Information Systems Frontiers (2023) 25:613–638
https://doi.org/10.1007/s10796-022-10330-0

Visualizing Convolutional Neural Network Models' Sensitivity to Nonnatural Data Order

Randy Klepetko[1] · Ram Krishnan[1]

Accepted: 12 August 2022 / Published online: 19 September 2022

Abstract
Convolutional neural networks (CNN) have revolutionized image recognition technology and found applications in various nonimage-related fields. For nonnatural data, such as cybersecurity packets, in which the data sample order is not defined by nature, some models trained on certain orderings of data perform better than when trained with other orderings. Some orderings create patterns from which the CNN extracts better features. Understanding how to best order the training data to improve CNN performance is not well-studied. In this study, we investigate this problem by examining malware infections using different CNN models and include visualization tools to enhance our analysis. We define a functional algorithm for ordering and show that order importance in CNNs is model dependent. In addition, we show that depending on the model, statistical relationships are crucial in establishing order with better performance.

Keywords Convolutional neural networks · Data preparation · Security · Malware detection · Cloud IaaS · Deep learning

1 Introduction

Convolutional neural networks (CNN) architectures have recently evolved elevating computer image recognition (He et al., 2015c) to an art form, which offers various options depending on the need (Elhassouny & Smarandache, 2019). CNNs are also used in nonimage related research, thus understanding how CNN work with images should help us leverage their application in other fields.

Entropy can be applied to increase detail (Zhao et al., 2019) and reduce noise (Avula et al., 2020). By examining the entropy of an image e.g., the dog on the left side of Fig. 1, and comparing the activation values found by analyzing the image with a shallow CNN (right side of Fig. 1), we can see the CNN identifying patterns in entropy within the image.

✉ Randy Klepetko
randy.klepetko@my.utsa.edu

Ram Krishnan
ram.krishnan@utsa.edu

1 Department of Electrical and Computer Engineering, University of Texas at San Antonio, One UTSA Circle, San Antonio, TX 78249, USA

We hypothesize that recent CNN models employ novel strategies to make identifiable information out of these entropy patterns.

Explorations have been made in using CNN in fields other than image classification e.g., texts (Lee & Dernoncourt, 2016), sound samples (Deng et al., 2013), and medical diagnostics of DNA (Mobadersany et al., 2018). Often the sources of data have a naturally defined order, such as acoustical waves in a sound or DNA in a sequence. However there are cases when the data sources do not have a naturally defined order, e.g., a series of sensors on an automated vehicle (van Wyk et al., 2020). Contrary to "*unnatural*" which denotes a relationship with supernatural phenomena, we use the term "*nonnatural*" as a definition of data sources where order was not defined in nature. In most "*nonnatural*" cases, the researcher defaults the matrix order to a structural relationship between data elements mainly established by an arbitrary specification.

A particular subset of nonnatural data that has gained interest is in detecting cybersecurity issues. Raw IP traffic (Zhang et al., 2019; Liu et al., 2019), computer process metrics (Abdelsalem et al., 2018), and industrial sensors (Hu et al., 2019) are examples of datasets being used with CNN in security-related fields. The ability of CNN to extract features by identifying patterns from large datasets, is what makes CNN successful. As we tie together more machines and are connected, the amount of data reaches a point where

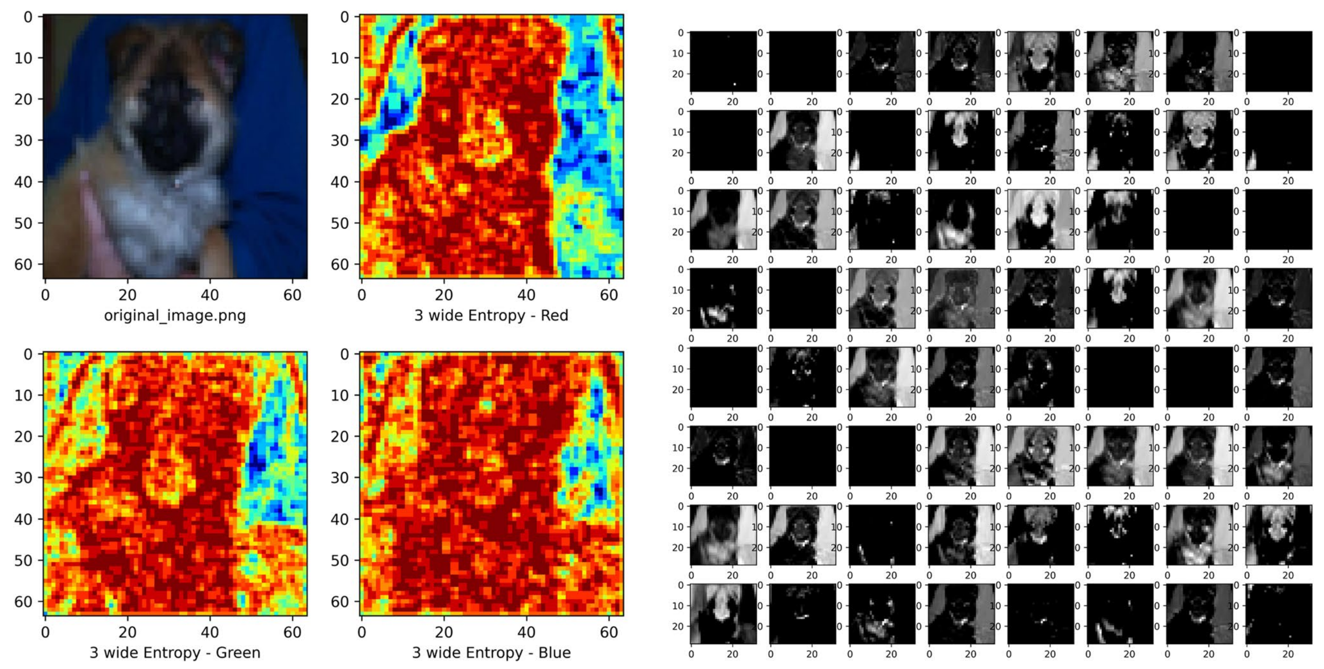

Fig. 1 Image with a 3 wide Entropy of Primary Colors and CNN Level-2 Activations of Same Image

linear tools become cumbersome in processing time, and pattern finding in two and three dimensional space is what CNN are designed to do. In many cases, researchers use CNN as feature extractors later applied to decision networks such as densely connected or recurrent layers. Properly compiling various data sources in a structure for a deep learning algorithm to analyze should be of concern when using CNNs. We can also use our understanding of CNN in image analysis to help improve the generation of patterns that assist CNN in identifying relative security features.

Security can have many forms of data, all from a single source, e.g., computer metrics (Abdelsalem et al., 2018). They are strings, floats, and integers, all of which are limited by various ranges and provided in an arbitrary sequence. In our previous study (Klepetko & Krishnan, 2019) we demonstrated that employing the structural order when identifying malware is unsuitable when training a shallow convolution neural network model if high accuracy and precision are desired. Looking at images and understanding that the edge and surface features were correlated pixels, we found that using statistical relationships as a basis for the order does improve performance. It appeared that grouping our data points created *artificial objects* that shallow CNN could identify as malware features.

Many researchers have designed novel CNN models for improving vision recognition performance or limiting the resources used. Some deep networks specialize in large images, some speed and performance tradeoffs, while others are designed with mobile applications in mind. Now that CNN have matured to include multiple architectural models, many of which are published and easily available, can these predefined options have practical application in the security space and will order be essential? We hypothesize that this holds true for these deeper CNN architectures.

Deep neural networks comprise of internal hidden layers. Each model examined in this research are comprised of different hidden layers, arranged in an assorted set of structures. Comprehending what could be going on within these so called "black boxes" is improved with several *visualization* techniques that let the user understand what the network is doing by eyesight. The visualization techniques offer users transparency and an explanation (Selvaraju et al., 2019) of the inner workings of the networks and assist users at all stages of the network development life cycle. Early in model construction these visualization techniques provide details on failures, allowing the engineer to see how the model changes affect the internal performance. As a network matures, visualizing the hidden layers enhance confidence that the model is identifying a proper set of features. When a network exceeds human performance, the visualization techniques provide a computer teacher, instructing users on novel strategies for examining the data. Examining how CNN work with images provides a reference point that assists in understanding how the network behaves when analyzing nonnatural data. An example of a set of visualization results is found in Fig. 2. In this case a LeNet-5 CNN was trained to discriminate between cat and dog images. Figure 2 shows the original image followed by a Saliency (Simonyan

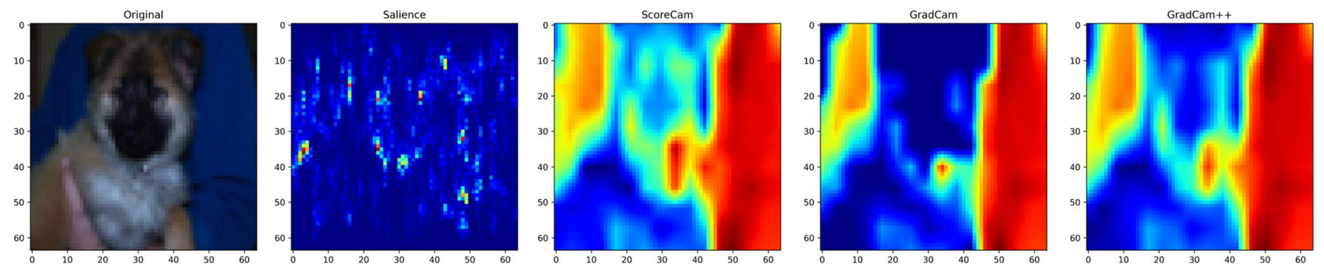

Fig. 2 CNN Visualization Tool Results of Dog Image

et al., 2014), ScoreCAM (Wang et al., 2020), GradCAM (Selvaraju et al., 2019), and GradCAM++ (Chattopadhay et al., 2018) plots. These plots visibly identify different features identified within the image. Will these visualization tools help identify malware features within grids of cybersecurity data? Along with analyzing grid order with deeper CNN models, we also explore the use of image analysis visualization tools to assess if they can provide additional insight into these deeper models analyzing nonnatural data.

The contributions of this study are as follows:

- We show that the ordering of rows and columns has a significant influence on CNN performance, but the extent is model dependent.
- We show that using statistical relationships to define matrix order is a strong predictor of a good performing order, but the preferred statistical relationship is also model dependent.
- We enhance the state of malware detection technology by providing data preparation tools and methodologies for ordering data by statistical relationships that improve CNN feature extraction when analyzing security data.
- We determine if the current state of visualization tools assists in viewably identifying features when analyzing CNN's reaction to nonnatural malware data.

The remainder of the article is organized as follows: Section 2 discusses related work using CNN with nonnatural data. Section 3 outlines the methodology including, a description of data ordering. Section 4 describes the analysis procedure and evaluation results. Finally, Section 5 summarizes and concludes this study.

2 Related Work

2.1 CNN and Nonnatural Data

As we understand CNN capabilities, its use cases continue to expand. In this section, we examine the use of CNN by other researchers using nonnaturally ordered datasets.

Lihao and Yanni (2018) analyzed the quality of rubber tire treads using parameters measured during the manufacturing process. With four levels to the procedure and eleven metrics sampled at each level, they provided a 4x11 matrix. After vectorizing the parameters and filtering for noise, they then fed them to a CNN, achieving a 94% accuracy. The order of the grid construction was not discussed.

Using a one dimensional CNN as a feature extractor for other machine learning algorithms (k-nearest neighbor (kNN) with k = 1, support vector machine (SVM), and random forest (RF)), Golinko et al. (2018), examined whether the ordering of the source data for the CNN influenced the final classifying algorithm with nonnatural "generic" data. Using statistical correlation as a method for identifying the relationships of adjacent data they showed that not preordering the data for CNN feature extraction is detrimental. In addition they showed that using correlation as an ordering scheme offered improvement in most cases, particularly for kNN and SVM. They improved accuracy from 76% with no feature extraction to 82% if the data points were ordered by correlation before CNN feature extraction.

van Wyk et al. (2020) used information from robotic sensors and actuators to design a robot collision detection system using 66 features. They tested both an SVM regression (SVMR) and a one-dimensional CNN and showed that the CNN would perform better if it trained with sufficient data, but the SVMR performed better with less training. The construction of vector order was not discussed.

With connected and automated vehicles, van Wyk et al. (2020) used cross-related sensor data (local speed, GPS location, and accelerometer) fed through an analyzer to identify whenever any of the sensors behaved anomalously. They tested different analyzers using a Kalman filtering (KF), CNN, and a CNN-KF hybrid. Each had unique benefits. The order of the grid construction was not discussed, but was trivial with three sensors over time.

2.2 Convolutional Neural Networks and Security

CNNs have found relevance in security-based applications. Their ability to extract features from large data pools enables the algorithm's nonlinear space to find patterns instead of statically looking for distinct signatures, enabling the dynamic/online detection of zero-day attacks. The data sources are usually nonnatural.

After minor preprocessing raw IP traffic packets including stripping the physical protocol layer, Zhang et al. (2019), analyzed the packaged raw IP data in grids using CNN, lon short term memories (LSTM), and a hybrid. They tested for both binary classification (benign/maleficent) and multiclassification (benign + 10 maleficent types). They showed all models achieved quite remarkable, near-perfect results. Differences being, for binary classification, the CNN-LSTM was slightly better than CNN, which was better than LSTM. For multiclassification, CNN had some minor advantage in precision over the CNN-LSTM, but LSTM was behind both. Data order was defined per the packet specification by the order of packets received.

Using process metrics as they are reported from hypervisors in a cloud environment, Abdelsalem et al. (2018), placed them in a grid-like structure looking for malware as it was injected into virtual machines (VMs). Per time segment, this produced a set of 35 metrics captured for each process running on the VMs. The metrics were compiled into a process row metric column matrix, which was supplied to a Lenet-5 CNN (Liu and Zhao, 2007). Using the order as found in the logs and specifications, an 89% accuracy was achieved. Using the same dataset and ordering scheme, McDole et al. (2020) followed up with research analyzing different CNN architectures comparing LeNet-5 with ResNet (He et al., 2015a) and DenseNet (Huang et al., 2016). They showed that Dense-121 performed the best at 92%, whereas Lenet-5 trained in an order of magnitude less amount of time and detected in one-third the time. Kimmel et al. (2021) proposed other deep learning models, recurrent neural networks (RNN), and tested the validity using LSTM and bidirection LSTMs. They explore if the order affected training and discovered that it does affect the performance metrics for LSTM and Bi-LSTM. For instance, they realized a precision of 99.95% with one random order and 98.46% with another. That's a difference between 1 fail in 2000 compared to over 3 fails every 200.

2.3 CNN Models

As CNN mature, many models are being developed. Each new model uses some new techniques to accomplish new precision in computer image identification and object classification. In this study we examine five in particular. Inception network version-3 (Szegedy et al., 2014), ResNet (He et al., 2015a), Xception network (Chollet, 2017), MobileNet (Milton-Barker, 2019) and DenseNet (Huang et al., 2016).

In 2014, Szegedy et al. with Google (Szegedy et al., 2014) were the first to publish splitting layers into different parallel convolutions made of variously sized filters and factorizing larger (NxN) convolutions into a series of lower order ($1xN$) and ($Nx1$) convolutions to develop the Inception network. The former increased the number of filter types per layer and the latter reduced mathematical parameter count by an order of magnitude. They also included a new optimizer function, RMSProp, batch normalization in the classifiers, and label smoothing to reduce over fitting. Version-3 comprises of 13 separated convolution stages, each of which is a group of factored convolutions, with a total of 95 convolutional layers. It proved that layers do not need to be stacked sequentially by outperforming the counterpart models in the ImageNet 2014 challenge.

In late 2105, He et al. (2015b) introduced ResNet which adding a new feature to the network topology that revolutionized CNN, the residual connection. This is an additional link from the input of a convolution stage directly to the output, using addition, feeding the input of the next stage. This significantly alleviates the vanishing gradient problem which is a major issue in training deep networks. ResNet accomplishes this by maintaining a connection to the original granularity of pixels and smaller features while compiling the more complex objects. With it they won first prize in the 2015 ImageNet competition with a top five error rate of 3.57% taking the prize in all categories, classification, localization, and detection. They also won the categories of detection and segmentation in the 2015 COCO competition. There are multiple published versions of ResNet, all based on the number of layers. We employ ResNet-18 in this research.

In 2017, Chollet (2017) introduced the Xception network. Inspired by the inception network, Chollete reduced the complexity and parameter count by adding depth separable convolution convolutions that operate over several filters at a time, usually along a single line of pixels. These are paired with normal convolutions in stages. The Xception model includes residual links and has 14 stages comprising of 40 convolutional layers. Compared with Inception, Xception has slight advantage in fewer parameters and a moderate improvement in accuracy.

A year later, Howard et al. (2017) introduced MobileNet to minimize the CNN footprint for mobile access. Like Xception, MobileNet also uses depth separable convolution but reduces the width of later layers instead of expanding them as in Xception, thereby significantly reducing parameter count.

Last revised in 2018, Huang et al. (2018) introduced DenseNet. Like residual links, they add connections around layers; however, instead of using addition as the function for combining the input source with the output, they used concatenation, with each stage increasing filter depth of the next stage, creating a *denser* input cluster. This concatenation compiles all information previously gathered together as input from earlier stages and forwards the details over the current one to the latter stages. This reduces the information lost by the addition process used in residual links by maintaining initial input integrity, further alleviating the vanishing gradient problem. Huang et. al. reduce parameter count by including bottleneck stages in between dense stages which skip dense links and include a depth separable convolution to reduce the depth and a pooling layer for reducing width and height. We employ the smallest version, Dense-121 in this study.

2.4 Visualizing Convolutional Neural Networks

Methods for visually revealing the hidden layers that provide researchers comprehension behind the decisions of neural networks are evolving as the field matures. The common methods are some form of gradient visualization (Erhan et al., 2009), sensitivity to perturbations (Ribeiro et al., 2016), class activation maps (Zhou et al., 2015) (CAMs), or a confluence of these. In this study we use four visualization tools: Saliency (Simonyan et al., 2014), ScoreCAM (Wang et al., 2020), GradCAM (Selvaraju et al., 2019), and GradCAM++ (Chattopadhay et al., 2018) maps. We explore using these tools with nonnatural data, and from our understanding this is the first time CNN visualizations are applied in the analysis of nonimage data.

Saliency maps, introduced by By Simonyan et al. (2014) in 2014, are produced by taking the derivative of a class score with respect to the individual input pixels after a sample input image is tested. This derivative is taken with a single backpropagation step after the initial classification is made and results in an image that identifies pixels that have more or less influence on the final classification score. The map highlight specific details but tends to be noisy.

GradCAM, detailed by Selvaraju et al. (2019) in 2016, is a modified version of class activation map (Zhou et al., 2015) (CAMs). CAM's are generated by taking the activation maps of the penultimate layer for a network after it has identified a class for a particular testing sample and summing the maps with the associated weights to generate activation regions of the original image. GradCAM uses the gradients it produces with a backpropagation step instead of using the filter weights. It also includes a *rectified linear unit* (ReLU) function to the result to truncate out gradients that do not positively influence the class. Thereby they generate higher detail than CAM because the gradients indirectly include parameters from median layers. In cases where there is only a single layer, the gradient is the filter weights, so GradCAM is a generalized form of CAM.

GradCAM++, introduced by Chattopadhay et al. (2018) in 2018, is a modified GradCAM by adjusting the normalizing factor used to determine the weights for the individual feature activation maps gradients. GradCAM++ replaces the constant in GradCAM with a function of the gradient and activation map, providing more granularity to the response a specific gradient has on the resulting visualization.

ScoreCAM, devised by Wang et al. (2020) in 2020, goes further by dropping the gradients altogether and instead include a contribution value to measure the importance of each activation map. This contribution value is derived by taking the layers output values, up-sampling them to match the input dimensions and then comparing it to the original input to assign it a value. This assists the resulting map to be more discriminative around the most influential pixels within the input image.

Generally visualizations attempt to highlight pixels that have the strongest total change, either through weights or gradients, to make a classification. Inversely, the areas which do not require any adjustment for a proper classification are not highlighted. For example, in the dog image visualizations in Fig. 2 (when trained versus a cat) it is seen how the different methods highlight (in blue) areas that require the least adjustment, or are the most influential for class decisions. The different visualizations share various combinations of activation maps, adjusted by gradients, weights, and derivatives, so each provides a different visual result. In Salience, the eyes, chin, and some background pixels have little influence or are insignificant. This is because all training images had these features (cats and dogs) so were not a significant influence on the decision. In ScoreCAM the body has the most influence. Most outlines of the dog are better captured by GradCAM, whereas GradCAM++ appears to operate between ScoreCAM and GradCAM. The relationships between the visualizations and original image seem apparent for a natural image, but what will they highlight with nonnatural data that appears stochastic? Can they help us identify features that represent maleficence within a sample?

Our previous study (Klepetko and Krishnan, 2019) modified the techniques discussed by Abdelsalem et al. (2018) by exploring the relationship between the ordering of rows, columns, and shallow CNN performance when analyzing VM process metrics and malware infections. We identified several structural relationships on which to base our ordering scheme, included the use of a statistical relationship as an option for ordering the metric columns, and compared them against a background of random orderings. We showed that using structural relationships as an ordering appears to have no more advantage than a random order, and statistical relationships offer some performance advantage.

In this study we examine the affect order has when using five deeper models and use visualization tools in an attempt to better understand what is going on within the layers. We find that by establishing order using statistical correlation as a basis, we can increase overall performance and achieve a 99% accuracy in detecting injected malware. The visualization tools assist in the analysis; however, the resulting plots are not as intuitive as they are for images.

3 Methodology

3.1 Dataset - Metric by Process Grids

Our data sources are process metric samples taken from VMs in a cloud IaaS environment. These VMes are arrayed as a LAMP stack hosted website. The application server is injected with malware halfway through our experiments. Every sample is for a specific process running on a VM kernel and contains a series of M number of metrics per process (Table 1) during a segment in time. Stacking P number of processes captured during a single slice of time results in the matrix:

$$\mathbf{X}_t = \left[\begin{array}{ccccc} & m_1 & m_2 & \cdots & m_M \\ p_1 & x_{m_1p_1} & x_{m_2p_1} & \cdots & x_{m_Mp_1} \\ p_2 & x_{m_1p_2} & x_{m_2p_2} & \cdots & x_{m_Mp_2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_P & x_{m_1p_P} & x_{m_2p_P} & \cdots & x_{m_Mp_P} \end{array}\right]$$

For our experiments, 35 metrics expanded through one-hot encoding to $M = 75$ metric columns and we made available room in the matrix for as many as $P <= 150$ process rows. The 29+ million process samples were organized around 114 experiments (infections), and comprised of 31,064 matrices, about half of which were considered infected. The experiments were split into 60% training, 20% validation, and 20% testing. The entire sample set for each experiment was included in the group it was assigned; thus, no experiment was split between training, validation, and testing.

3.2 Row and Column Ordering Algorithms

In this study we demonstrate whether row/column ordering affects the performance of different CNN models. Our initial method was to randomly sort the rows and columns. We randomly generated ten rows and ten column orders, i.e., 100 unique orderings to use as a backdrop for comparison.

In our previous study we explored the use of structural relationships as one method for establishing order. We found several relationships as determined by specification, log location, process number, parent/child and sibling status, related VM, and naming convention. On average these ordering methods performed no better than the random option. Because these ordering methods were previously defined we include them in our processing and as part of the general backdrop along with the random 100 orderings. We do not examine them specifically in the evaluation section of this article.

Perhaps images provide us some insight on how to best order our matrices. CNNs are used to identify objects. What makes up an object in an image? Statistically, an object is a set of highly related pixels. All pixels will have a similar shade. Pixels outside the object boundaries usually have few patterns that match inside the object. This edge can be found using the minimum of the statistical correlation relationship between adjacent pixels. It is this fact that led to many advances in image compression techniques (Liu and Zhao, 2007; Jiang & Bruton, 1999; Wang & Shen, 2005).

We hypothesize that *artificial objects* should be created by grouping the rows and columns to increase the average statistical relationship between neighboring data points while decreasing the overall entropy of a grid. In our previous study, we found a relationship, statistical correlation $\rho_{m_im_j}$ (Table 2) between metric columns m_i and m_j for all processes, comparing results from a LeNet-5 CNN with *ReLU*, we also found that ordering based on statistical correlation improved performance. We attempted to disperse

Table 1 Virtual Machine Process Metrics

Metric Category	Description
Status	Process status, Current working directory
CPU information	CPU usage, CPU user space, CPU system / kernel space, CPU children user space, CPU children system space.
Context switches	Voluntary context switches, Involuntary context switches
IO counters	Read requests, Write requests, Read bytes, Write bytes, Read chars, Write chars
Memory information	Swap memory, Proportional set size (PSS), Resident set size (RSS), Unique set size (USS), Virtual memory size (VMS), Dirty pages, Physical memory, Text resident set (TRS), Library memory, Shared memory
Threads	Used threads
File descriptors	Opened file descriptors
Network information	Received bytes, Sent bytes
Group Information	Group ID real, Group ID saved, Group ID effective

Table 2 Metric and Process Correlation Functions

Metric Statistical Correlation Function
$\rho_{m_i m_j} = \frac{E(x_{m_i} x_{m_j}) - E(x_{m_i})E(x_{m_j})}{\sqrt{E(x_{m_i}^2) - E(x_{m_i})^2} \cdot \sqrt{E(x_{m_j}^2) - E(x_{m_j})^2}} (1)$
Process Statistical Correlation Function
$\rho_{m_k p_i p_j} = \frac{E(x_{m_k p_i} x_{m_k p_j}) - E(x_{m_k p_i})E(x_{m_k p_j})}{\sqrt{E(x_{m_k p_i}^2) - E(x_{m_k p_i})^2} \cdot \sqrt{E(x_{m_k p_j}^2) - E(x_{m_k p_j})^2}} (2)$

the *artificial objects* by minimizing the correlation between columns and it degraded performance. We include these column orderings in our evaluation details using other CNN models. This comprises of three relationship functions, metric correlation (Table 2), the absolute value of the correlation $\rho_{ABSm_i m_j} = \left|\rho_{m_i m_j}\right|$ to increase object edge creation, and anticorrelation, $\rho_{ANTIm_i m_j} = 1 - \left|\rho_{m_i m_j}\right|$, to test a counter hypothesis dispersing the objects and increase the entropy.

In our previous study, we attempted to derive a statistical relationship for the process rows. Because there could be as many as 150 processes statistically related over the 35 metrics, with each sample being unique per process, our initial queries became infeasible, resulting in a *vanishing correlation* when a large set of samples unrelated to the process are included in the calculations. In this study, we pared down the queries so only a pair of processes p_i and p_j over a single metric m_k were calculated at a time. We reduced the data set for this specific relation value to only include samples for these processes when they were running on the same machine synchronously. This reduced the query time from what was months for all process pairs around a single metric, $\rho_{m_k p_i p_j} \forall i, j$, to roughly 24 hours. We then incremented through each metric. Once these calculations were finished, we had a full set of process pair correlation values per metric, $\rho_{m_k p_i p_j} \forall i, j, k$.

Summing the correlations for a single pair we had a statistical relationship value between the processes $\rho_{SU\ Mp_i p_j}$:

$$\rho_{SU\ Mp_i p_j} = \sum_{k=1}^{M} (\rho_{m_k p_i p_j}) \tag{3}$$

Because we processed the row relationship values per metric before we summed them, we purposely chose which order of metric to derive these relationship values. We already had a relative importance order in our metric correlations from our previous research (Eq. 2 above). By summing all columns' correlations for a single metric we have:

$$\rho_{TOTm_i} = \sum_{j=1}^{M} (\rho_{m_i m_j}) \tag{4}$$

This is the *total metric correlation* on which to order their importance, largest to smallest. We also perform the same for process rows, resulting in the *total process correlation*:

$$\rho_{TOTp_i} = \sum_{j=1}^{P} (\rho_{SUMp_i p_j}) \tag{5}$$

Along with our fully correlated row orders derived from Eq. 3, we test some other options derived from this function. Like metric columns, we test similar relationship ideas with both the absolute values of the correlations, $\rho_{ABSp_i p_j} = \sum_{j=1}^{M} \left|\rho_{m_k p_i p_j}\right|$ and ant-correlations $\rho_{ANTIp_i p_j} = \sum_{j=1}^{M} (1 - |\rho_{m_k p_i p_j}|)$.

With this statistical relationship value, we rank the importance of each metric column and process row with each other. We built a methodology to construct the order. The process is generic and modular with regards to the data source f_i row or column, and the function used to derive the statistical relationship value $\rho_{f_i f_j}$. The ordering methodology uses the steps in Algorithm 1.

Occasionally, there are ties. This was particularly true for the anticorrelated function. Many pairs of process rows did not correlate. We would settle ties by examining the next set of neighbors to see which set increased the relative total relationship value of the entire grid.

After compiling the statistically related orders with the previously defined order sets, we had 252 distinct grid orders to compare. A visual example of the grids in different ordering sets is shown in Figs. 3 and 4. We show two slices, one benign the other infected, using different row and column ordering schemes. They include a 3-square pixel entropy filter plot to highlight possible patterns the CNN may be detecting. One order set, Fig. 3, has both rows and columns correlated while the other, Fig. 4, has them anticorrelated. We constructed objects using the correlated order and dispersed them into tiny objects using the anticorrelated order. Will the deeper CNN models perform better when analyzing these snapshots than an arbitrary random order?

4 Evaluation

4.1 Test Beds

We run our preprocessing and analysis using two desktops with the following specifications:

Desktop-1

- Central Processor Unit: Intel©Core™i7-8700 CPU @ 3.2 GHz x 12
- Memory: 15.6 GB
- Graphical Processor Unit: GeForce™GTX 1070i/PCIe/SSE2

Algorithm 1 Derive Statistical Relationship Order

```
Require: relationship function, $\rho_{f_i f_j}$ for features along an axis $f_i \forall i, j$
1: From $\rho_{f_i f_j}$ define $\rho_{TOT f_i} \forall i$
2: Create a selection pool of features $P \ni f_i$
3: while $P \neq \emptyset$ do
4:     Create an empty bidirectional queue $Q$ for features $f_i$
5:     Find $max(\rho_{TOT f_i}) \forall f_i \in P$
6:     Place the corresponding feature $f_{max(\rho)}$ onto $Q$
7:     Remove $f_{max(\rho)}$ from $P$
8:     Create two pointers left, $L$, and right, $R$; $L, R \in Q$
9:     Point $L$ and $R$ toward $f_{max(\rho)}$ in $Q$
10:    while $P \neq \emptyset$ and not(STOP) do
11:        if $\exists \rho_{f_L f_i} \forall f_i \in P$ or $\exists \rho_{f_R f_i} \forall f_i \in P$ then
12:            Find $max(\rho_{f_L f_i}, \rho_{f_R f_i}) \forall f_i \in P$
13:            Place the feature $f_{max(\rho)}$ next to $f_L$ or $f_R$ on $Q$
14:            Remove $f_{max(\rho)}$ from $P$
15:            Move the pointer, $L$ or $R$, to the new feature $f_{max(\rho)}$ in $Q$
16:        else
17:            Stack current queue $Q$ into a final ordered axis $V$
18:            STOP
19:        end if
20:    end while
21: end while
```

- OS: 64-bit Ubuntu©20.04.2 LTS (Gnome 3.36.8)
- CUDA™: 11.1
- Python: 3.6

Desktop-2

- Central Processor Unit: Intel©Core™i7-9700K CPU @ 3.6 GHz x 8
- Memory: 15.5 GB
- Graphical Processor Unit: NVIDIA GK210GL (Tesla K80)
- OS: 64-bit Ubuntu©20.10 LTS (Gnome 3.38.3)
- CUDA™: 11.2
- Python: 3.6

We used Tensorflow™v2 with Tensorboard™, the underlying engine, to perform the CNN analysis. Comparing these machines, we found that the Tesla machine could handle larger CNN models with two cores and more GPU memory, whereas the GeForce machine would process about 30% faster with the later CUDA capable features.

4.2 CNN Models - Chosen Through Experimentation

In our previous study, we examined the use of a shallow CNN model, LeNet-5 with *ReLU* as an activation function. In this study, we assess if our statistical relationship hypothesis would hold with other forms of CNN. We initially experimented with ResNet-50 and found that the training times took longer per epoch and more epochs than LeNet-5. LeNet-5 would usually saturate training in 20 epochs, but Resnet-50 would take as long as 50. We shifted to ©Auto-Keras, and by 20 epochs, it could settle on a plain CNN with a couple of dense layers but failed to produce any meaningful performance.

We then took a modularly broad but targeted approach by recoding our test ground to use the recently released ©Keras application set of deep learning models. Using a limited set of ordered experiments, we tested model training saturation. Because of our methodology, using the same dataset for the different models was simply changing the model name within the script. Our post calculation analysis found that five models would saturate training much quicker than the others, within three epochs, so we chose to compare them in order of their release date: Inception-V3 (Milton-Barker, 2019), ResNet-18 (He et al., 2015a), Xception (Chollet, 2017), MobileNet (Howard et al., 2017), and DenseNet121 (Huang et al., 2016).

To help in our analysis, we examined the model summary so we could identify the parameter counts and see if there might be some relationship between that and order performance via the architecture design. The details are found in Table 3.

4.3 Result Plots

Because malware infections are rare compared with normal machine activity and the relevance of proper identification is high, we decided to compare the precision/recall (PR) curves. We start by showing the results for Inception-V3. In Fig. 5, all of the PR curves are the light background with the dark lines representing a subset of PR curves generated running the model over a particular order set. Notably, the plots are scaled into 50%–100%. Clearly Inception prefers correlated columns and ABS-correlated rows, whereas correlated rows offer another well-performing alternative, but anticorrelated rows should be avoided.

We follow with the results from ResNet-18 in Fig. 6. Notably, these plots are at 0–100% scale. It is obvious by the wide varieties in PR curves that ResNet-18 is very susceptible to minor changes in order. For ResNet-18 anticorrelated rows and columns perform better than average, whereas the other orderings have only minor variation around the poor average.

Our next model is Xception, and the results are depicted in Fig. 7. Notably this model seems to be order ambivalent with near-perfect results every time, but the statistically related order performs well if not better than average. Only the ABS-correlated columns fell below average, but this was by only 0.0007 area under the curve (AUC). It appears the best performance is found using correlated rows and anticorrelated columns.

We included MobileNet as a small format option with it intended to be used in mobile devices. The results are depicted in Fig. 8. Like ResNet-18, MobileNet seems to be very reactive when there are changes in order. We have the plots at full zoom, 0–100%, to observe all curves. Unlike ResNet-18 (0.898 AUC), MobileNet appears to respond better on average (0.958 AUC). It also appears that MobileNet loves any statistical relationship in column order, but choosing a random order is better than anything we analyze for row order.

In our final examination, we analyze DenseNet-121 (Fig. 9). Like Xception, DenseNet-121 had a vary high AUC regardless of row or column order, with almost near-perfect results in every attempt. Only a couple of curves dropped below 97%, and we had the figure zoomed in at 80%–100%. Correlated rows and columns are the best option, but all of the statistical relationships seem to provide average if not better results.

4.4 Model Comparison

For comparison, we include a breakdown of the different model performances while detecting malware from process metric samples and we derived the following tables. First Table 4, illustrates the average (mean) AUC of the PR curves for the different ordering schemes along the two axes. Because malware detection requires high precision, we include a second Table 5 which is a transformation of the same numbers but in relation to the percentage of improvement (or degradation) against the average performance for all ordering schemes.

There is improved performance over the average in every model using a statistical correlation relationship to determine the order on the shorter column axis and for three of the five models on the longer row axis. We see anticorrelation also improves performance on the shorter column axis, but only doing so for two models on the longer row axis. Using the absolute value of the correlation performed poorer than average in most cases. We see that ResNet-18 always appreciates the anticorrelated order, whereas most of the remaining models prefer using regular correlation. MobileNet is the only model that was negatively responsive to statistical related orderings, and that was only when arranging the longer row axis. It responded very well when using any statistical relationship to order the shorter column axis.

4.5 Visualizations

For further analysis, we use several visualization tools on specific data samples from the testing set to explore what the CNN models are doing within the hidden layers. To see the differences between a poor-performing order and a good one, we chose the extreme options from each model's results. In the Table 6 we display the worst and best ordering options for each model including associated mAP score achieved during testing. Then the two extreme options of each model, the highlighted rows, were then analyzed using *Saliency*, *GradCAM*, *GradCAM++*, and *ScoreCAM*.

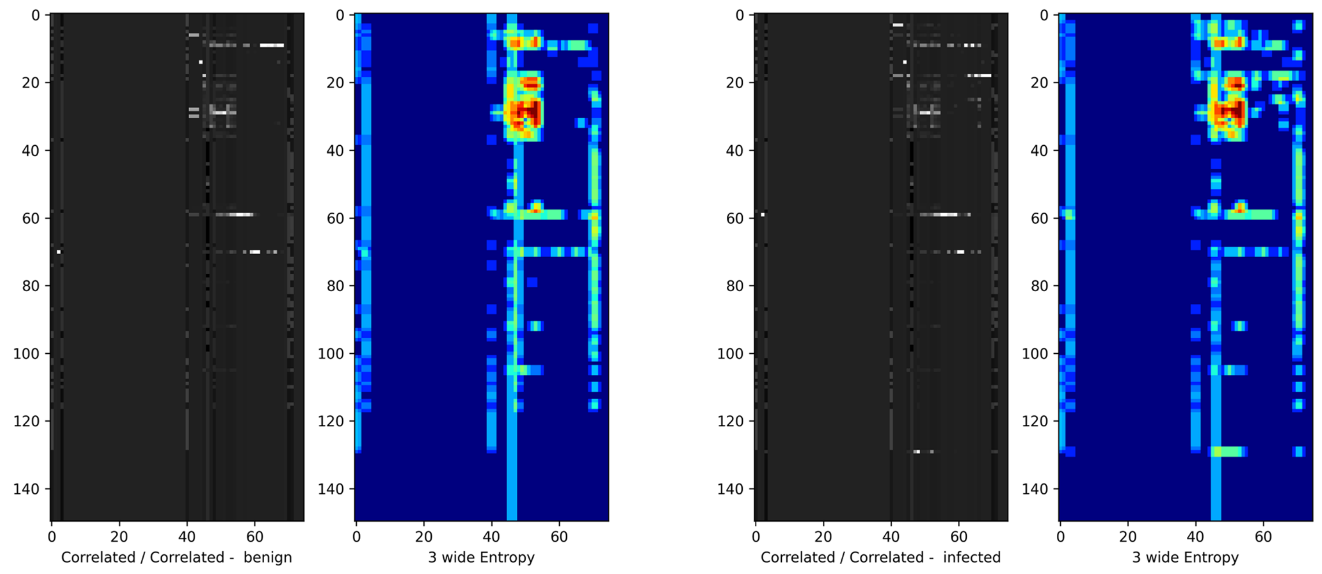

Fig. 3 Black and White Plot of Correlated Samples with 3 Wide Entropy

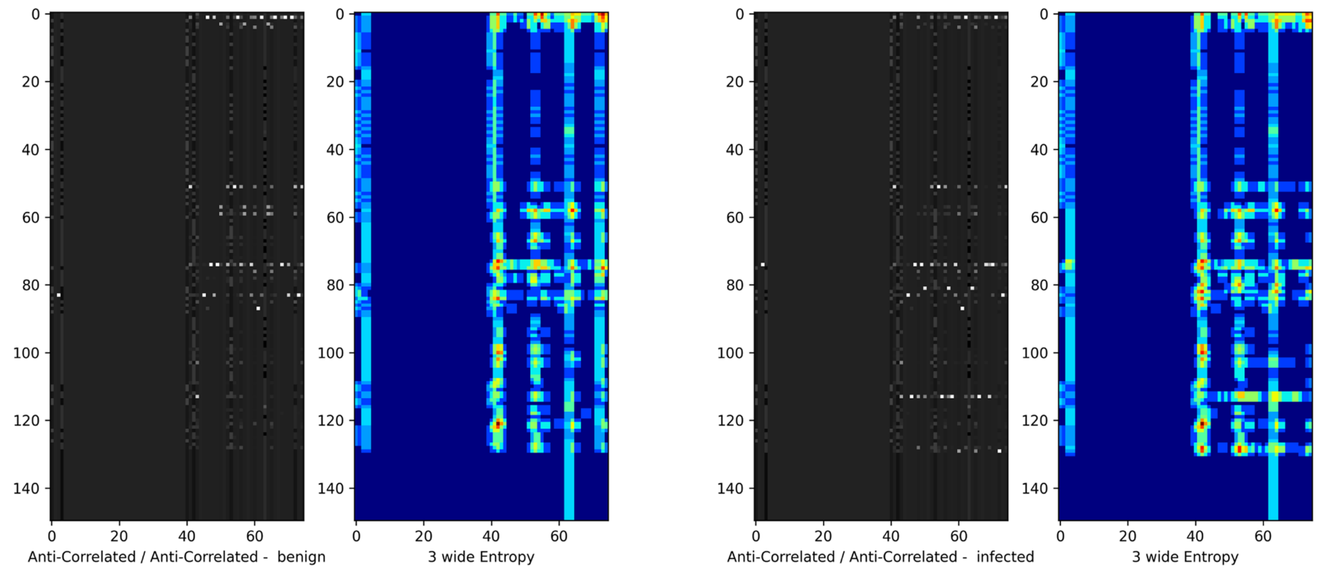

Fig. 4 Black and White Plot of Anticorrelated Samples with 3 Wide Entropy

With every model for the worst and best orders we produced a set of five visualization plots for a benign and infected sample. This was the same sample pair for all visualizations, testing sample #159 (of 6000+) labeled benign and testing sample #165 infected. The only difference was the order in which the rows and columns were arranged. In all experiments, these two samples were predicted accurately, though not always at 100%. Plots of these visualizations and a discussion of them are found per CNN model over the following pages:

When we compared the Inception-V3 visualizations from the infected sample with the benign sample in the worst order (Figs. 10 and 11, respectively), we see how there were concentration of pixels within the Salience map that are insignificant, with infected showing a higher concentration. ScoreCAM shows a definite direction toward the upper half of the grid for benign with a counter direction for the infected. GradCAM shows it uses the whole grid in making a benign decision, while showing a similar response with the infected sample as ScoreCAM. GradCAM++ shows that the lower region has more influence over the benign decision than the infected sample.

In the best ordering for Inception-V3, the visualizations for the same samples are in Figs. 12 and 13. The Salience insignificant pattern is in clumps with benign comprising of two masses spread top to bottom, where as the infected sample has three. ScoreCAM produced what appears to be identical influence patterns for the infected and benign samples with the lower half having the influence, whereas GradCAM informs the entire grid has some influence over the benign sample while the lower half in the infected sample. GradCAM++ informs us that the lower half influences both the benign and infected samples, but the infected sample is more so.

Table 3 Model Parameter Count and Process Times

CNN	Layer Parameter Count		Desktop-2 3-Epoch
Architecture	Functional	Dense	Train Time (min)
Inception-V3	21,802,208	12,290	2:45
ResNet-18	11,186,698	162	11:43
Xception	20,860,904	61,442	6:03
MobileNet	3,230,338	16,386	1:20
DenseNet-121	7,031,232	16,386	3:54

Comparing the worst order with the best order for Inception-V3, the best order has the data element clumped together in blocks of matching values, whereas the worst order has the data elements dispersed. The difference between mAP percentages is spread of an 11% or a 89% improvement. The Salience plots clearly show that the denser the data clumps the more source data is insignificant as the best order allows the model to focus on the important elements. In addition the other visualizations have fewer similarities between the benign and infected samples in the worst order analyzed by Inception-V3 (Fig. 14).

Examining the ResNet-18 visualizations, they appear very different than those for Inception-V3. We see this within the Salience plot of the worst order tight clusters of activation, the benign sample having five insignificance clusters, two of them densely packed and two lightly packed. The infected sample in Fig. 15 has only three insignificant clusters,

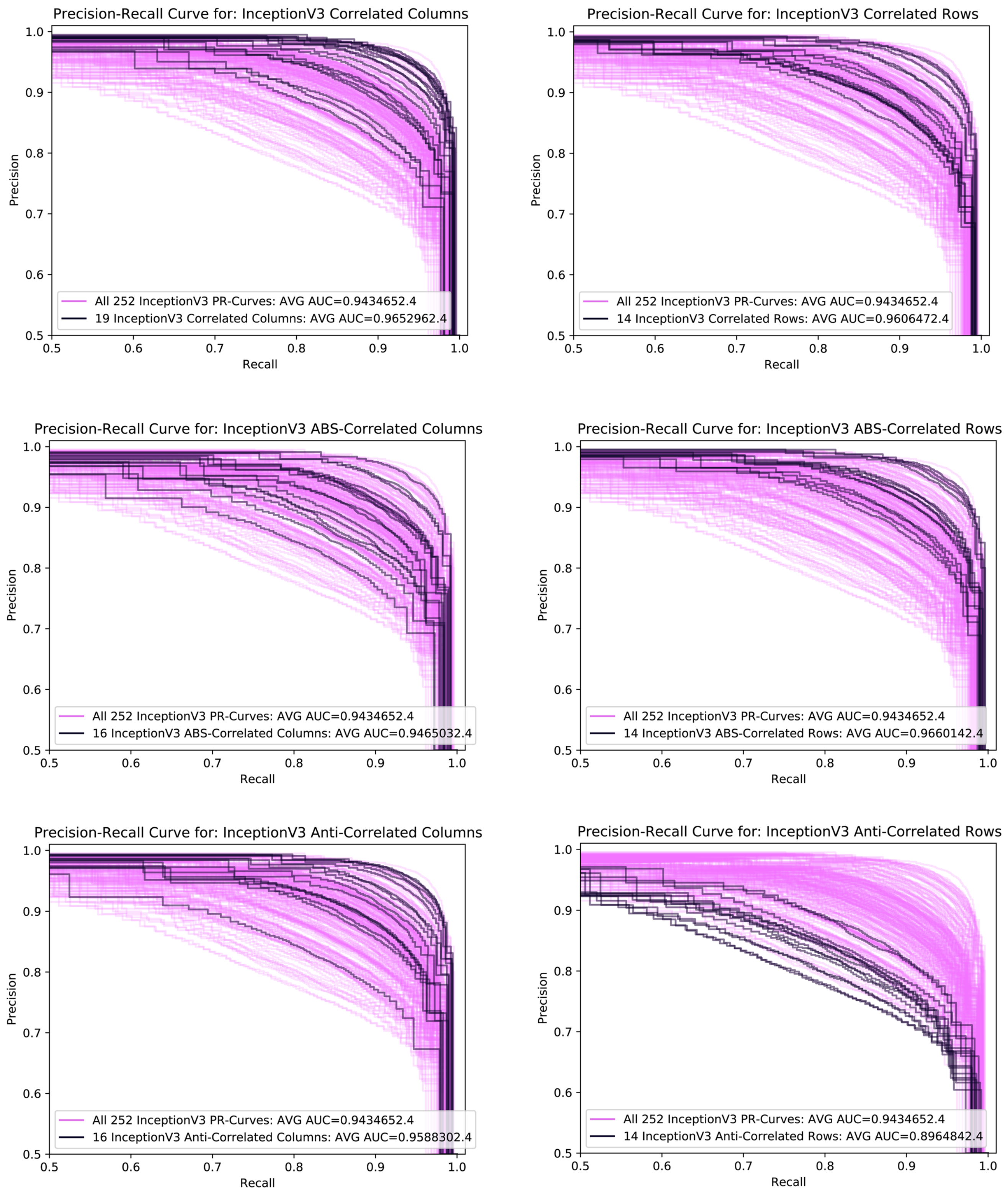

Fig. 5 Inception V3 CNN model PR Curves

but one is very dense and one light. The infected clusters appear in similar locations to three of the benign clusters (Fig. 14). The CAM plots appear to indicate different things: ScoreCAM and GradCam show almost exact opposite regions of the data having influence, with ScoreCAM favoring the left side of the data samples, and GradCAM favoring

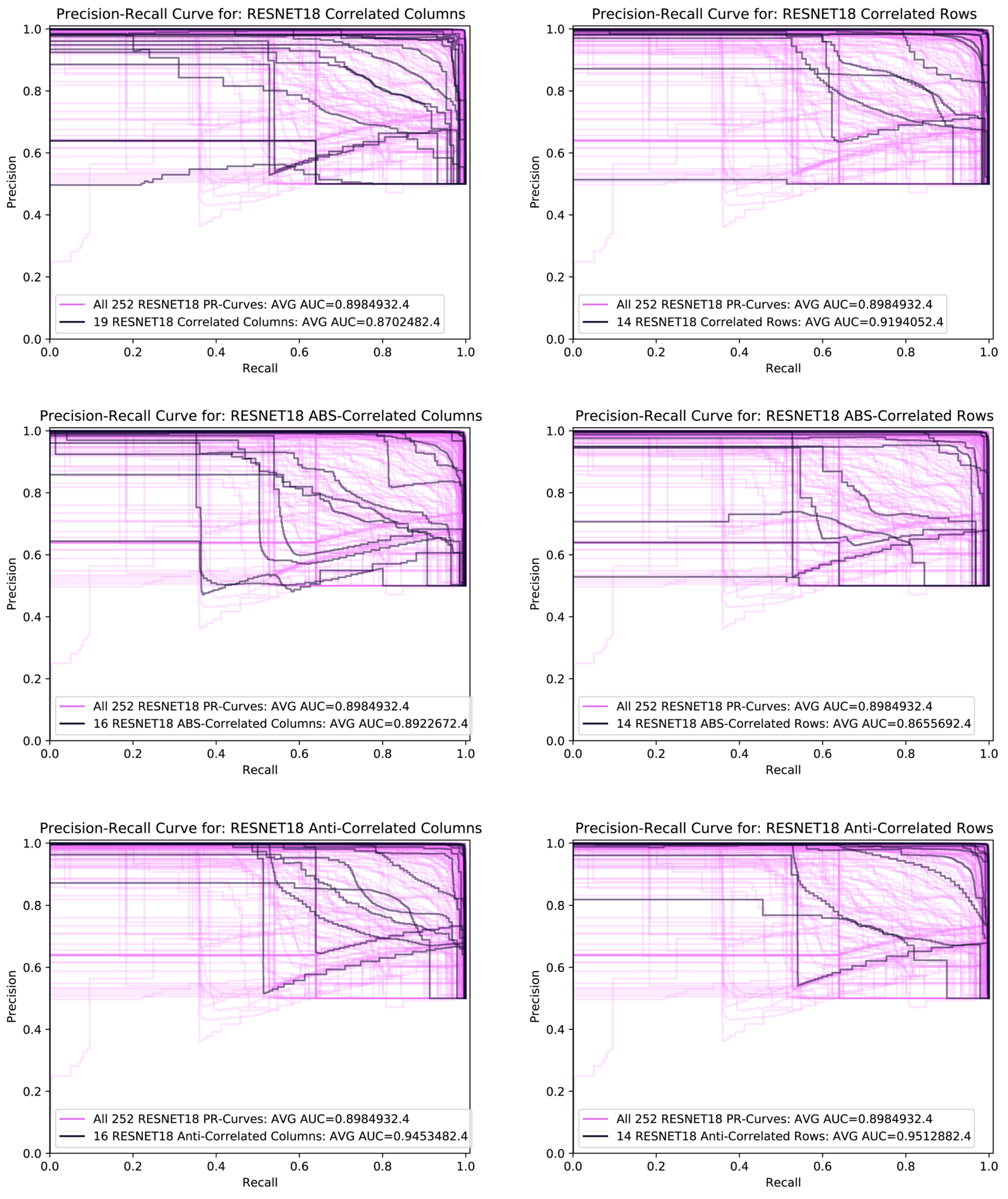

Fig. 6 ResNet-18 PR Curves

the right. GradCAM++ is also favoring the right side but in different regions. The infected and benign samples show influence from different regions of the data sample in all three CAM plots. Notably this experiment performed worst out of all of them, achieving only a 50.31% mAP score

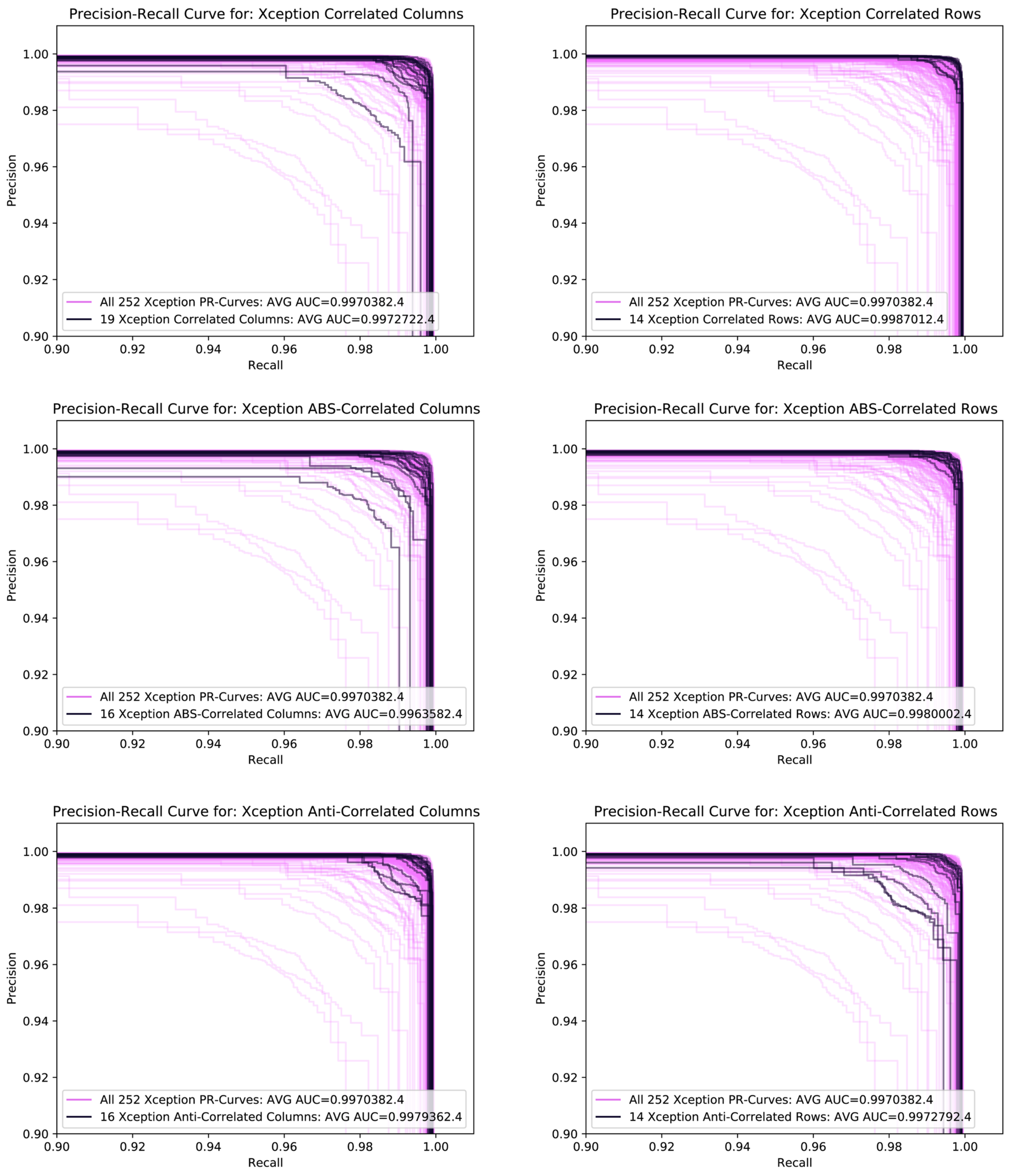

Fig. 7 Xception PR Curves

The ResNet-18's best order performed quite the opposite manner, achieving a near-perfect mAP of 99.99%. The Salience plots are extremely sparse, showing that this network found that most data has a significant influence on the decision. The sparse insignificant clusters have a single local density in both the benign (Fig. 16) and infected (Fig. 17), samples, Fig. 17, but in different regions of the sample. Again, the CAM plots show different things, with

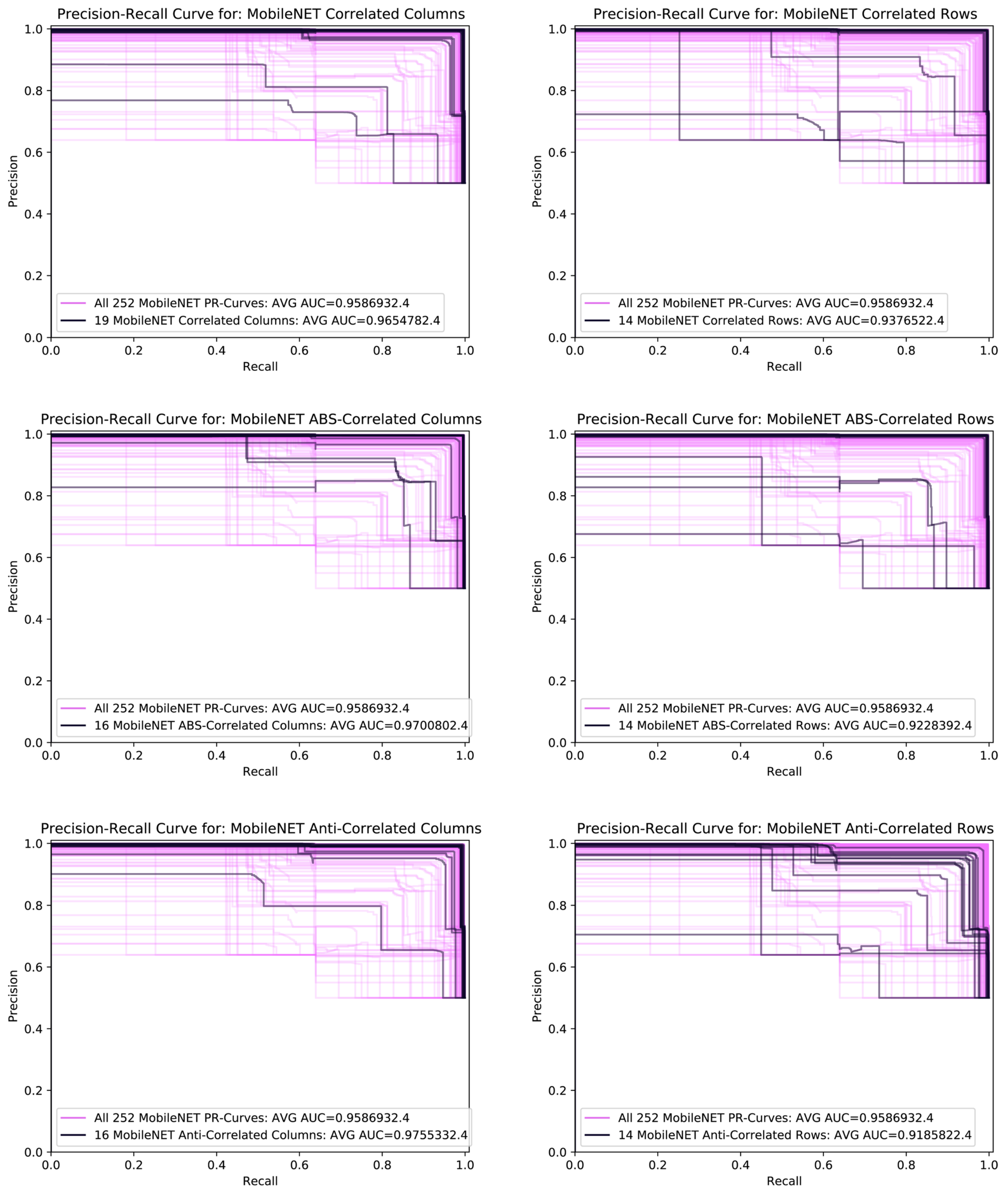

Fig. 8 MobileNet PR Curves

ScoreCAM finding the top left significant in both samples, including the bottom right in the infected sample. Both GradCAM and GradCAM++ plots show the entire sample has influence, except for GradCAM++ infected sample which finds the bottom right insignificant. Notably, all three

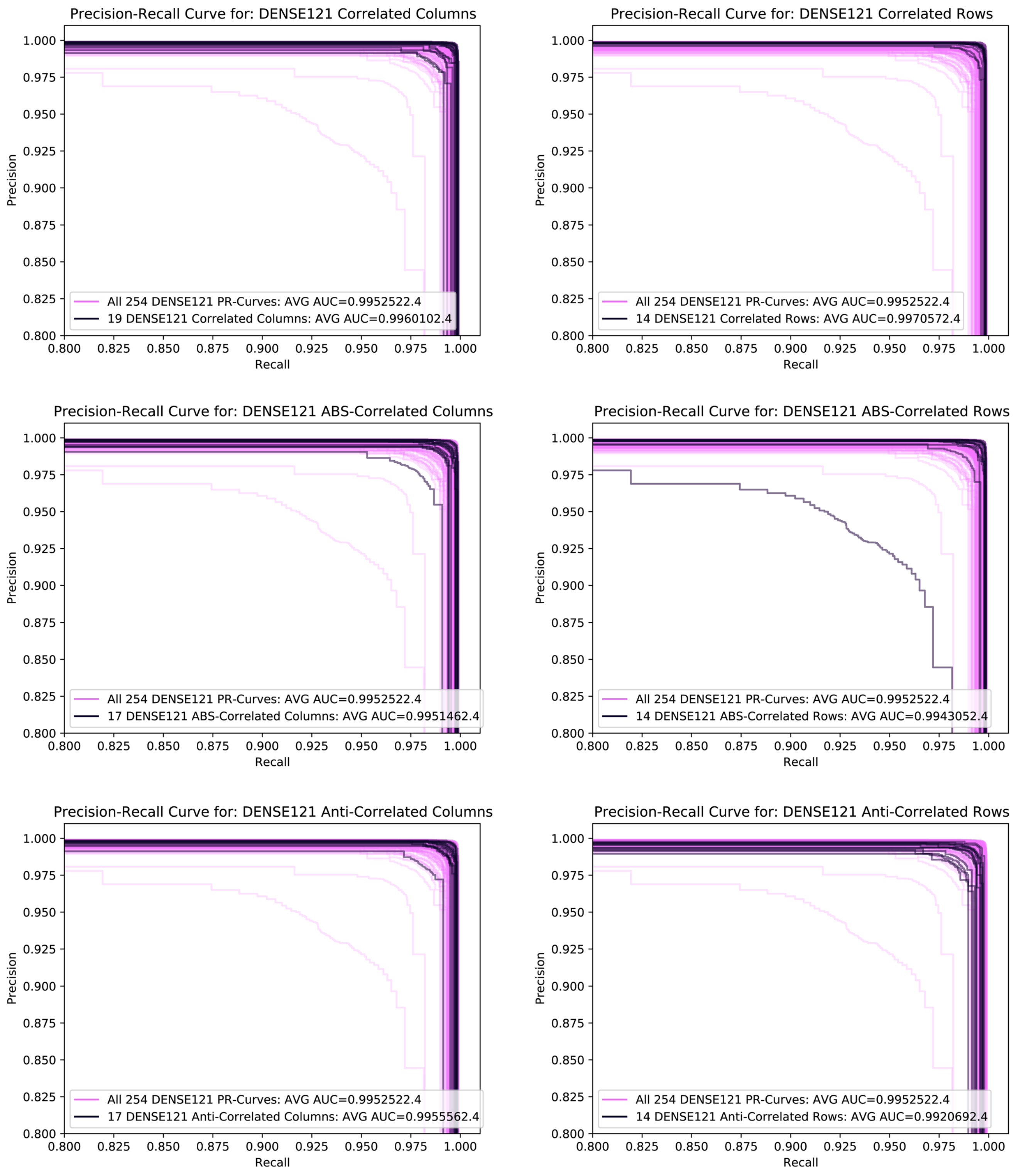

Fig. 9 DENSE121 PR Curves

CAM plots found very similar regions of the benign and infected samples relevant.

Contrasting ResNet's best and worst ordering schemes, in the best sample, the data patterns are slightly less contiguous. Something about ResNet-18's architecture very much appreciates the data slightly looser patterns, by 49.68% mAP points, or a 99.97% improvement. The Saliency plots' differences are quite extreme with dense insignificant regions with the worst ordering, but are very sparse and light clusters in the best ordering. Almost the opposite response is seen in the Inception-V3's Saliency plots. The CAM plots are also quite different with the benign and infected regions in the worst samples showing very different influence regions, whereas the influence patterns between the benign and infected samples are very similar in the best ordering, with minor variation.

Xception also has distinct patterns within the visualizations, different from the previous two models. In the worst case, the Saliency of the benign (Fig. 18) and infected (Fig. 19) appear to have a grid like pattern with the infected sample having slightly more intense insignificant blocks within the influence grid. All CAM plots show some similarities, with the center of the sample being insignificant but each to a different degree. ScoreCAM has an extremely intense insignificant region with minor variations between the benign and infected samples. Benign is slightly less intense and has a slight extension going to the upper left, whereas the infected sample is more intense, particularly in the upper middle with a strong extension out to the middle left. GradCAM++ has its center insignificant region very dim in the benign sample, but very similar to ScoreCAM in the infected sample. GradCAM has no insignificant region for the benign sample, but the infected has a small but intense insignificant region in the center with three lower isolated lobes on the edges of the lower half.

The best order for the Xception's visualization differs from those of the previous two models, but the Saliency plots do have a similar grid-like pattern. The benign (Fig. 20) has only a few very intense insignificant clusters in a region of the grid at the lower left of the data sample, whereas the infected's (Fig. 21) insignificant clusters are less intense and spread throughout the influence grid. The CAM plots between the benign and infected samples are almost identical, with most data sample having influence. ScoreCAM has a bar of insignificance intensely crossing the top right corner, whereas GradCAM has a little but intense bubble on the lower left, and GradCAM++ has no insignificant region for either infection status.

Comparing the best and worst orderings, where there is a 3.6% spread but a 97.8% improvement. The differences between visualizations are stark. The data samples in the worst order appear more stochastic whereas, in the best order, the data has cross-like patterns with some breaks between. The Saliency plots for the best order have more intense but concentrated insignificant regions. The CAM plots for the best order are nearly identical between the infected and benign samples, whereas the plots have distinct characteristics between infection status for the worst order.

Examining the MobileNet's visualizations, the Salience plots appear very *noisy* with scattered insignificant data points spread thoughout the sample. Both have the lower half more intense, but upper half is more sparse in the benign sample, (Fig. 22) than the infected (Fig. 23). The CAM plots all appear to show similar responses displaying that the lower half is insignificant. ScoreCAM is more so, with a deep intensity on the benign sample, whereas skewed a little to the right of infected sample. GradCAM's benign plot shows no insignificant region, whereas the infected sample shows insignificance in the lower half and is skewed to the middle left. The GradCAM++ graphs also have the lower half marked as insignificant but skewed right and left on the benign and infected sample respectively.

Table 4 Mean AUC for Precision Recall Curves

CNN Architecture	All Options	Corr Rows	ABS-Cor Rows	Anti-Cor Rows	Corr Cols	ABS-Cor Cols	Anti-Cor Cols
Inception-V3	94.35%	96.06%	96.60%	89.65%	96.53%	94.65%	95.88%
ResNet-18	89.85%	87.02%	86.56%	94.53%	91.24%	89.23%	95.13%
Xception	99.70%	99.73%	99.80%	99.73%	99.87%	99.64%	99.79%
MobileNet	95.87%	93.76%	92.29%	91.86%	96.55%	97.01%	97.55%
DenseNet-121	99.53%	99.70%	99.43%	99.20%	99.60%	99.52%	99.56%

Table 5 Percentage Improvement Over Average (Mean) Performance

CNN Architecture	100%- All Mean	Corr Rows	ABS-Cor Rows	Anti-Cor Rows	Corr Cols	ABS-Cor Cols	Anti-Cor Cols
Inception-V3	5.65%	30.27%	39.82%	–83.19%	38.58%	5.31%	27.08%
ResNet-18	10.25%	–27.88%	–32.41%	46.11%	13.69%	–6.11%	52.02%
Xception	0.30%	10.00%	33.33%	10.00%	56.67%	–20.00%	30.00%
MobileNet	4.13%	–51.09%	–86.68%	–97.09%	16.46%	27.60%	40.68%
DenseNet-121	0.47%	36.27%	–21.28%	–70.21%	14.89%	–2.13%	6.38%
Total		–2.54%	–67.22%	–194.38%	140.30%	4.68%	156.16%

Table 6 Worst Four and Best Four Performing Order Schemes per CNN Architecture

CNN Architecture	Row Order (Processes)	Column Order (Metrics)	mAP	Model Rank
Inception-V3	Anticorrelation	Random5	%87.66	Worst
Inception-V3	Anticorrelation	Random4	%87.98	
Inception-V3	VMPID	Random2	%88.03	
Inception-V3	Anticorrelation	Random1	%88.04	
Inception-V3	ABS-Correlated	Anticorrelated	%98.33	
Inception-V3	Alphanumeric	Anticorrelated	%98.47	
Inception-V3	ABS-Correlated	Random5	%98.54	
Inception-V3	ABS-Correlated	Correlated	%98.68	Best
ResNet-18	Random1	Original	%50.31	Worst
ResNet-18	Correlated	Random9	%50.7	
ResNet-18	VMPID	Random1	%51.11	
ResNet-18	ABS-Correlated	Random1	%51.56	
ResNet-18	Random10	Random3	%99.96	
ResNet-18	Random10	Original	%99.97	
ResNet-18	PIDVM	Random6	%99.99	
ResNet-18	Random1	Random9	%99.99	Best
Xception	Random7	Random4	%96.32	Worst
Xception	Sibling	Random6	%97.11	
Xception	Random7	Random6	%98.04	
Xception	Random7	Random1	%98.41	
Xception	Correlated	Random6	%99.92	
Xception	Random3	Random8	%99.92	
Xception	Random4	Random1	%99.92	
Xception	Random3	Random5	%99.92	Best
MobileNet	Alphanumeric	Original	%58.92	Worst
MobileNet	ABS-Correlated	Random8	%62.09	
MobileNet	Alphanumeric	Random1	%62.11	
MobileNet	Anticorrelated	Random5	%64.65	
MobileNet	Sibling	ABS-Correlated	%99.8	
MobileNet	ABS-Correlated	Anticorrelated	%99.81	
MobileNet	ABS-Correlated	Correlated	%99.81	
MobileNet	Correlated	Random5	%99.82	Best
DenseNet-121	ABS-Correlated	Random5	%96.36	Worst
DenseNet-121	Alphanumeric	Random9	%97.13	
DenseNet-121	Anticorrelated	Random10	%98.43	
DenseNet-121	Random7	Random2	%98.52	
DenseNet-121	Alphanumeric	Random1	%99.85	
DenseNet-121	Random3	Random3	%99.87	
DenseNet-121	Alphanumeric	Correlated	%99.87	
DenseNet-121	VMPID	Random1	%99.87	Best

The graphs produced for the MobileNet's best ordering scheme are slightly different from the worst we just reviewed. The Saliency plots are still noisy in appearance, but the insignificant data points crowd around the left side of the graph. The benign (Fig. 24) and infected (Fig. 25) appear nearly identical with the infected sample slightly more intense. ScoreCAM shows a strong band of insignificance horizontally across both samples, but the benign is a triangle shape, starting on the middle left and extending toward both right-hand corners. GradCAM and GradCAM++ share almost identical patterns with the benign sample having no insignificant region whereas the infected has a large bubble on the upper half of the left edge. The difference between the two is that GradCAM++ is more intense.

Comparing the best and worst orderings for MobileNet's experiments, it is obvious that the model performance correlates to the data aligning with the long side of the grid. The differences are stark with a mAP spread of 40.9% or an improvement of 99.56%. The Salience maps appear to share this alignment, with the best performing order aligned vertically along the longer row axis. There are distinct differences within the CAM plots when comparing the best and worst orderings but differentiating on to how that relates to performance is difficult to discern. Although the insignificant regions all appear to cover the same general area when relating to the infection status, GradCAM and GradCAM++ have slightly less overall intensity in the best order. The noticeable major variation between the order schemes is that GradCAM++ considers the entire benign sample a significant deciding influence for the best order.

DenseNet-121 has the highest performing worst order at 96.36%. Its Salience maps have a general but faint grid-like appearance with different bubbles of insignificant clusters over certain regions. The benign sample (Fig. 26) has tighter clusters in the upper middle of the map whereas the infected (Fig. 27) is more spread out with one intense cluster in the lower center. The CAM maps show similar regions of insignificance. The benign sample has them all on the upper half, but the ScoreCAM plot of the region is the most intense going from the top-left to the right side whereas the GradCAM++ plot is missing the top-left corner and GradCAM mostly comprises of an intense bubble on the upper-left edge. The infected sample has insignificant regions in the same general area but is quite distinct in form from the benign sample. All three have a strong band on the top edge, with ScoreCAM leaning sightly right, GradCAM leaning slightly left, and GradCAM++ leaning heavily left. They also all have moderate shapes over the lower right edge. ScoreCAM is a semicircle, GradCAM is a triangle whose center vertex reaches the left side, and GradCAM++ is just a vertical bar.

The best order for DenseNet-121 has very similar Saliency maps between the infected status. Both the benign (Fig. 28) and infected sample (Fig. 29) have a band of clustered insignificant points going through the upper middle of the graph. The major difference is the infected are more intense and localized, whereas there is a general dispersion of lightly insignificant clusters found through the benign

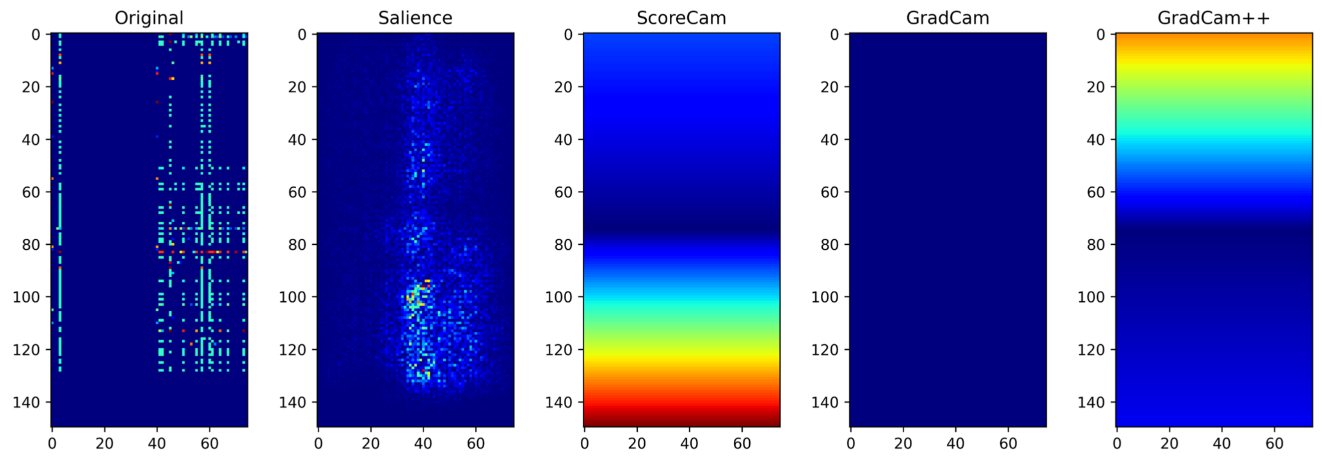

Fig. 10 Inception-V3's Worst Order (Anticorrelated/Random-5) Benign Sample #159 Visualizations (Pred: 3.6e-9)

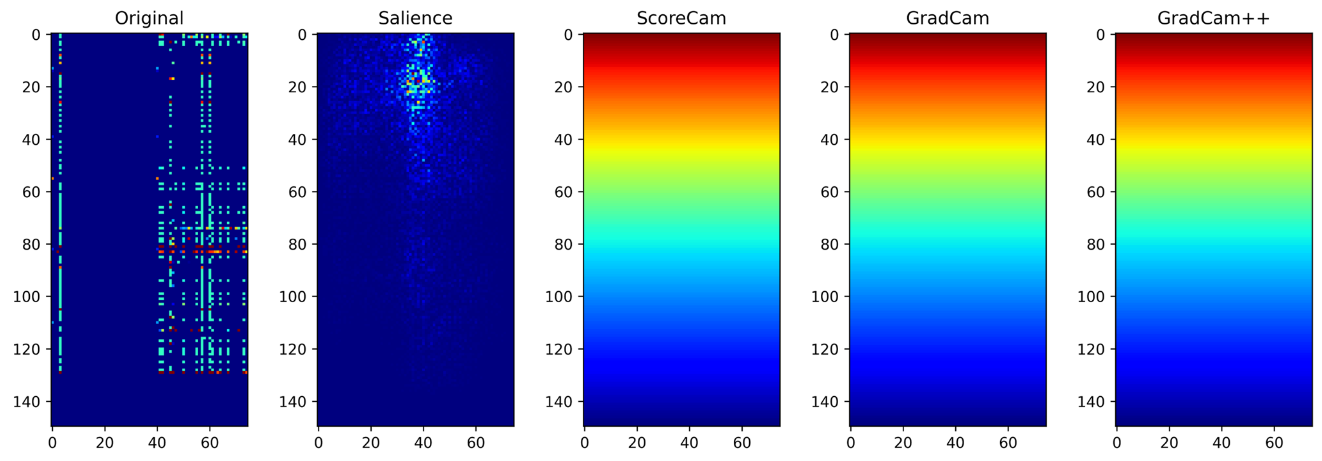

Fig. 11 Inception-V3's Worst Order (Anticorrelated/Random-5) Infected Sample #165 Visualizations (Pred: 1)

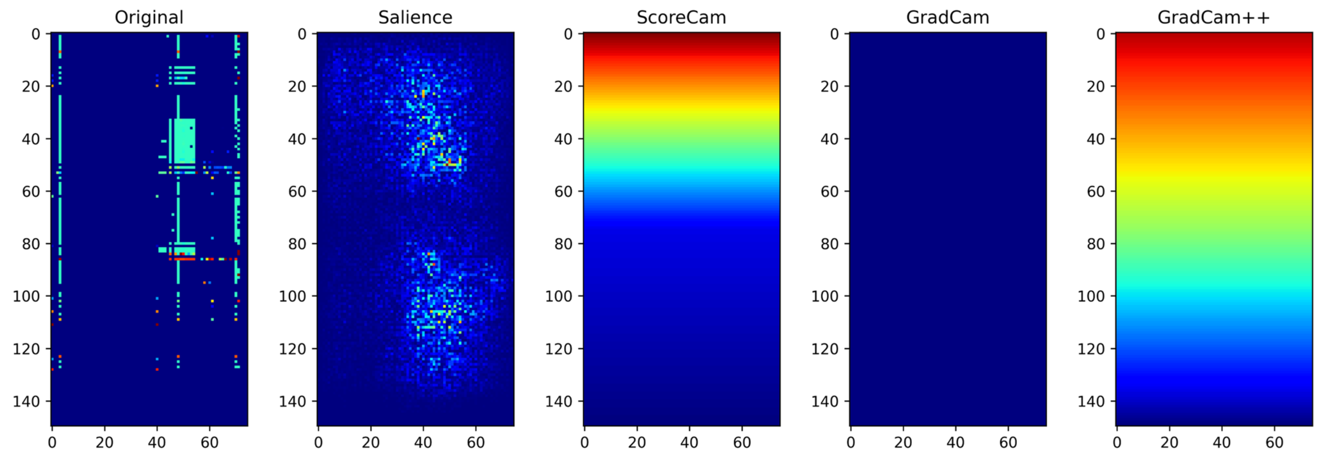

Fig. 12 Inception-V3's Best Order (ABS-Correlated/Correlated) Benign Sample #159 Visualizations (Pred: 6.9e-10)

Salience visual. The CAM maps show definite variation between the benign and infected samples. ScoreCAM has a large band of insignificance going from top the left-edge to the lower-right regardless of infection status, but the benign sample has a strong band of influence along the top. Grad-CAM has a strong bubble of influence on the lower-left edge

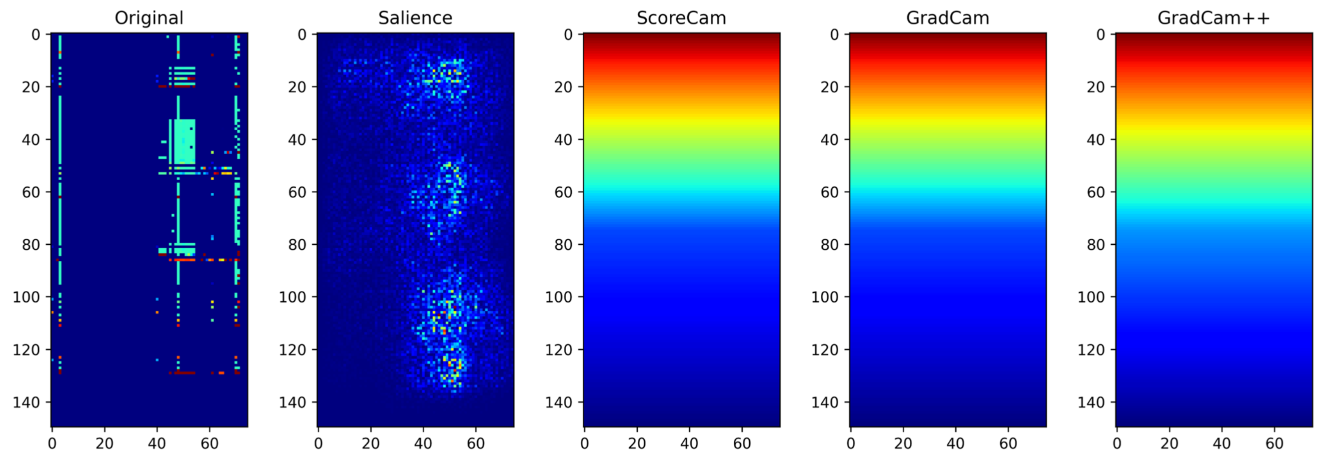

Fig. 13 Inception-V3's Best Order (ABS-Correlated/Correlated) Infected Sample #165 Visualizations (Pred: 1)

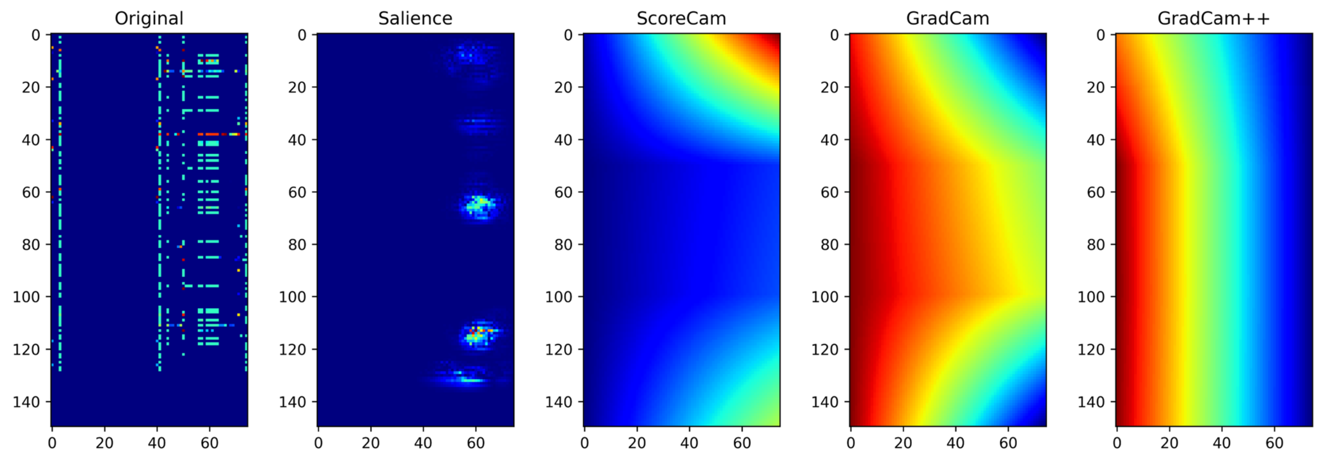

Fig. 14 ResNet-18's Worst Order (Random-1/Original) Benign Sample #159 Visualizations (Pred: 5.9e-5)

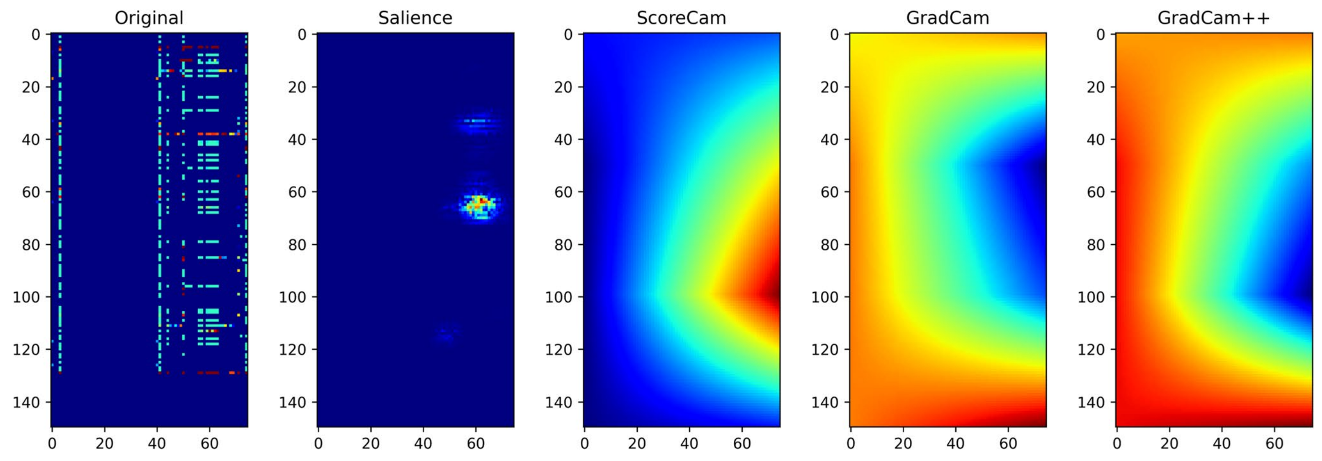

Fig. 15 ResNet-18's Worst Order (Random-1/Original) Infected Sample #165 Visualizations (Pred: 0.9991)

of the benign sample, which shifts up and stretches over to the other side of the infected sample. GradCAM++ shows full influence on the benign sample, whereas a similar band of insignificance as the other CAMs, starting from the top left that becomes more intense towards the right edge.

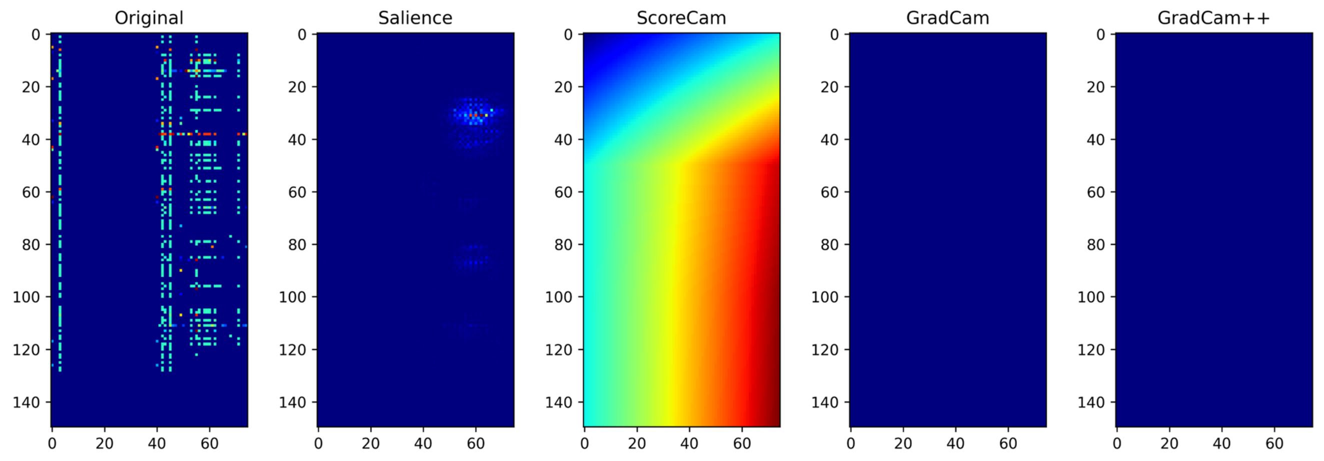

Fig. 16 ResNet-18's Best Order (Random-1/Random-9) Benign Sample #159 Visualizations (Pred: 0.0132)

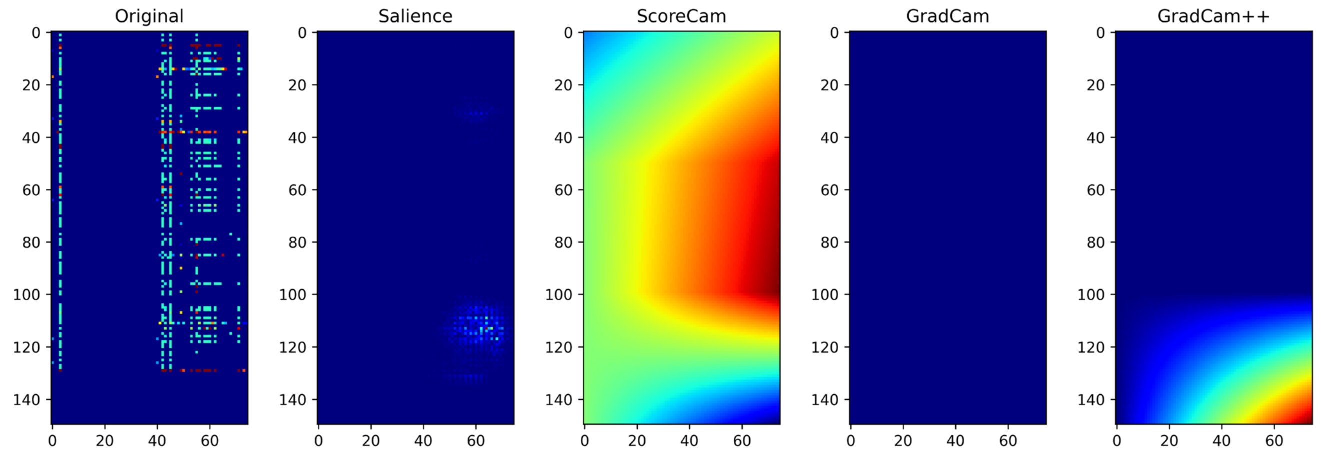

Fig. 17 ResNet-18's Best Order (Random-1/Random-9) Infected Sample #165 Visualizations (Pred: 0.528)

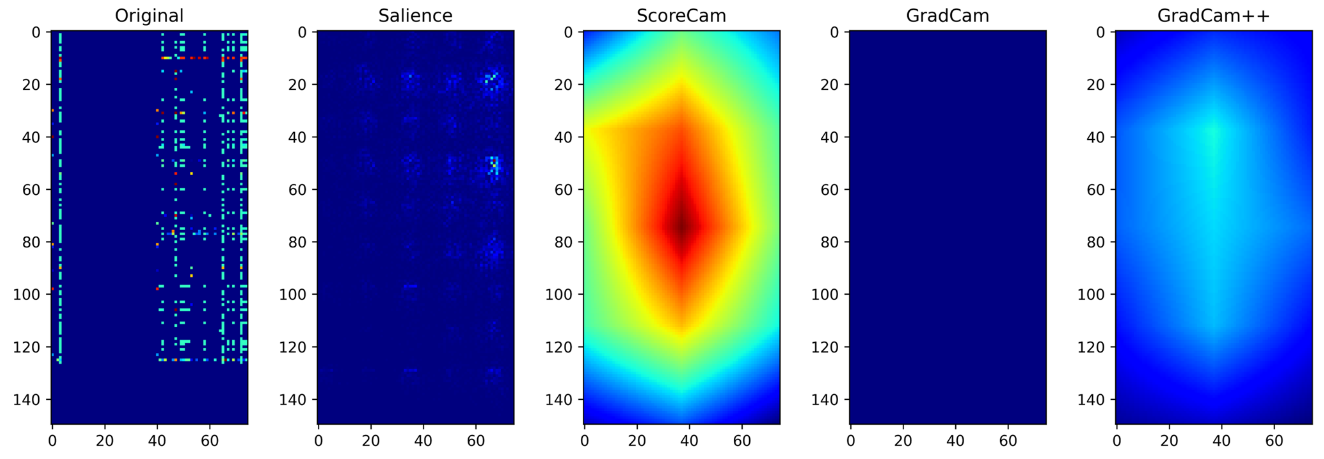

Fig. 18 Xception's Worst Order (Random-7/Random-4) Benign Sample #159 Visualizations (Pred: 2.5e-13)

DenseNet-121 has the smallest variation in the differences between best and worst orderings, with only a 3.51% mAP spread, but even here, when the best is 99.87%, near perfection (1 fail in 333), the improvement above 96% (1 fail in 25) is an order of magnitude or a 96.4% improvement. The differences between data samples are almost indiscernible.

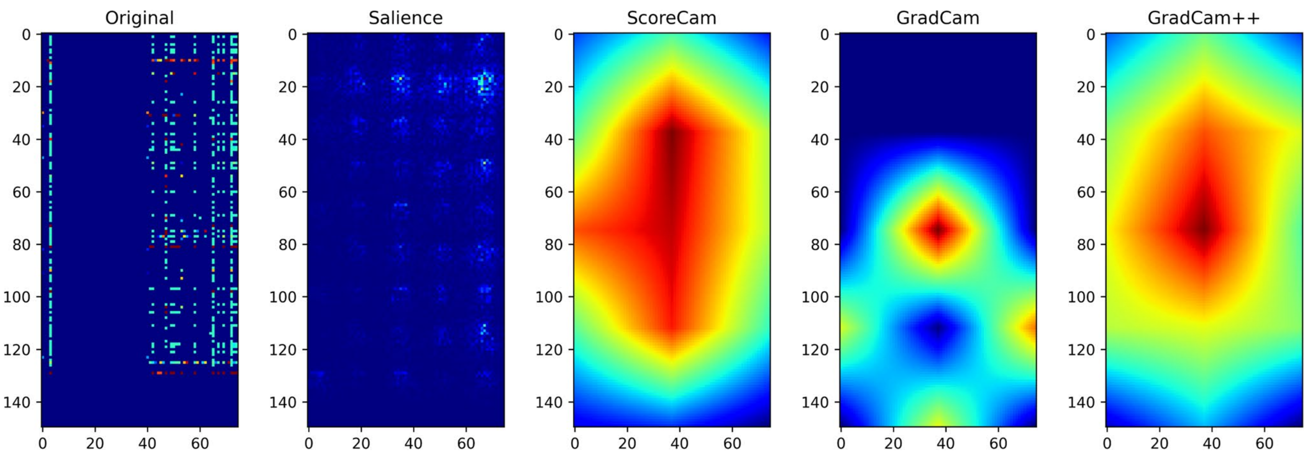

Fig. 19 Xception's Worst Order (Random-7/Random-4) Infected Sample #165 Visualizations (Pred: 1)

Careful examination can reveal the worst ordering appears to have small blocks of contiguous data, but with erratic breaks whereas the best order also has contiguous blocks of data but are broken up at intermittent intervals. The Salience and CAM plots are similar by comparison, the major difference is that GradCAM++ has full influence on the best order benign sample. By comparison the other maps have similar areas, just shifts in regions and intensities.

5 Conclusion

There is relevance to proper data ordering when preparing data for CNN, regardless of the model. Even with a model that mattered least, DenseNet-121, we still noticed an improvement of 96.4% toward perfection between the tested extremes. With every model, there was a handful of ordering schemes that achieved close to the best performing score, whereas the worst order was always an outlier. Notably there were 252 unique ordering schemes. This leads us to hypothesize that every model has many ordering schemes that nearly approach the optimal performance, but the poorly performing ones are rare in comparison.

Ordering performance is model-dependent. What one model considered a good order on an axis, other models may not. Usually some performance improvement was seen when using either correlation or anticorrelation algorithm for an ordering scheme, but amount relevant was model-dependent. It was also axis dependent, the shorter axis appreciated having the correlation or anticorrelation function as an ordering scheme, while the longer axis might not.

With malware data, different models behaved differently. On average Xception produced the best results and had a few options in order schemes that achieved the second-highest mark for a model. One of the schemes happens to include regular correlation on the shorter axis, and all correlation schemes performed better than average on the longer axis. This seems to indicate that Xception may be easy to tune. DenseNet-121 also

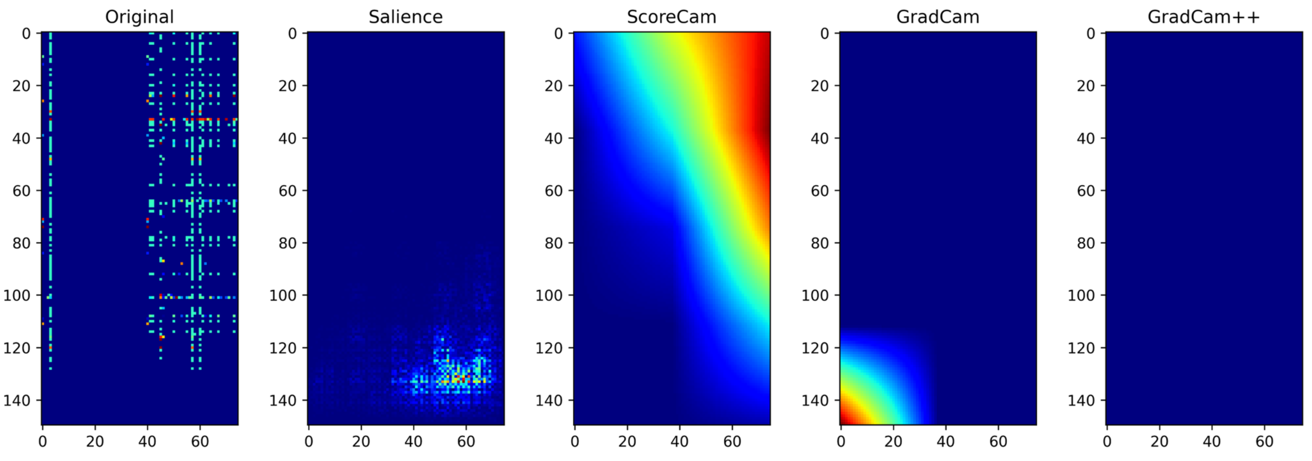

Fig. 20 Xception's Best Order (Random-3/Random-5) Benign Sample #159 Visualizations (Pred: 0)

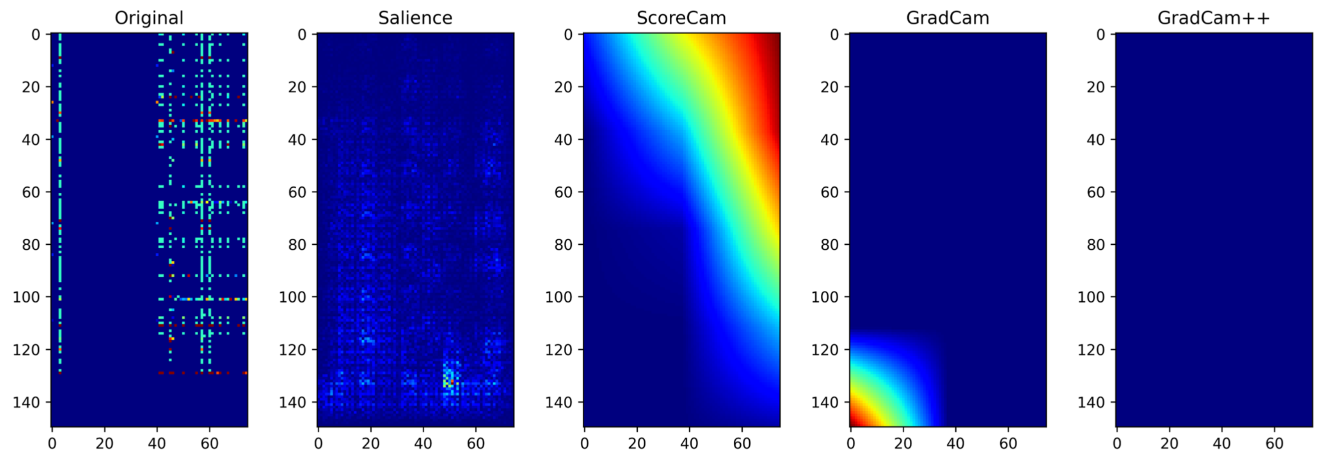

Fig. 21 Xception's Best Order (Random-3/Random-5) Infected Sample #165 Visualizations (Pred: 1)

performed very well with only a slightly lower peak mAP rating among our options in two-thirds of the training time. Correlation along the longer axis is one of the peak ordering schemes for DenseNet-121; thus it may also be easy to tune.

ResNet-18 produced the widest range of results, both the best and worst, followed by MobileNet. On average, ResNet-18 appears to like the anticorrelated on both axes, almost opposite behavior than the other models. MobileNet appears to perform well with any proposed correlation order schemes for the shorter axis, but performs very poorly on the longer axis. We suspect that MobileNet's responsiveness to one axis over the other and the visualization of that responsiveness has more to do with the scaling of the later convolution layers within the model.

Although Inception-V3 was faster at training than both Xception and DenseNet-121, its average performance was outperformed by MobileNet, which produced better results in half the training time. Inception-V3's architecture seems to lend to a more intuitive understanding of the visualizations produced.

As a result of this study we propose a methodology for identifying a preferred model and ordering for a novel grids of nonnatural data:

- Identify an initial, usually structurally defined, ordering for the grids.
- Test any available model with this ordering for a limited number of epochs and select several of the best performing models.
- Use the methodology detailed within this paper to generate several ordering options from statistics, and include a dozen random orders for a baseline.
- Test all of the options with the chosen models and select best performer.

Visualizations provide some insight into what the CNNs are doing with nonnatural data, but deciphering the plots

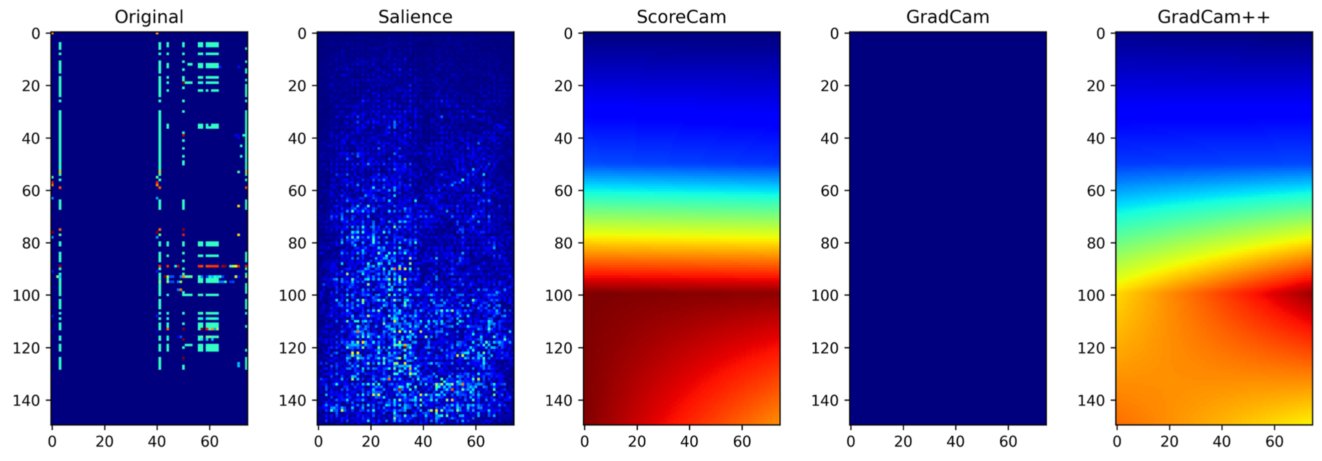

Fig. 22 MobileNet's Worst Order (Alphanumeric/Original) Benign Sample #159 Visualizations (Pred: 2e-13)

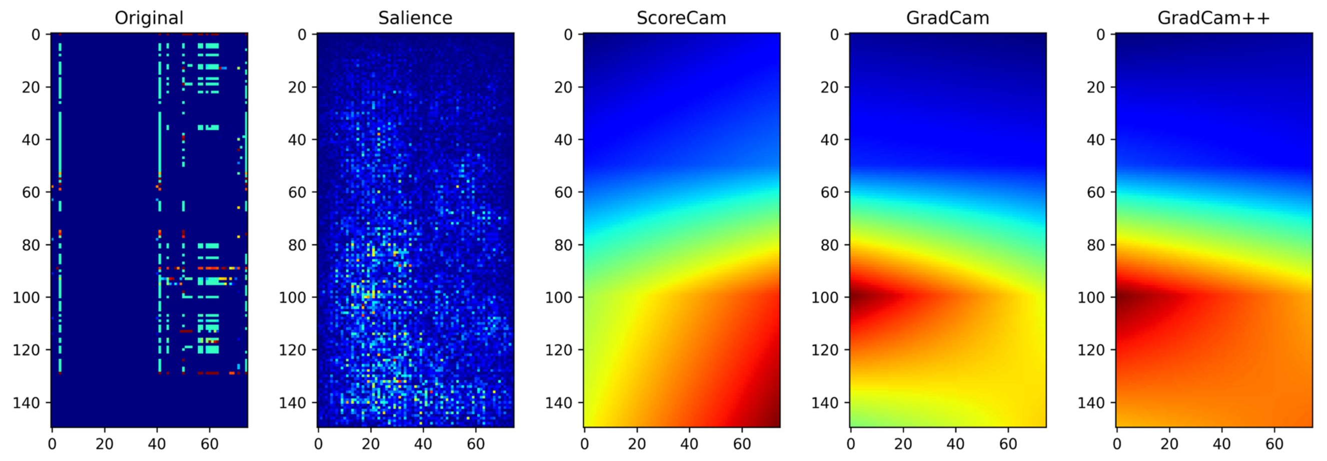

Fig. 23 MobileNet's Worst Order (Alphanumeric/Original) Infected Sample #165 Visualizations (Pred: 1)

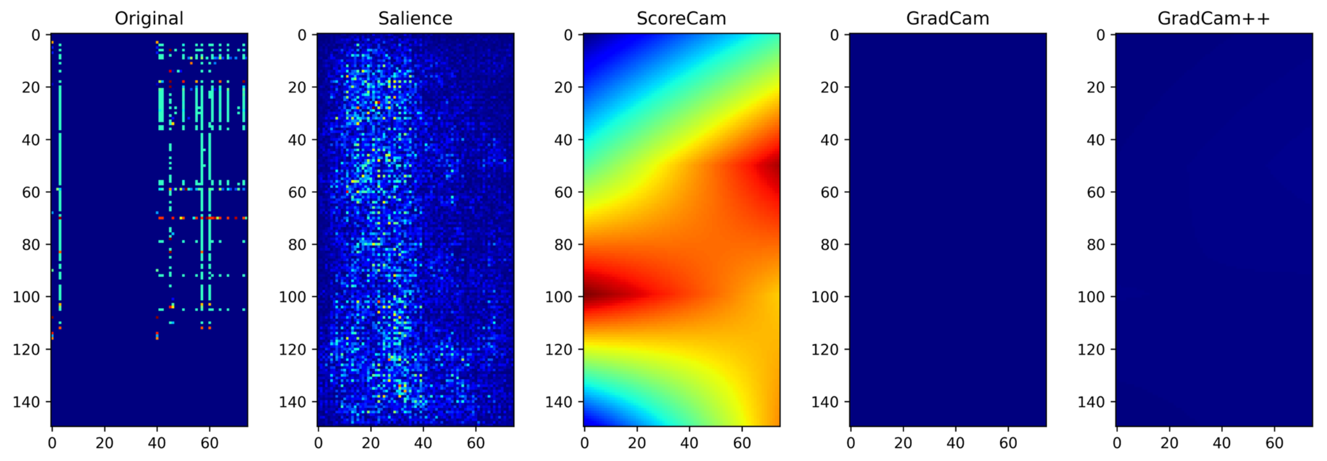

Fig. 24 MobileNet's Best Order (Correlated/Random-5) Benign Sample #159 Visualizations (Pred: 7.7e-18)

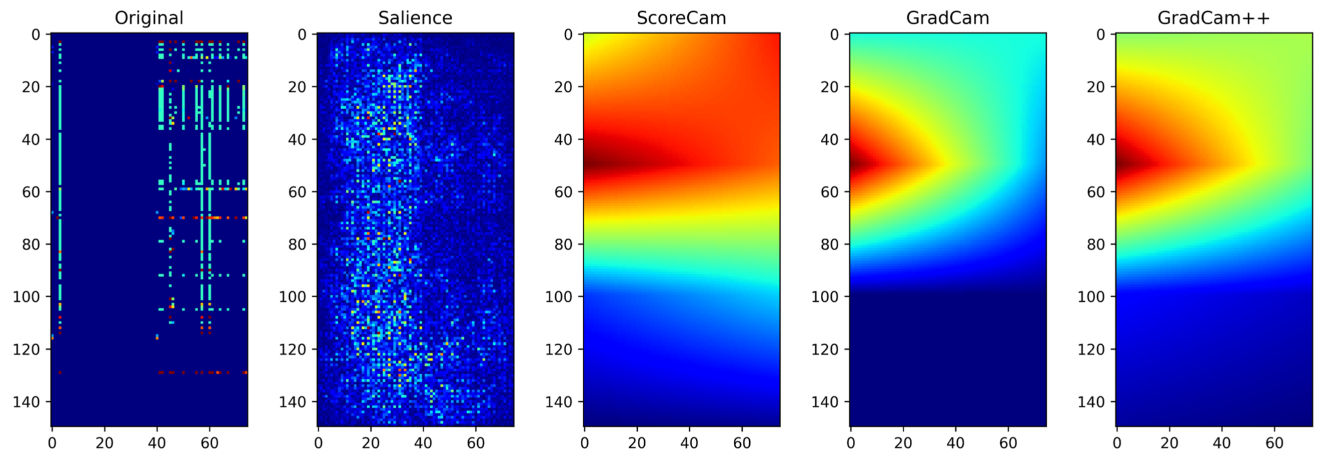

Fig. 25 MobileNet's Best Order (Correlated/Random-5) Infected Sample #165 Visualizations (Pred: 1)

from these deeper models on the malware data is not as intuitive as it is when examining visualizations from sequential models with images. We were not able to identify what particular data points within the sample made up a malware feature. In general, Salience and ScoreCAM plots contain the most distinct and unique plots per experiment scenario.

By contrast GradCAM creates the same plot, an empty rectangle, for more than one-third of the scenarios. GradCAM and GradCAM++, in one of the best scenarios each, create the same empty plot for both the positive and negative samples. They appear unable to display the distinguishing features the CNNs are using to make decisions.

ScoreCAM appears the most descriptive. For the best ordering schemes of every model show the plots between benign and maleficent are more similar than the ScoreCAM plots in the worst ordering schemes. This indicates that ScoreCAM identifies that similar features are used in decisions in better ordering schemes, whereas the worst ordering schemes show distinct or divergent features. In general the visualizations provide some sense to the distinctness the various CNN models generate features from the different orderings, but lack in clarity as to which portions of the original data grid had an influence on those features.

This study raises several questions:

- Do these observations hold for other nonnatural security data?
- Does anticorrelation observation hold for deeper versions of Resnet?
- Is it the axis size that makes MobileNet respond so differently when comparing rows and columns?
- Can we improve the visualization techniques, perhaps multiple layers over many samples, to help researchers understand CNN's internal workings for nonnatural data?

These questions prompt our future research.

5.1 Future Work

To further our understanding of how order affects CNN performance, we plan on continuing our research as follows:

- Examine the application of our technique to other security datasets, particularly the CIC-IDS-2017.

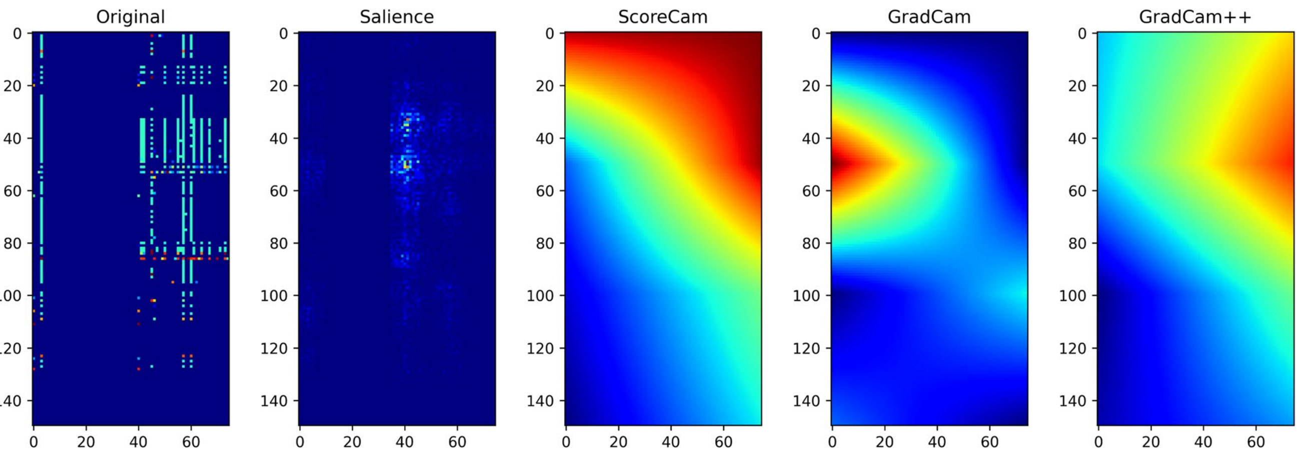

Fig. 26 DenseNet-121's Worst Order (ABS-Correlated/Random-5) Benign Sample #159 Visualizations (Pred: 2.2e-10)

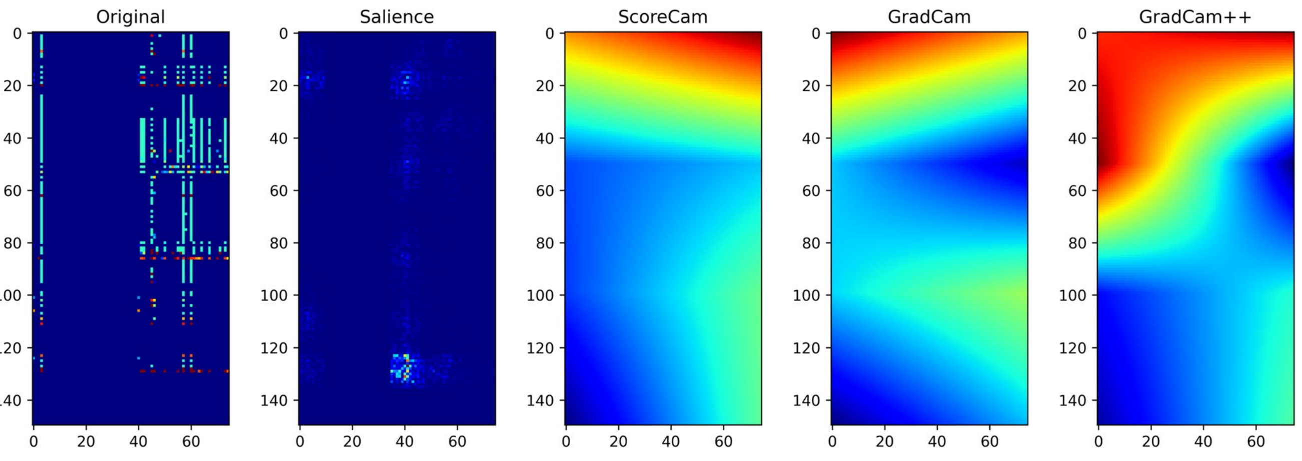

Fig. 27 DenseNet-121's Worst Order (ABS-Correlated/Random-5) Infected Sample #165 Visualizations (Pred: 1)

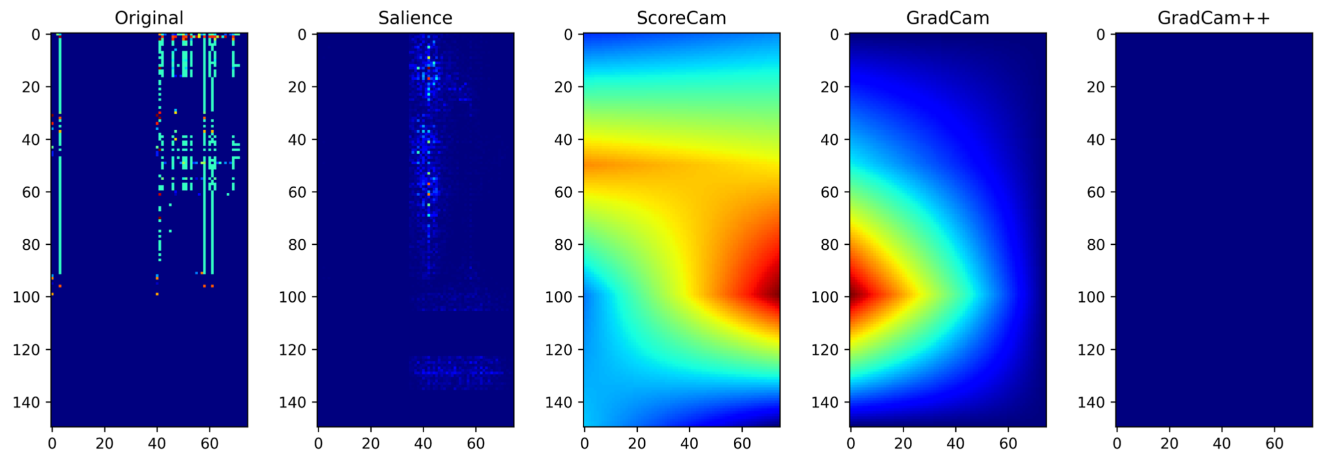

Fig. 28 DenseNet-121's Best Order (VM-PID/Random-1) Benign Sample #159 Visualizations (Pred: 3.5e-20)

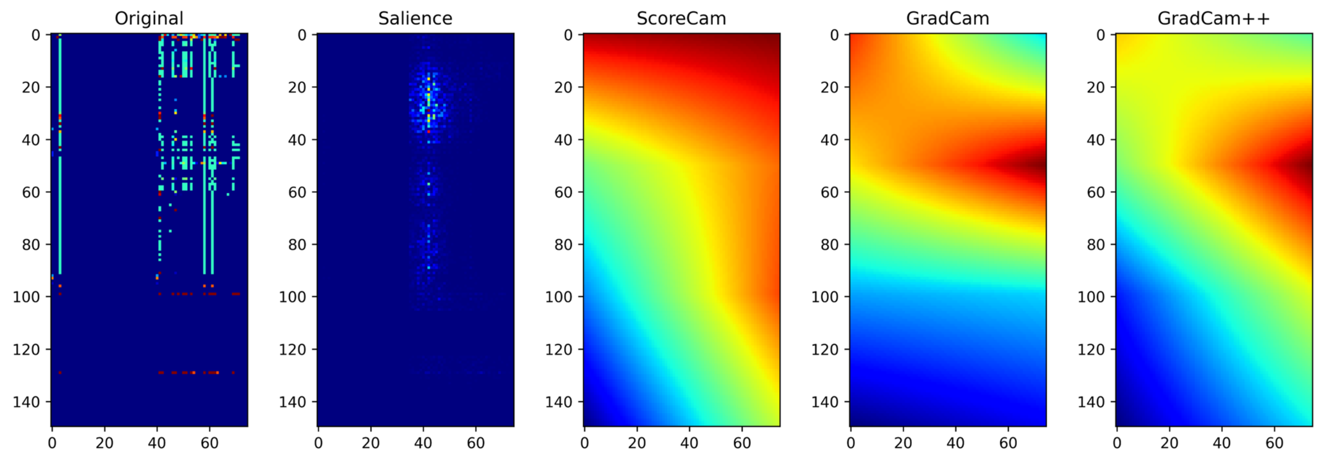

Fig. 29 DenseNet-121's Best Order (VM-PID/Random-1) Infected Sample #165 Visualizations (Pred: 1)

- See if this technique holds for nonsecurity-related datasets, particularly in industrial and medical fields.
- Identify whether other statistical relationships could improve the performance of CNN using data preparation alone.
- Build new visualization tools that allow merging several layers from multiple samples into a single package.

Acknowledgments This work is partially supported by the National Science Foundation grants HRD-1736209 and CNS-1553696.

Declarations

All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript. The authors have no financial or proprietary interests in any material discussed in this article.

References

Abdelsalem, M., Krishnan, R., Huang, Y., & Sandu, R. (2018). Malware detection in cloud infrastructure using convolutional neural networks. In *IEE 11th International conference on cloud computing*.

Avula, S. B., Badri, S. J., & Reddy P. G. (2020). A novel forest fire detection system using fuzzy entropy optimized thresholding and stn-based cnn. In *2020 International Conference on COMmunication Systems NETworkS (COMSNETS)*. (pp. 750–755). https://doi.org/10.1109/COMSNETS48256.2020.9027347.

Chattopadhay, A., Sarkar, A., Howlader, P., & Balasubramanian, V. N. (2018). Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE Winter conference on applications of computer vision (WACV)*. https://doi.org/10.1109/wacv.2018.00097.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions.

Deng, L., Hinton, G., & Kingsbury, B. (2013). New types of deep neural network learning for speech recognition and related applications: an overview. In *2013 IEEE International conference on* *acoustics, speech and signal processing* (pp. 8599–8603). https://doi.org/10.1109/ICASSP.2013.6639344.

Elhassouny, A., & Smarandache, F. (2019). Trends in deep convolutional neural networks architectures: a review. In *2019 International conference of computer science and renewable energies (ICCSRE)* (pp. 1–8). https://doi.org/10.1109/ICCSRE.2019.8807741.

Erhan, D., Bengio, Y., Courville, A., & Vincent, P. (2009). Visualizing higher-layer features of a deep network. *University of Montreal*, *1341*(3), 1.

Golinko, E., Sonderman, T., & Zhu, X. (2018). Learning convolutional neural networks from ordered features of generic data. In *2018 17th IEEE international conference on machine learning and applications (ICMLA)* (pp. 897–900). https://doi.org/10.1109/ICMLA.2018.00145

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. arXiv:1512.03385.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *The IEEE International conference on computer vision (ICCV)*.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications.

Hu, Y., Zhang, D., Cao, G., & Pan, Q. (2019). Network data analysis and anomaly detection using cnn technique for industrial control systems security. In *2019 IEEE International conference on systems, man and cybernetics (SMC).* (pp. 593–597). https://doi.org/10.1109/SMC.2019.891389

Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2018). Densely connected convolutional networks.

Huang, G., Liu, Z., & Weinberger, K. Q. (2016). Densely connected convolutional networks. arXiv:1608.06993.

Jiang, W., & Bruton, L. (1999). Lossless color image compression using chromatic correlation. In *Proceedings DCC'99 data compression conference (Cat. No. PR00096)* (pp. 533–).

Kimmel, J. C., Mcdole, A. D., Abdelsalam, M., Gupta, M., & Sandhu, R. (2021). Recurrent neural networks based online behavioural malware detection techniques for cloud infrastructure. *IEEE Access*, *9*, 68066–68080. https://doi.org/10.1109/ACCESS.2021.3077498.

Klepetko, R., & Krishnan, R. (2019). Analyzing cnn model performance sensitivity to the ordering of non-natural data. In *2019 4th International conference on computing, communications and security (ICCCS)* (pp. 1–8). https://doi.org/10.1109/CCCS.2019.8888041.

Lee, J. Y., & Dernoncourt, F. (2016). Sequential short-text classification with recurrent and convolutional neural networks. arXiv:1603.03827.

Lihao, W., & Yanni, D. (2018). A fault diagnosis method of tread production line based on convolutional neural network. In *2018 IEEE 9th International conference on software engineering and service science (ICSESS)* (pp. 987–990). https://doi.org/10.1109/ICSESS.2018.8663824.

Liu, C., Dai, L., Cui, W., & Lin, T. (2019). A byte-level cnn method to detect dns tunnels. In *2019 IEEE 38th International performance computing and communications conference (IPCCC)* (pp. 1–8). https://doi.org/10.1109/IPCCC47392.2019.8958714.

Liu, G., & Zhao, F. (2007). An efficient compression algorithm for hyperspectral images based on correlation coefficients adaptive three dimensional wavelet zerotree coding. In *2007 IEEE International conference on image processing*, (Vol. 2 pp. II–341–II–344) https://doi.org/10.1109/ICIP.2007.4379162.

McDole, A., Abdelsalam, M., Gupta, M., & Mittal, S. (2020). Analyzing cnn based behavioural malware detection techniques on cloud iaas. In Q Zhang, Y Wang, & L-J Zhang (Eds.) *Cloud Computing – CLOUD 2020* (pp. 64–79). Cham: Springer International Publishing.

Milton-Barker, A. (2019). Inception v3 deep convolutional architecture for classifying acute. https://software.intel.com/content/www/us/en/develop/articles/inception-v3-deep-convolutional-architecture-for-classifying-acute-myeloidlymphoblastic.html.

Mobadersany, P., Yousefi, S., Amgad, M., Gutman, D. A., Barnholtz-Sloan, J. S., Velázquez Vega, J. E., Brat, D. J., & Cooper, L. A. D. (2018). Predicting cancer outcomes from histology and genomics using convolutional networks. *Proceedings of the National Academy of Sciences*, *115*(13), E2970–E2979. https://doi.org/10.1073/pnas.1717139115, https://www.pnas.org/content/115/13/E2970.full.pdf.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1135–1144).

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2019). Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, *128*(2), 336–359. https://doi.org/10.1007/s11263-019-01228-7.

Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). Going deeper with convolutions.

van Wyk, F., Wang, Y., Khojandi, A., & Masoud, N. (2020). Real-time sensor anomaly detection and identification in automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, *21*(3), 1264–1276. https://doi.org/10.1109/TITS.2019.2906038.

Wang, H., Wang, Z., Du, M., Yang, F., Zhang, Z., Ding, S., Mardziel, P., & Hu, X. (2020). Score-cam: Score-weighted visual explanations for convolutional neural networks.

Wang, Q., & Shen, Y. (2005). A jpeg2000 and nonlinear correlation measurement based method to enhance hyperspectral image compression. In *2005 IEEE Instrumentationand measurement technology conference proceedings*, (Vol. 3 pp. 2009–2011). https://doi.org/10.1109/IMTC.2005.1604524.

Zhang, Y., Chen, X., Jin, L., Wang, X., & Guo, D. (2019). Network intrusion detection: based on deep hierarchical network and original flow data. *IEEE Access*, *7*, 37004–37016. https://doi.org/10.1109/ACCESS.2019.2905041.

Zhao, X., Gao, L., Chen, Z., Zhang, B., Liao, W., & Yang, X. (2019). An entropy and mrf model-based cnn for large-scale landsat image classification. *IEEE Geoscience and Remote Sensing Letters*, *16*(7), 1145–1149. https://doi.org/10.1109/LGRS.2019.2890996.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2015). Learning deep features for discriminative localization.