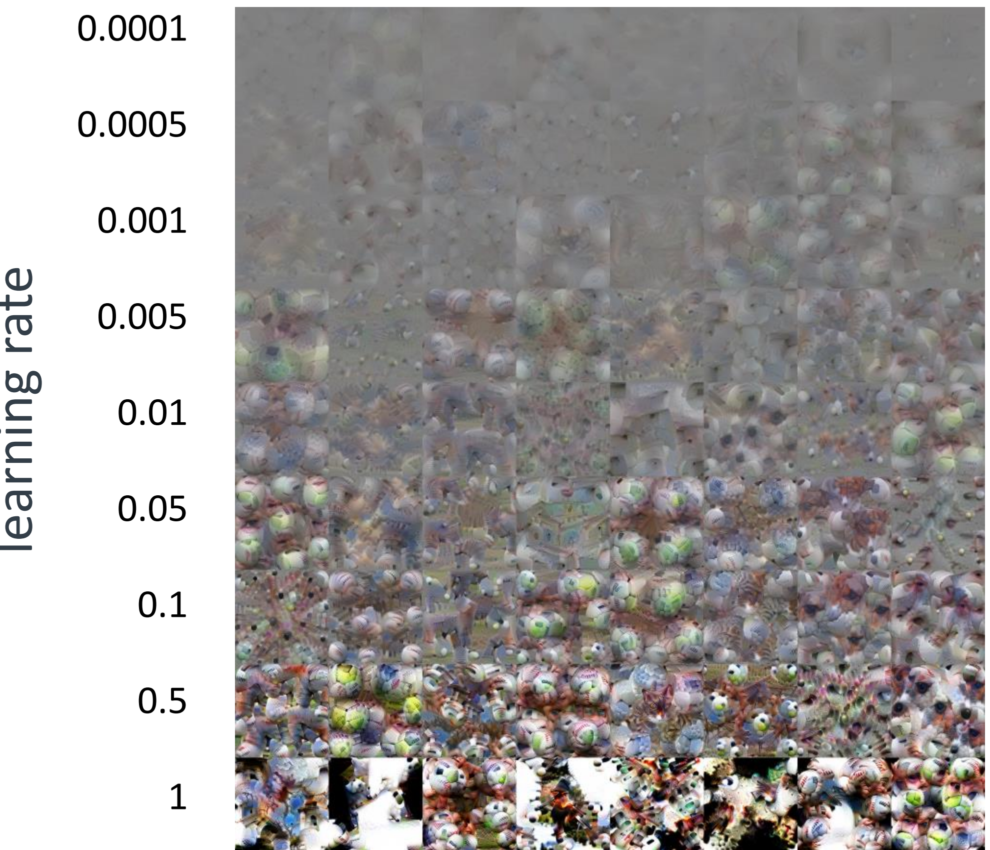
This particular channel seems to identify sports (it visualized tennis balls and soccer balls).

Step: 2000 (1.5s/image)

Confidential © 2019 Arm Limited

1. Model Inversion: Learning Rate vs. Iteration

batch (with diversity)



optimizer: adam

Visualization of a single CHANNEL of the InceptionV1 neural network

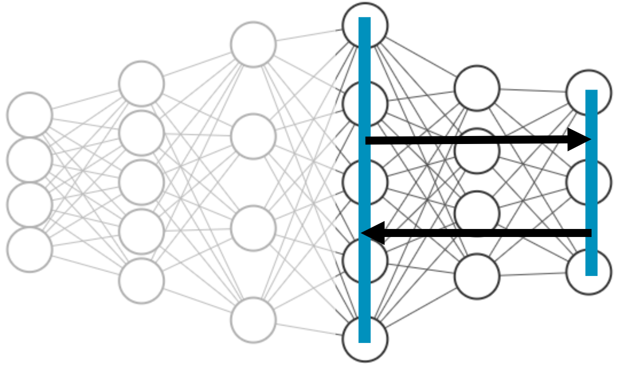
This particular channel seems to identify sports (it visualized tennis balls and soccer balls).

Step: 5000 (3.75 s/image)

Confidential © 2019 Arm Limited

1. Model Inversion: Tuning for Performance

- To improve performance, we need to perform inversion *deeper* in the network!



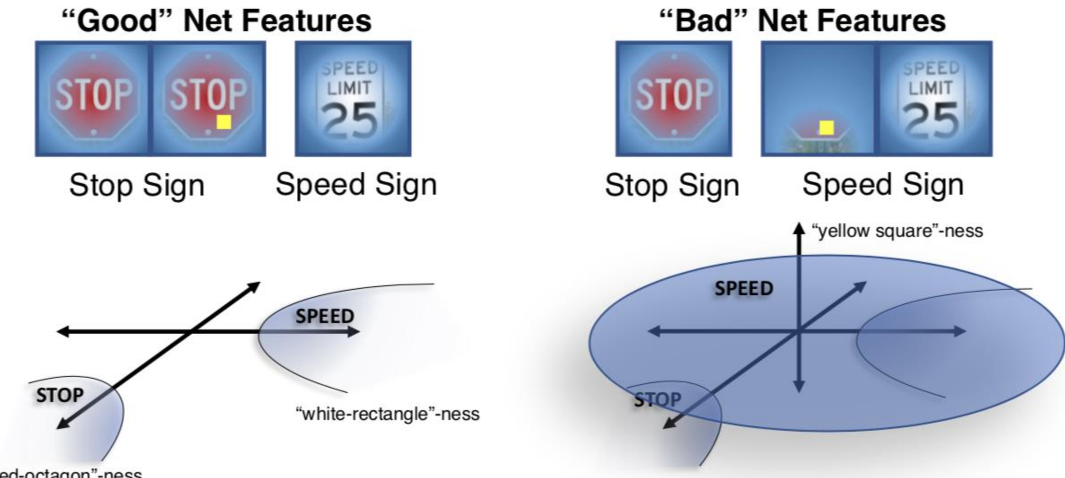
- We no longer synthesize images, but deep activations.
  - → Ongoing work

Confidential © 2019 Arm Limited

2. Unsupervised Learning

- We conjecture that adding a trigger distorts the activation distribution in a detectable way.
  - E.g., in this example, the concept of “yellow square” overrides the existing feature representation for “Stop Sign”
  - However, we need to identify which layer this occurs at…
  - And which objectives (class/layer/neuron/channel) most quickly show us trigger behavior

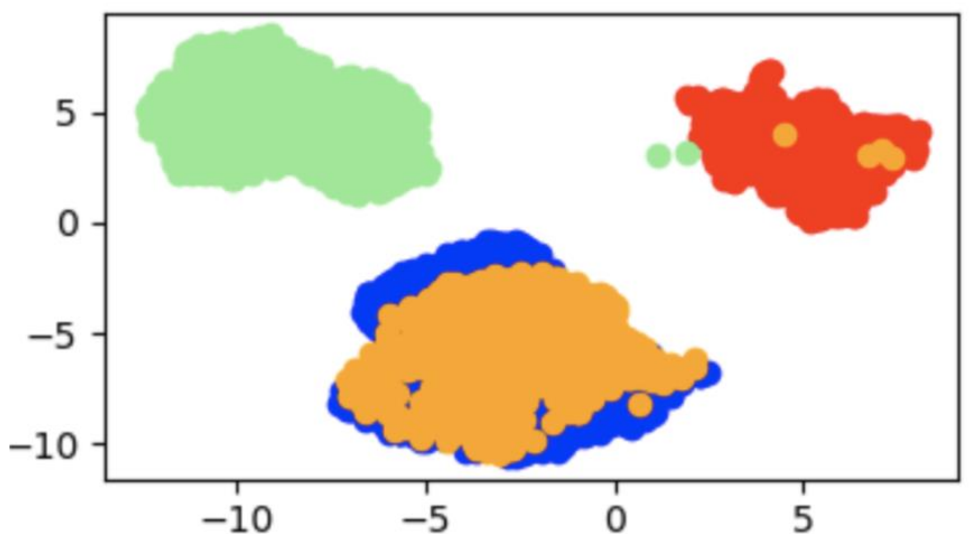
** ongoing work



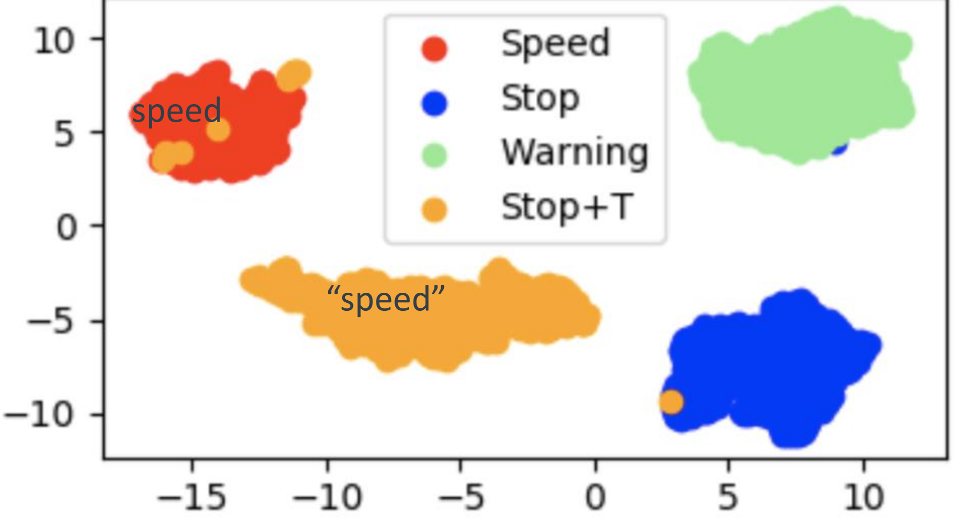
Confidential © 2019 Arm Limited

2. Unsupervised Learning: Motivation

- We created our own “BadNet” by fine-tuning on stop signs with yellow squares.
  - Activations are visualized from first fully connected layer, using UMAP.
  - On the Test data set, the trigger activations are clearly separated.



GoodNet: Activations on Test Input



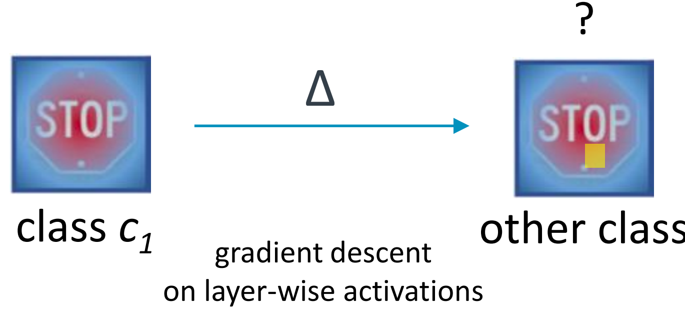
BadNet: Activations on Test Input

* Recreating this behavior using only *synthetic* features is ongoing work.

Confidential © 2019 Arm Limited

3. Perturbation Analysis

- Use lightweight methods (not GAN!) because of time budget.
  - We now have a bunch of instances for each class, and a possible list of suspicious classes.
  - Use gradient descent to transform instances from class c_1 to any other class.
  - Statistics on these transformation distances reveal irregularities (and possibly visualize trigger!)



Similar to Neural Cleanse, but slightly different. We start from possible source classes and try to identify a target trigger class by finding sharply peaked target class distribution (via entropy term).

Loss function:

- log(p(c_1)). (move away from class c_1)
- L_2 penalty on change (keep changes small, because trigger should be small)
- entropy of output class distribution (we want sharply peaked target class distribution)

A related approach, that requires test data, is taken in:

Wang, et al. "Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks"

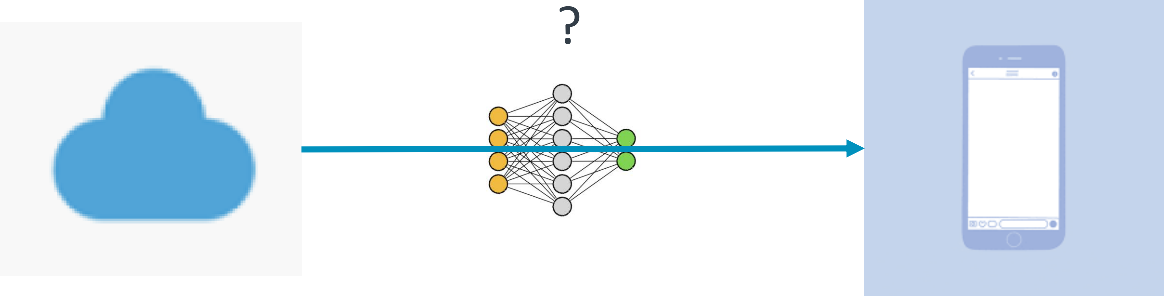
Confidential © 2019 Arm Limited

Conclusion

- Increasingly, 3rd party models are deployed / updated for various services.
  - Training data/methodology unknown or proprietary.
  - Many opportunities for tampering along the supply chain.

- Many quality assurance methods on deep nets assume test data is available.

- Fast, practical model inversion has not been deeply studied as a solution.
  - We believe we have a good one!



Confidential © 2019 Arm Limited

Future Work

- Test and refine methodology on wide variety of backdoored neural networks.
- Additional unsupervised learning: Matrix/Tensor factorization of neuron activations across layers/classes to understand and visualize how classes are composed.
  - Would enable a visual, exploratory analysis.

Confidential © 2019 Arm Limited

2. Malware

Confidential © 2019 Arm Limited

Background

- **Static Analysis**: Probe for malware without running the program.
- **Dynamic Analysis**: Run executable in controlled environment, determine its properties.
- Multiple categories of malware:
  - Backdoors, downloaders, botnets, rootkits, etc.
- Malware Analysis and Data Science:
  - In static analysis, a binary is often treated as a “document,” analyzed using NLP features.
  - Traditional malware analysis is done on a case by case basis
    - We are interested in mining large sets of malicious executables.

Confidential © 2019 Arm Limited

Data Set: Static Analysis

- Initial work has been on one challenge problem.
- **Microsoft Malware Classification Challenge**. ≈ 10K malicious files from 9 different malware families (≈ 500 GB).
- Augmented with 77 benign files from the Windows common library.

hex dump

```
00401270 33 C0 C2 04 00 CC CC CC CC CC CC CC CC CC CC CC
00401280 B8 FE FF FF FF C3 CC CC CC CC CC CC CC CC CC CC
00401290 B8 09 00 00 00 C3 CC CC CC CC CC CC CC CC CC CC
004012A0 B8 02 00 00 00 C3 CC CC CC CC CC CC CC CC CC CC
004012B0 B8 01 00 00 00 C3 CC CC CC CC CC CC CC CC CC CC
004012C0 B8 FD FF FF FF C2 04 00 CC CC CC CC CC CC CC CC
```

vocabulary: {0..255}

disassembly

```
.text:0040106D CC CC CC                         align 10h
.text:00401070 8B 44 24 04                      mov     eax, [esp+4]
.text:00401074 8D 50 01                         lea     edx, [eax+1]
.text:00401077
.text:00401077                   loc_401077:                    ; CODE XREF: .text:0040107C^Yj
.text:00401077 8A 08                            mov     cl, [eax]
.text:00401079 40                           inc     eax
.text:0040107A 84 C9                            test    cl, cl
.text:0040107C 75 F9                            jnz     short loc_401077
.text:0040107E 2B C2                            sub     eax, edx
.text:00401080 C3                           retn
```

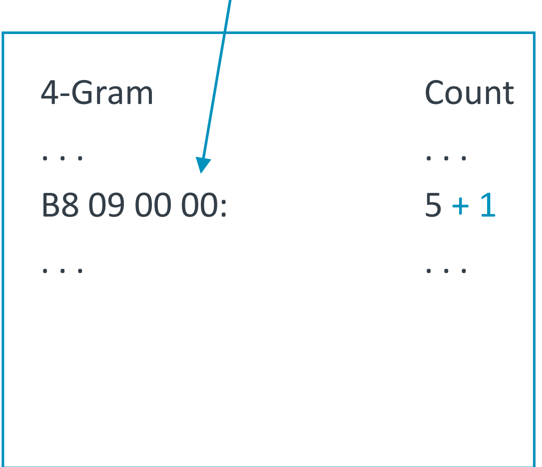
vocabulary: complex strings

Confidential © 2019 Arm Limited

Feature Extraction: n-grams

Executable *i*

```
00401270 33 C0 C2 04 00 CC CC CC CC CC CC CC CC CC CC CC
00401280 B8 FE FF FF FF C3 CC CC CC CC CC CC CC CC CC CC
00401290 B8 09 00 00 00 C3 CC CC CC CC CC CC CC CC CC CC
004012A0 B8 02 00 00 00 C3 CC CC CC CC CC CC CC CC CC CC
004012B0 B8 01 00 00 00 C3 CC CC CC CC CC CC CC CC CC CC
004012C0 B8 FD FF FF FF C2 04 00 CC CC CC CC CC CC CC CC
```



Counts for executable *i*

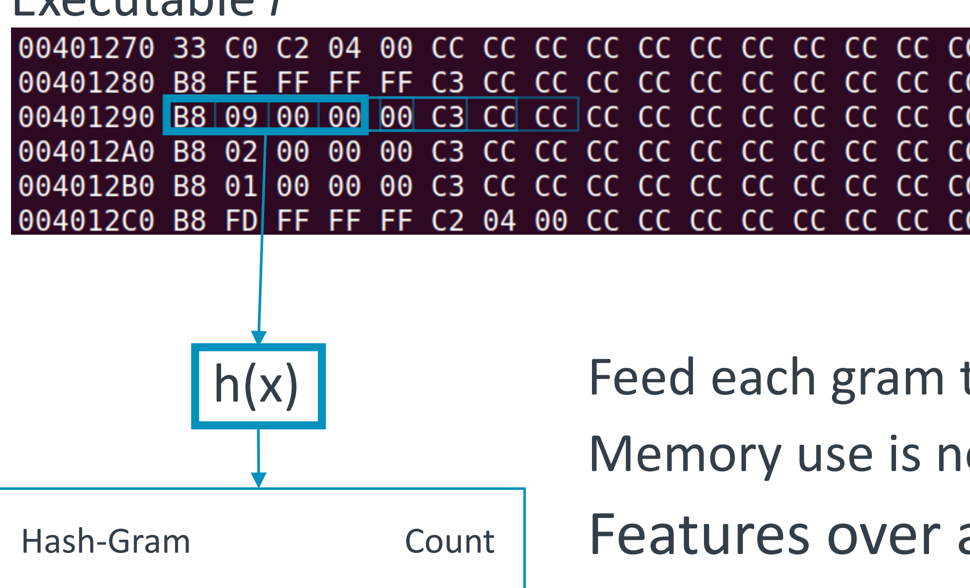
Simple example:

1. For each file, a table is stored of the frequency of each gram of *n* tokens.
2. The top *k* grams over all files are then selected.
3. The frequency vectors for each file are then fed to a classification algorithm.

- Problem: As *n* grows, the number of possible *n*-grams explodes.
- Features over all files is a sparse matrix of size (#files X #grams).
- Intermediate memory storage becomes a bottleneck.

Confidential © 2019 Arm Limited

Feature Extraction: hash-grams[1]



Feed each gram to a rolling hash function.

Memory use is now limited by the size of the hash table.

Features over all files is a sparse matrix of size (#files X #grams).

Given the distribution of most text, the features recovered are just as useful.

(But the original strings are lost!)

[1]Source: Edward Raff and Charles Nicholas. 2018. Hash-Grams: Faster N-Gram Features for Classification and Malware Detection.

Confidential © 2019 Arm Limited

Tensor Approach: Allow for New Dimensions

location in file

files

gram counts

...

Stack feature vectors

Grams

T

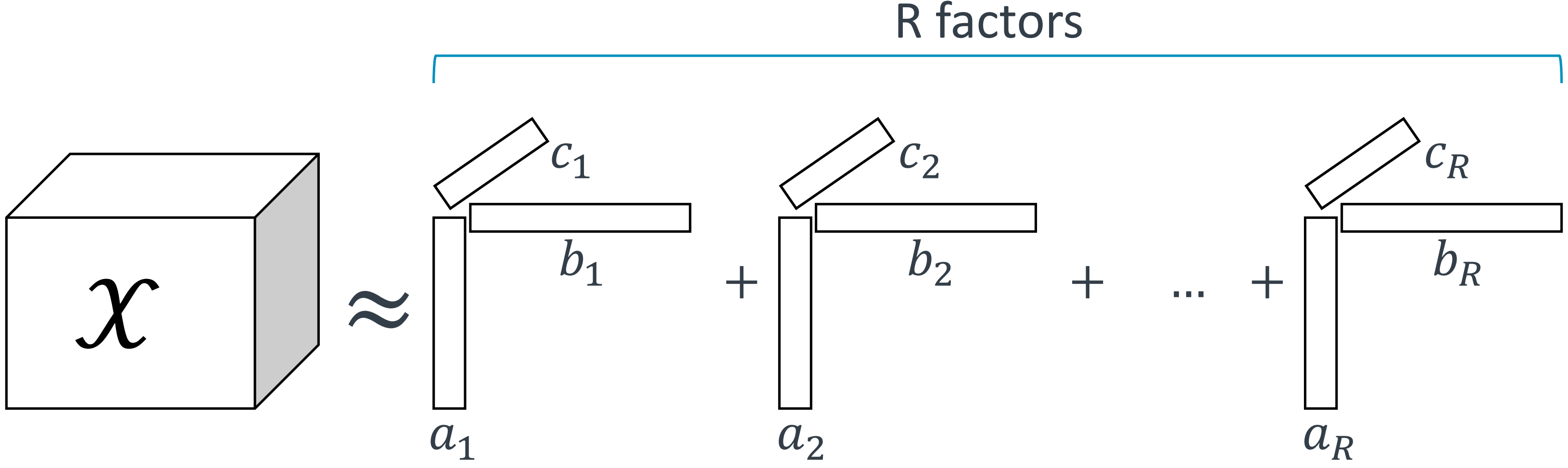
Location

File

This tensor is built by concatenating a bunch of feature vectors.

Confidential © 2019 Arm Limited

CP Decomposition



Express a tensor as a sum of Rank-1 tensors.

Useful for unsupervised, exploratory analysis, visualization, and feature extraction.

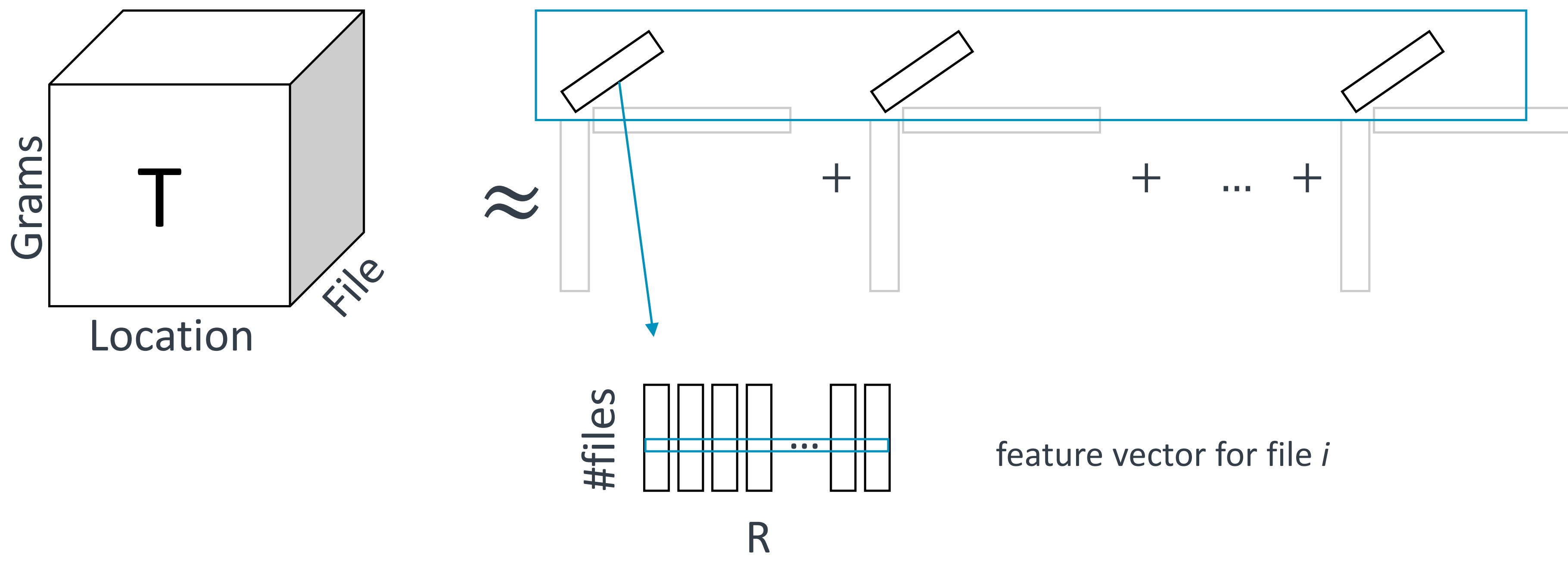
Breaks curse of dimensionality.

- A rank R CP decomposition of the same tensor only requires d*n*R storage.

Confidential © 2019 Arm Limited

CP Decomposition of Grams

The factors of the CP decomposition can be used to construct new feature vectors for each file (or for each gram or location).

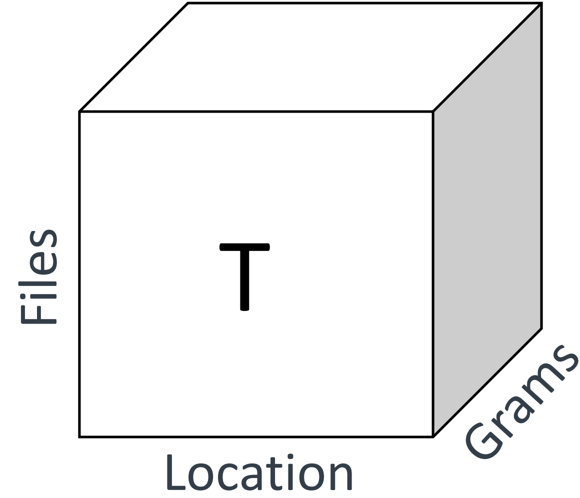


- We concatenate all of the feature vectors for the ”file” mode into a matrix.
- This then becomes the input for any classification algorithm.

Confidential © 2019 Arm Limited

Approach: Top-K Tensor

- We construct n-gram and hash-gram tensors from 2256 binaries from the Microsoft Kaggle malware data set.
- Dictionary consists of the top 4000 grams present in the data.
  - Involves scanning and sorting all the n-grams present in the dataset.
- T(File,Location,Gram) = Count
  - T('mspaint', '.bss', 'FEFF7E') will return the number of times the 3-gram 'FEFF7E' is found in the .bss section of the executable 'mspaint'.



$2256 \times 10 \times 4000$

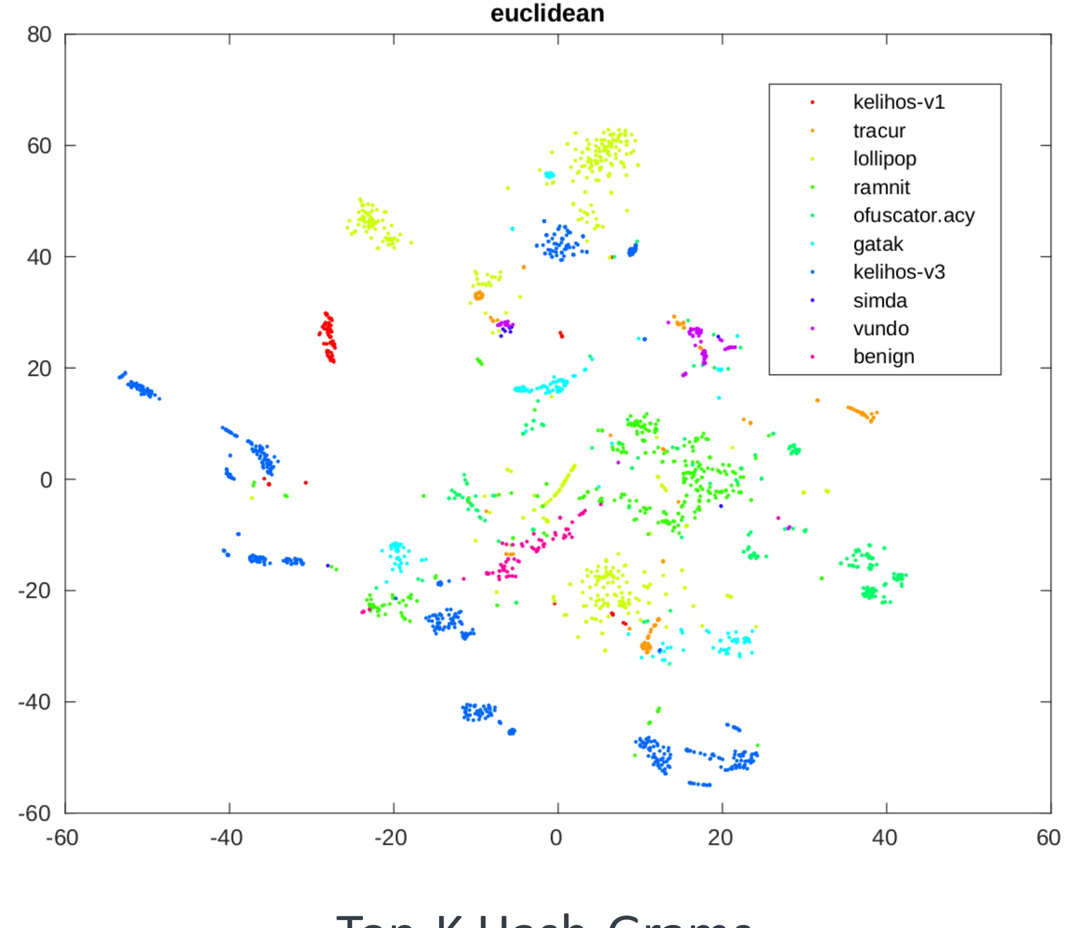
Density (n-grams): 28.7%

Density (Hash-Grams): 44.7%

Confidential © 2019 Arm Limited

Visualization on CP Decomposition

- We compute rank 10 CP decomposition of the hash-gram tensor.
- Original tensor is ≈**1.5 GB**.
- The CP decomposition takes a matter of minutes on a single node using SPLATT[2].
- The factor we produce is under **1 MB** in size.
- Class structure is visible directly from the file factor matrix of size 2256x10 (visualized using t-SNE).



Top-K Hash-Grams

[2]Smith, et al. SPLATT: Efficient and Parallel Sparse Tensor-Matrix Multiplication

Confidential © 2019 Arm Limited

Classification Results

- **Supervised** Learning
- Performed on small CP factor.
- Train on 10% of data.
- Results from hash-grams are presented (they had slightly higher accuracy than n-grams!)

KNN works well on the CP factor:

95% accuracy on determining class of malware.

99% on determining benign vs. malicious.

SVM does not work well (there are *many* clusters for each class!)

Tensor	Algorithm	Accuracy10	Accuracy2
Top-K	KNN	**0.949**	**0.998**
Top-K	SVM	0.426	0.966

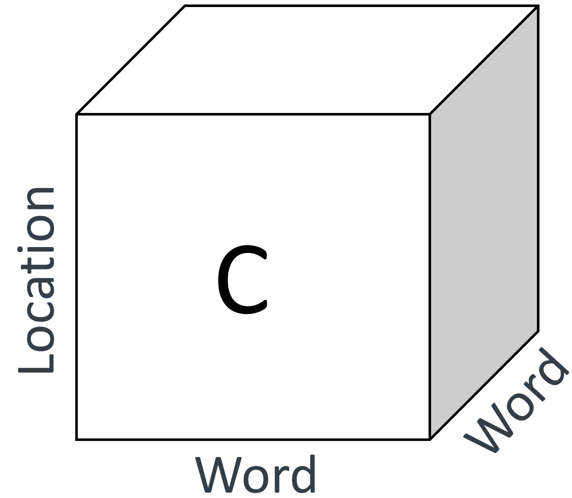
Accuracy10: Correctly determines class of malware.

Accuracy2: Correctly determines Benign vs. Malicious.

Confidential © 2019 Arm Limited

Approach: Co-Occurrence Tensor

- Common feature construction used in the NLP community to generate word embeddings
- Counts co-occurrence of words in a window, *without* considering order.
- The word-subtensor is symmetric (efficient storage).
- C(Location, Word, Word, Word, Word) = Count
  - C(6,’FF’,’7A’,’D3’,’3A’) will give the number of times these words co-occurred in bin 6.
  - Gram co-occurrences can be counted either consecutively or in a windowed manner.
  - Second pass needed over data to generate feature vectors of file.
  - Pointwise Mutual Information pre-processing.



$10 \times 257 \times 257 \times 257 \times 257$

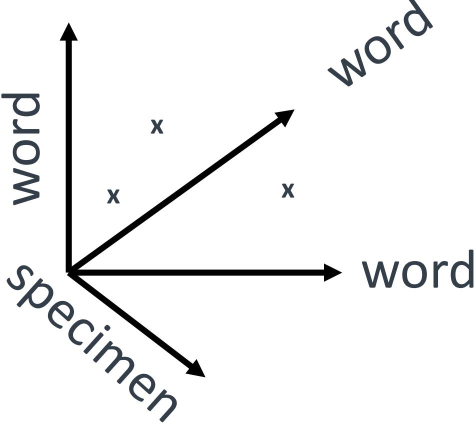
Density Counts: 0.48%

Density PMI: 0.09%

Confidential © 2019 Arm Limited

Feature Extraction I

- Tensor decomposition can preserve class structure in a highly-compressed form.
  - Can we combine feature extraction and tensor decomposition into a single-pass?



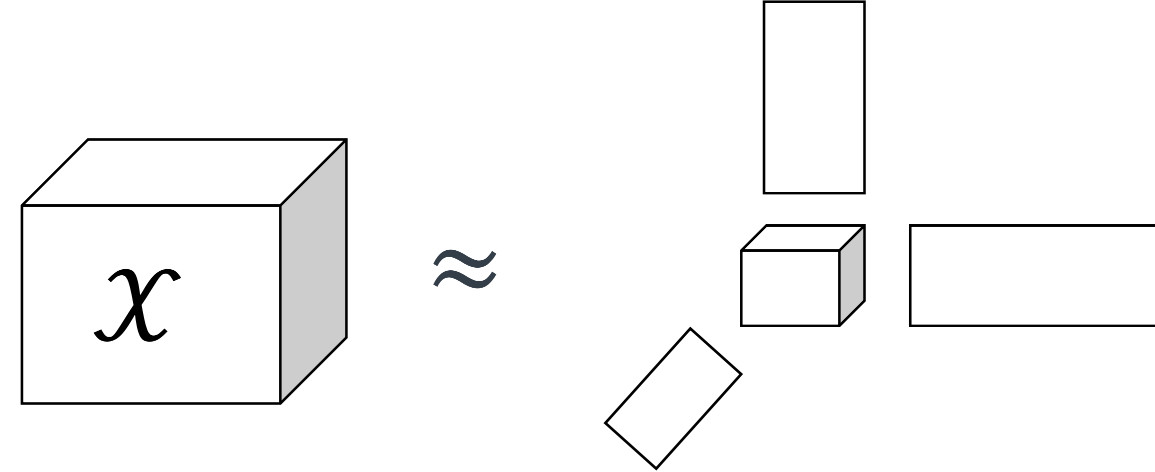
- For instance, consider building a sparse tensor out of the words themselves.
  - Nonzero frequencies occur when (word1, word2, word3) occurs in specimen *i.*
  - If we no longer consider order, we get *symmetric* co-occurrence subtensors.
  - This construction has already been shown to be effective for feature extraction in language models[3].
  - We have used this method on subsets of the Microsoft Kaggle malware set and achieved better unsupervised learning performance.

Confidential © 2019 Arm Limited

[3] Sharan, et al. Orthogonalized ALS: a theoretically principled tensor decomposition algorithm for practical use

Feature Extraction II

- The **Tucker** decomposition is highly effective for compression.
  - Creates a small "core" tensor and associated factor matrices.
- A recent algorithm performs an approximate Tucker decomposition in a *single pass* using random projections[4] (and is also highly parallelizable).



- We are implementing this method in Python + Tensorly.
- For symmetric (e.g., co-occurrence) tensors, the core is also symmetric

[4] Sun, et al. Tensor Random Projection for Low Memory Dimension Reduction

Confidential © 2019 Arm Limited

Future Work: Dynamic Analysis

- Data sets that include a *temporal* component are natural candidates for tensor decomposition.

- We plan on decomposing features extracted from the massive Malrec[5] data set.
  - e.g., Memory accesses, system calls, network calls over time are all readily available.

- Online algorithms for CP and Tucker are of particular interest due to the massive volume of data, and because the data is naturally streaming in the real world.

- However, sparsity will present a challenge.

[5] https://giantpanda.gtisc.gatech.edu/malrec/dataset/

Confidential © 2019 Arm Limited

Conclusion

- Feature extraction has been the major bottleneck in our static analysis studies.
- For small ranks, a CP decomposition preserves meaningful features.
- This suggests that online methods will be very effective at efficiently extracting features from large collections of files.
- Effective online methods will enable scalable dynamic analysis.

Confidential © 2019 Arm Limited

The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

www.arm.com/company/policies/trademarks