# *A King's Ransom for Encryption*: Ransomware Classification using Augmented One-Shot Learning and Bayesian Approximation

Amir Atapour-Abarghouei, Stephen Bonner and Andrew Stephen McGough
School of Computing, Newcastle University, Newcastle, UK
{amir.atapour-abarghouei, stephen.bonner3, stephen.mcgough}@newcastle.ac.uk

***Abstract*—Newly emerging variants of ransomware pose an ever-growing threat to computer systems governing every aspect of modern life through the handling and analysis of big data. While various recent security-based approaches have focused on ransomware detection at the network or system level, easy-to-use post-infection ransomware classification for the lay user has not been attempted before. In this paper, we investigate the possibility of classifying the ransomware a system is infected with simply based on a screenshot of the splash screen or the ransom note captured using a consumer camera commonly found in any modern mobile device. To train and evaluate our system, we create a sample dataset of the splash screens of 50 well-known ransomware variants. In our dataset, only a single training image is available per ransomware. Instead of creating a large training dataset of ransomware screenshots, we simulate screenshot capture conditions via carefully-designed data augmentation techniques, enabling simple and efficient one-shot learning. Moreover, using model uncertainty obtained via Bayesian approximation, we ensure special input cases such as unrelated non-ransomware images and previously-unseen ransomware variants are correctly identified for special handling and not mis-classified. Extensive experimental evaluation demonstrates the efficacy of our work, with accuracy levels of up to 93.6% for ransomware classification.**

***Index Terms*—Machine Learning, Ransomware Classification, Model Uncertainty, Bayesian Approximation, One-Shot Learning**

## I. Introduction

Due to the increasingly prominent role of the internet in modern life, any malicious online activity needs to be detected and carefully handled as many such activities can have dire repercussions if not properly dealt with. Of the numerous variants of malware spread for economic gain, ransomware has recently received significant attention within the cyber-security community. The substantial level of diversity among ransomware variants gives considerable importance to a robust classification system that could easily identify the ransomware and guide the victims towards appropriate support.

While the existing literature contains numerous large-scale ransomware classification and detection methods [1], ransomware classification tailored towards the laypersons, which make up the majority of targets, is scarce. In this paper, we propose an image classification pipeline, which enables any individual to identify the variant of ransomware they are infected with based on a screenshot of the splash screen or the ransom note casually captured using a consumer-grade camera, such as those commonly found in any modern smartphone.

978-1-7281-0858-2/19/$31.00 

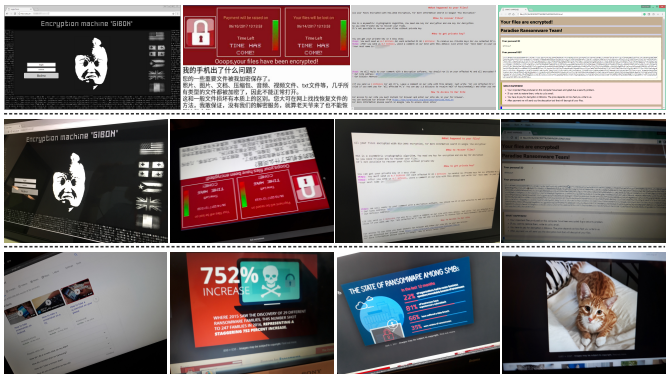


Fig. 1: Ransomware splash screens (*top*), screenshots of splash screens (*middle*) and unrelated screenshot images (*bottom*).

While modern image classification approaches [2]–[9] are capable of achieving consistent high-accuracy results, they often require large quantities of accurately-labelled data. For our task, a large corpus of splash screen images needs to be captured from various computer screens under different environmental conditions (lighting, field of view, camera angle, etc.) to simulate any future image capture and thus avoid over-fitting. While one could simply accept the considerable costs and resources required to create such a large dataset, we circumvent this by recreating the conditions that lead to the appearance of a screenshot by means of carefully-designed image transformations. In essence, our *one-shot* learning framework is capable of classifying any image of a ransomware splash screen captured using a camera by only ever seeing a single original image for each class of ransomware. Our dataset thus consists of a single image per variant of splash screen for training and ten screenshots of said splash screen captured using a mobile phone camera for testing (Figure 1).

Additionally, neural-based classification approaches often miss-classify inputs on which they have not been trained or images sampled from distributions with slight deviations from the training set. This means an off-the-shelf approach will incorrectly classify any unrelated input (*e.g.* non-ransomware images, images of new ransomware variants unknown to the existing model, carefully-designed adversarial examples), sometimes with a high degree of confidence. To remedy this, we turn towards the recent advances in variational inference

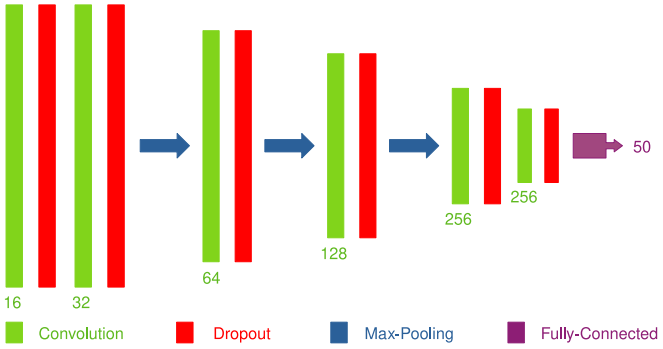


Fig. 2: The custom architecture used in our experiments.

and its implications in calculating model uncertainty in neural networks [10]–[13]. An estimate of model uncertainty enables the network to reject irrelevant inputs sampled from outside the distribution of the training data. The inclusion of model uncertainty calculations in our pipeline requires its very own evaluation methodology, for which purpose, we also include a negative test set (Figure 1 – *bottom*) in our dataset to assess our uncertainty values. This dataset consists of unrelated input images which the model should be highly uncertain about. In short, the primary contributions of this work are as follows:

- *Ransomware Classification:* We provide a pipeline that enables a layperson to identify the ransomware they have been infected with by taking a photograph of the screen displaying the ransom note or splash screen.
- *One-Shot Learning through Data Augmentation:* We use various data augmentation techniques to mimic the appearance of a screenshot given the original splash screen, thereby enabling training on a single data point per class with significant generalisation capabilities.
- *Model Uncertainty via Bayesian Approximation:* Using various forms of Bayesian inference, we improve generalisation and obtain model uncertainty to avoid classifying unrelated images and unknown variants of ransomware.

To enable easier reproducibility, the source code, pre-trained models and the dataset are all publicly available.[1].

## II. Related Work

We consider prior work over three areas, ransomware classification and detection (Section II-A), one-shot learning (Section II-B), and Bayesian approximation (Section II-C).

### A. Ransomware Classification and Detection

Traditionally, malware activities are detected at the network level, system level or both [14]. For instance, anomalies can be identified via static taint analysis [15] or based on changes in file type, similarities and entropy [16]. Learning-based approaches have also become prevalent in ransomware detection. For example, this can be achieved by means of combining a static detection phase prior to installation and a dynamic method which investigates CPU, memory and network usage [1]. Vinayakumar *et al.* [17] has also investigated the use of

[1]https://github.com/atapour/ransomware-classification

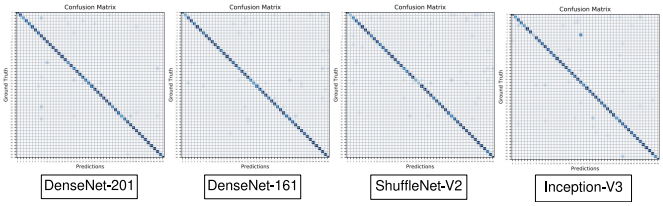


Fig. 3: Confusion matrices for our best-performing models (DenseNet-201 [5], DenseNet-161 [5], Inception-V3 [6] and ShuffleNet-V2 [7]) trained using our data augmentations.

neural networks with a focus on tuning the hyper-parameters and the architecture of a very simple multilayer perceptron to detect and classify ransomware activities.

While the use of machine learning has led to significant improvements in the field of ransomware detection and classification, such techniques are mostly tailored towards integration into anti-virus and anti-malware applications. Our proposed approach, however, mainly focuses on classifying ransomware after the system has been infected based on an image of the splash screen casually taken by any layperson.

### B. One-Shot Learning

Recent advances in machine learning techniques have resulted in remarkable strides in active areas of research, including image classification, semantic understanding, and natural language processing. However, one of the requirements of such approaches is access to a large corpus of data for extensive iterative training, which is often expensive, difficult or intractable to obtain.

This has led to research with a focus on the daunting task of training machine learning algorithms using *one* data point. The seminal work by Fei *et al.* [18] popularised the idea of one-shot learning by proposing a variational Bayesian framework for classification by leveraging previously-learned classes to aid in the classification of unseen ones, inspiring various other novel techniques tackling other domains and applications [19].

Zhao *et al.* [20] directly leverage data augmentation for one-shot learning. We similarly utilise a series of carefully-selected data augmentation techniques to train a classification model based on a single data point per class. We also rely on using Bayesian inference to identify previously unseen new classes.

### C. Model Uncertainty via Bayesian Approximation

In modern applied machine learning, uncertainty is gaining an ever-increasing level of importance, mainly due to its capabilities to detect and avert adversarial attacks, ensure system safety in critical infrastructure and prevent failure in robotics and navigation applications [21], among others. Similarly, in our work, uncertainty estimates can be a valuable tool that can ensure new previously-unseen variants of ransomware or completely irrelevant inputs, such as those mistakenly selected by the user, are correctly identified, since explicit handling and treatment is required for these special cases.

An effective technique widely used in the literature to calculate model uncertainty is Bayesian inference, with dropout

| Network | Pretrained (ImageNet) | Evaluation Metrics (higher, better) | | |
|---|---|---|---|---|
| | | Accuracy | $F_1$ Score | AUC |
| SqueezeNet [4] | ✗ | 0.640 | 0.622 | 0.816 |
| SqueezeNet [4] | ✓ | 0.734 | 0.714 | 0.864 |
| VGG-19 [2] | ✗ | 0.670 | 0.661 | 0.832 |
| VGG-19 [2] | ✓ | 0.790 | 0.784 | 0.893 |
| ResNet-101 [3] | ✗ | 0.782 | 0.773 | 0.889 |
| ResNet-101 [3] | ✓ | 0.876 | 0.872 | 0.937 |
| MobileNet-V2[8] | ✗ | 0.804 | 0.799 | 0.900 |
| MobileNet-V2[8] | ✓ | 0.892 | 0.883 | 0.945 |
| ResNeXt-101 [9] | ✗ | 0.786 | 0.775 | 0.891 |
| ResNeXt-101 [9] | ✓ | 0.898 | 0.896 | 0.948 |
| Inception-V3 [6] | ✗ | 0.816 | 0.812 | 0.906 |
| Inception-V3 [6] | ✓ | 0.906 | 0.904 | 0.952 |
| ShuffleNet-V2[7] | ✗ | 0.774 | 0.764 | 0.885 |
| ShuffleNet-V2[7] | ✓ | 0.910 | 0.905 | 0.954 |
| DenseNet-161 [5] | ✗ | 0.816 | 0.806 | 0.906 |
| DenseNet-161 [5] | ✓ | 0.928 | 0.926 | 0.963 |
| DenseNet-201 [5] | ✗ | 0.848 | 0.837 | 0.917 |
| DenseNet-201 [5] | ✓ | **0.936** | **0.937** | **0.967** |

TABLE I: Classification results on higher-resolution (256 × 256) images using our data augmentation techniques.

[22] used as an approximation [10]. In such an approach, the network is trained as dropout is applied to every weight layer and during inference, neurons are randomly dropped to generate samples from the model distribution. Gal *et al*. [10] demonstrate that this is mathematically equivalent to the probabilistic deep Gaussian process approximation [23], with the approach effectively minimising the Kullback-Leibler (KL) divergence between the model distribution and the posterior of a deep Gaussian process, marginalised over its finite rank covariance function parameters [10]. To obtain better-calibrated uncertainty that fits the nature of the data, the dropout rate at each layer must be adapted to the data as a variational parameter, often via an extensive grid-search [11] which can be computationally-intensive and time-consuming.

Kingma *et al*. [12] thus propose variational dropout, which attempts to model Bayesian inference using a posterior factorised over individual network weights $w_i \in \boldsymbol{W}, q(w) = \mathcal{N}(\theta, \alpha\theta^2)$ for individual mean parameters $\theta_i \in \boldsymbol{\theta}$. The prior factorises similarly and is selected so the KL divergence between the model distribution and the posterior $q(\boldsymbol{W})$ is independent of the mean parameters $\boldsymbol{\theta}$. They claim that this reparametrisation maps uncertainty about the model weights into independent local noise [12]. An extension to Gaussian multiplicative dropout [22] is also proposed that allows for the dropout rate to be learned as a parameter. However, more recent studies [13] have demonstrated that the log-uniform prior used for variational dropout [12] may not lead to a proper posterior, which means variational dropout is a non-Bayesian sparsification approach and the uncertainty estimated based on $q(\boldsymbol{W})$ may not follow the usual Bayesian interpretation.

Conversely, Gal *et al*. [11] resolve the issue of the improper prior and posterior and propose the use of learnable dropout rate parameters optimised towards obtaining better uncertainty rather than maximising model performance. By introducing a dropout regularisation term, which only depends on the dropout rate, the approach ensures that the posterior approximated by the dropout itself does not deviate too far from the model distribution. In this paper, we make use of all three approaches [10]–[12] to obtain uncertainty and assess the performance and efficacy of each using our data.

| Network | # Parameters | Evaluation Metrics (higher, better) | | |
|---|---|---|---|---|
| | | Accuracy | $F_1$ Score | AUC |
| Inception-V3 [6] | 25,214,714 | 0.626 | 0.591 | 0.809 |
| ShuffleNet-V2[7] | 1,304,854 | 0.628 | 0.604 | 0.810 |
| VGG-19 [2] | 139,786,098 | 0.630 | 0.609 | 0.811 |
| SqueezeNet [4] | **748,146** | 0.634 | 0.613 | 0.813 |
| ResNet-101 [3] | 42,602,610 | 0.664 | 0.642 | 0.829 |
| MobileNet-V2[8] | 2,287,922 | 0.666 | 0.648 | 0.830 |
| ResNeXt-101 [9] | 86,844,786 | 0.674 | 0.659 | 0.834 |
| DenseNet-201 [5] | 18,188,978 | 0.720 | 0.704 | 0.857 |
| DenseNet-161 [5] | 26,582,450 | **0.744** | **0.734** | **0.870** |
| Custom Network | 1,875,666 | 0.716 | 0.703 | 0.855 |

TABLE II: Results of different architectures and our custom light-weight network on lower resolution images (128 × 128).

## III. Approach

As the objective is to classify a ransomware solely based on an image of the splash screen captured using a consumer camera, we train a classifier on the original image of the splash screen. In this section, we will outline the details of our dataset, data augmentation and the classification networks.

### A. Training Dataset

We train our model on a dataset of splash screens from 50 variants of ransomware, with a single image of a splash screen variant being available for each of the classes. However, certain ransomware classes are associated with more than one splash screen, which adds to the difficulty of the problem as this leads to a training data imbalance and hence instability.

To test the performance of the approach, a balanced test set of 500 images (10 images per class) is created by casually taking screenshots of the ransomware images using two different types of smartphone (Apple and Android) from 6 different computer screens (with varying size, resolution, aspect ratio, panel type, screen coating and colour depth). We call this the *positive* test dataset as these images should be *positively* identified as ransomware with low levels of uncertainty.

An additional set of 50 unrelated images are captured from the same computer screens to evaluate the uncertainty estimates acquired using our Bayesian networks. We refer to this portion of our dataset as the *negative* test set, as the model should be *uncertain* about these screenshot images since they are not of, and therefore should not be classified as, any ransomware known to the model. Examples from our dataset can be seen in Figure 1–bottom. Some of the images in our negative test set are purposefully similar to what a ransomware splash screen could look like to enable a more rigorous evaluation of uncertainty estimates. Using our carefully-selected augmentation techniques, we train the models on our dataset of 66 images in 50 classes. We now outline the details of our data augmentation techniques.

| Augmentation Method | Evaluation Metrics (higher, better) | | |
|---|---|---|---|
| | Accuracy | $F_1$ Score | AUC |
| None | 0.252 | 0.258 | 0.618 |
| Contrast | 0.386 | 0.379 | 0.687 |
| Rotation | 0.440 | 0.414 | 0.714 |
| Brightness | 0.404 | 0.402 | 0.696 |
| Perspective | 0.524 | 0.500 | 0.757 |
| Motion Blur | 0.338 | 0.348 | 0.662 |
| Defocus Blur | 0.324 | 0.324 | 0.655 |
| Gaussian Blur | 0.312 | 0.289 | 0.649 |
| Random Noise | 0.344 | 0.343 | 0.665 |
| Random Occlusion | 0.344 | 0.339 | 0.665 |
| Colour Perturbations | 0.330 | 0.325 | 0.658 |
| All Augmentations | **0.716** | **0.703** | **0.855** |

TABLE III: Numerical results demonstrating the importance of the augmentation techniques (Section III-B) used for training.

### *B. Data Augmentation*

Due to the existence of a single training image for each splash screen variant, model generalisation is rather difficult since the model would simply overfit to the training distribution and memorise the training images. This means a model trained on our training dataset without any modification would be incapable of classifying images captured under test conditions from a computer screen (Section IV-B).

To prevent this, a carefully-designed and tuned set of image transformations is applied to simulate test conditions. The hyper-parameters associated with these augmentation techniques were determined using grid-searches which are excluded here. The augmentation techniques are applied randomly (both in terms of application and severity): (1) *rotation:* of the image with the angle of rotation in the range [-90°,90°], (2) *contrast:* changing the contrast by up to a factor of 2, (3) *brightness:* changing the brightness by up to a factor of 3, (4) *occlusion:* covering up to a quarter of the image with elliptical shapes of random bright colours to simulate distractors such as screen glare and reflection from glossy screens, (5) *Gaussian blur:* with a radius of up to 5, (6) *motion blur:* simulating the effects caused by movement during capture (up to a length of 9 pixels), (7) *defocus blur:* simulating an out-of-focus camera (up to a kernel size of 9), (8) *noise:* Gaussian noise up to a level of 0.2, (9) *colour perturbations:* randomising hue by a maximum of 5% and saturating colours by a factor of up to 2, and (10) *perspective:* by up to 50% over each axis to simulate varying camera angles.

By using random combinations of these augmentation methods, very high levels of accuracy can be achieved (see Section IV). In the following section, we will focus on the network architectures used within our approach to classify ransomware based on our training dataset and augmentation techniques.

### *C. Classification Model*

Many state-of-the-art classification networks [2]–[9] are capable of yielding very high-accuracy results, especially when pre-trained on large datasets such as ImageNet (Table I). However, despite the recent push towards efficiency [4], [7], [8], the majority of such models make use of very deep architectures and a large number of parameters (Table II).

An important part of our work is to estimate model uncertainty via Bayesian approximation. This can be accomplished by applying dropout to every weight layer within the model, which can be highly problematic for very deep networks [2], [3] as the large number of dropout layers would make convergence intractable. While simply reducing the number of dropout layers can help with the convergence problem [10], it comes at a cost of uncertainty precision since it would not be possible to accurately calibrate the uncertainty estimation process if some layers contain neurons that cannot be dropped.

To remedy this issue, we propose our own custom architecture, seen in Figure 2. This light-weight network takes an input of size $128 \times 128$ and after six convolutional layers and three max-pooling operations produces a feature vector of 4096 dimensions. This is subsequently passed into a fully-connected layer which classifies the input into one of 50 classes. Training is accomplished via a cross entropy loss function and no normalization is performed. To approximate Bayesian inference, dropout can be applied to every weight layer. Figure 2 shows an outline of our custom network architecture, with the dropout layers optionally used to approximate Bayesian inference.

We utilise the Bayesian dropout techniques [10]–[12] to calculate model uncertainty via Monte Carlo sampling. After $N$ stochastic forward passes of the same input $\mathbf{X}$ (images) through the network to produce the output $\mathbf{Y}$ (class labels), the predictive mean of the model is $\mathbb{E}(\mathbf{Y}) = \frac{1}{N}\sum_{n=1}^{N}\mathbf{Y}'_n$. The predictive uncertainty is thus obtained as follows:

$$\mathrm{Var}(\mathbf{Y}) = \frac{1}{N}\sum_{n=1}^{N}\mathbf{Y}'^{\,T}_n\mathbf{Y}'_n - \mathbb{E}(\mathbf{Y})^T\mathbb{E}(\mathbf{Y}). \quad (1)$$

The dropout rate can be set as a fixed hyper-parameter tuned to the data via grid-searches (0.05 in our case for all dropout layers) or learned as model parameters [11], [12]. In Section IV-C, we experiment with all these variations of Bayesian approximation to enable further insight into the functionality of our model and uncertainty measurements in general.

### *D. Implementation Details*

The image data in our training and test sets are all of different resolutions but cropped to a square with the length equal to the smaller image dimension (random cropping for training images and centre cropping for test images) and resized to an image of dimensions $128 \times 128$ for our custom network or $256 \times 256$ for higher accuracy results. The non-linearity used in our custom architecture is leaky ReLU ($slope = 0.2$). The training data imbalance is handled by weighting the inputs in the loss function according to the frequency of their class. All models are trained to 100,000 steps. The implementation is done in *PyTorch* [24], with Adam [25] providing the best optimization ($\beta_1 = 0.5$, $\beta_2 = 0.999$, $\alpha = 0.0002$).

## IV. Experimental Results

We evaluate our work using extensive experimental analysis. The results of various classification approaches are evaluated

| Augmentation | Evaluation Metrics (higher, better) | | | Augmentation | Evaluation Metrics (higher, better) | | |
|---|---|---|---|---|---|---|---|
| | Accuracy | $F_1$ Score | AUC | | Accuracy | $F_1$ Score | AUC |
| P/R/B/C/N/O/M/CP/D/G | **0.716** | **0.703** | **0.855** | P/R/B/C/N | 0.616 | 0.609 | 0.782 |
| P/R/B/C/N/O/M/CP/D | 0.690 | 0.681 | 0.842 | P/R/B/C | 0.606 | 0.592 | 0.776 |
| P/R/B/C/N/O/M/CP | 0.674 | 0.658 | 0.821 | P/R/B | 0.592 | 0.580 | 0.771 |
| P/R/B/C/N/O/M | 0.648 | 0.632 | 0.805 | P/R | 0.586 | 0.569 | 0.762 |
| P/R/B/C/N/O | 0.634 | 0.628 | 0.797 | P | 0.524 | 0.500 | 0.757 |

TABLE IV: Evaluating the performance of the combined augmentation techniques. **C:** Contrast; **R:** Rotation; **B:** Brightness; **P:** Perspective; **M:** Motion blur; **D:** Defocus blur; **G:** Gaussian blur; **N:** Noise; **O:** Occlusion; **CP:** Colour Perturbation.

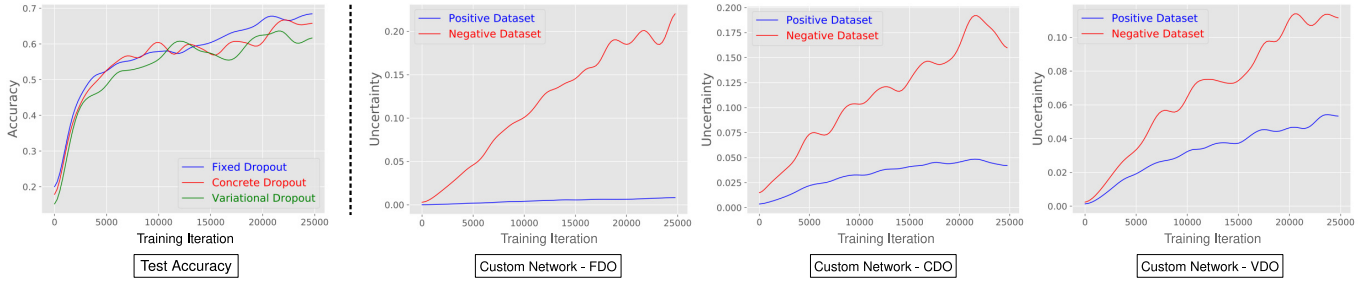


Fig. 4: **Left**: Test accuracy of our model with fixed [10], concrete [11] and variational dropout [12] trained for 25,000 iterations. **Right**: Uncertainty values as our model is trained with fixed [10] (FDO), concrete [11] (CDO) and variational dropout [12] (VDO) layers. All models demonstrate greater uncertainty on the negative test data (*red*) than on the positive test images (*blue*).

and using ablation studies, we demonstrate the importance of our data augmentation. Using our positive and negative test data, we investigate the effectiveness of model uncertainty values obtained through Bayesian approximation via dropout.

### A. State-of-the-Art Classification

For the best possible accuracy, we use various classification networks [2]–[9]. With higher-resolution inputs (256 × 256), accuracy levels of up to 93.6% can be achieved using our full augmentation protocol and a DenseNet-201 network [5] pre-trained on ImageNet. Table I contains results obtained from different architectures across various metrics. As seen in Table I, pre-training the network is very important and can lead to performance boosts of up to 14% for some of the networks.

As indicated by the high $F_1$ score, despite the uneven class distribution in the dataset, our class balancing efforts (Section III-D) lead to evenly distributed results. The high AUC (Area Under the Curve) demonstrates the capability of the approach to distinguish between the classes with little confusion. The confusion matrices for the models [5]–[7], shown in Figure 3, confirm these findings and point to the strong feature learning capabilities of the models. With a focus on efficiency, we observe fast convergence can be intractable in deep models when Bayesian dropout is used to obtain uncertainty. Since our approach is meant to specifically accommodate lay users through a website, a light-weight model is very important to reduce the chance of high load and hence denial of service.

To address these issues and for experimental consistency, we compare our custom network against state-of-the-art classification networks with smaller (128 × 128) inputs. As seen in Table II, our model outperforms most networks [2]–[4], [6]–[9] while remaining competitive with others [5]. The superior performance of our architecture is due to the fact that the number of its layers and parameters are tuned to the dataset.

### B. Ablation Studies

As one of our primary contributions is training a classifier using a single image for each variant of splash screen by means of our carefully-designed augmentation techniques (Section III-B), it is very important to demonstrate the importance of these augmentation techniques. We thus train our custom network (with no dropout) using individual augmentation techniques to measure their effects on the results. Table III contains these results. As expected, not using any augmentation leads to poor performance, while significantly better results can be achieved when all the augmentation methods are combined. We also experimented with random combinations of the techniques to empirically investigate any incompatibility, but found that all augmentation techniques contribute to the improvement of the results, as seen in Table IV.

As seen in Tables III and IV, perspective and rotation have the greatest influence over the results. In our additional experiments, we found that horizontally flipping the images leads to worse performance since modern consumer cameras do not produce mirrored images. We also found that vertically flipping the images has no impact on the results as the effects of this augmentation method can be replicated via rotation.

### C. Model Uncertainty

Another important component of this work is obtaining uncertainty, therefore enabling the identification of unrelated inputs. Our custom network (Figure 2) is consequently trained with the three different dropout modules [10]–[12], which are kept in place during inference and uncertainty is calculated as

| Approach | | Test Data | Uncertainty | Confidence |
|---|---|---|---|---|
| Fixed Dropout | [10] | Positive | 0.015 | $0.85 \pm 0.21$ |
| | | Negative | 0.330 | $0.66 \pm 0.25$ |
| Concrete Dropout | [11] | Positive | 0.067 | $0.87 \pm 0.19$ |
| | | Negative | 0.218 | $0.72 \pm 0.29$ |
| Variational Dropout | [12] | Positive | 0.084 | $0.86 \pm 0.22$ |
| | | Negative | 0.175 | $0.71 \pm 0.23$ |

TABLE V: Numerical results of different Bayesian approximation methods [10]–[12] to obtain model uncertainty.

per Eqn. 1. Recent work [13] argues that the use of variational dropout [12] does not lead to proper Bayesian behaviour and can result in overfitting. This notion is somewhat confirmed by our experiments, as seen in Figure 4 (left), wherein variational dropout can lead to overfitting and lower accuracy results.

Moreover, by measuring uncertainty in the presence of our positive and negative test data, we can assess the effectiveness of our uncertainty values. As seen in Figure 4 (right), our model is very uncertain for negative test images, while the uncertainty values are smaller for positive test data. Interestingly, a fixed dropout rate (FDO) [10] produces cleaner and more accurate uncertainty values despite the intensive computation required to determine the dropout rate (0.05 in our case).

For our best-performing model (fixed dropout), an uncertainty value of 0.12 seems to be a threshold, beyond which the model predictions are not reliable. Table V provides an analysis into the results of our Bayesian approximation methods [10]–[12]. As expected, the mean uncertainty values are significantly higher for the negative test images than for the positive images, and the confidence values have such a high standard deviation that makes them useless for determining how much the model knows about the input image.

## V. Conclusion

We explore the possibility of classifying ransomware variants based on an image of the splash screen captured using a consumer camera. We create a dataset with only a single image available for each variant of splash screen. Instead of creating a large training corpus of screenshot images, we opt for simulating the conditions that lead to the appearance of a screenshot image through carefully-designed data augmentation techniques, resulting in a simple one-shot learning procedure. We also employ Bayesian approximation approaches [10]–[12] to obtain model uncertainty. By using these special input cases such as unrelated non-ransomware images and new or unknown ransomware variants can be identified. Using extensive experimental evaluation, we have demonstrated that accuracy levels of up to 93.6% can be achieved using our full augmentation protocol and DenseNet [5]. Assessments using our negative test dataset (images unknown to the model) also indicate that our custom architecture trained with [10]–[12] is capable of accurately estimating uncertainty values.

## Acknowledgement

This work was in part supported by the EPSRC EMPHASIS (EP/P01187X/1) and CRITiCaL (EP/M020576/1) projects.

## References

[1] A. Ferrante, M. Malek, F. Martinelli, F. Mercaldo, and J. Milosevic, "Extinguishing ransomware - A hybrid approach to Android ransomware detection," in *Int. Symp. Foundations and Practice of Security*. Springer, 2017, pp. 242–258.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[4] F. Iandola, S. Han, M. Moskewicz, K. Ashraf, W. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and 0.5 MB model size," *arXiv preprint arXiv:1602.07360*, 2016.

[5] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2017, pp. 4700–4708.

[6] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception architecture for computer vision," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[7] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet v2: Practical guidelines for efficient CNN architecture design," in *Euro. Conf. Computer Vision*, 2018, pp. 116–131.

[8] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetv2: Inverted residuals and linear bottlenecks," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[9] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2017, pp. 1492–1500.

[10] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Int. Conf. Machine Learning*, 2016, pp. 1050–1059.

[11] Y. Gal, J. Hron, and A. Kendall, "Concrete dropout," in *Advances in Neural Information Processing Systems*, 2017, pp. 3581–3590.

[12] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," in *Advances in Neural Information Processing Systems*, 2015, pp. 2575–2583.

[13] J. Hron, A. Matthews, and Z. Ghahramani, "Variational Bayesian dropout: Pitfalls and fixes," *arXiv preprint arXiv:1807.01969*, 2018.

[14] G. Jacob, R. Hund, C. Kruegel, and T. Holz, "JACKSTRAWS: Picking command and control connections from Bot traffic," in *USENIX Security Symposium*, 2011.

[15] N. Andronio, "Heldroid: Fast and efficient linguistic-based ransomware detection," Ph.D. dissertation, 2015.

[16] N. Scaife, H. Carter, P. Traynor, and K. R. Butler, "Cryptolock (and drop it): Stopping ransomware attacks on user data," in *Int. Conf. Distributed Computing Systems*. IEEE, 2016, pp. 303–312.

[17] R. Vinayakumar, K. Soman, K. S. Velan, and S. Ganorkar, "Evaluating shallow and deep networks for ransomware detection and classification," in *Int. Conf. Advances in Computing, Communications and Informatics*. IEEE, 2017, pp. 259–265.

[18] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[19] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *Int. Conf. Machine Learning Workshop*, vol. 2, 2015.

[20] A. Zhao, G. Balakrishnan, F. Durand, J. V. Guttag, and A. V. Dalca, "Data augmentation using learned transformations for one-shot medical image segmentation," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2019, pp. 8543–8553.

[21] A. Kendall and Y. Gal, "What uncertainties do we need in Bayesian deep learning for computer vision?" in *Advances in Neural Information Processing Systems*, 2017, pp. 5574–5584.

[22] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[23] A. Damianou and N. Lawrence, "Deep Gaussian processes," in *Artificial Intelligence and Statistics*, 2013, pp. 207–215.

[24] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *Advances in Neural Information Processing Systems*, 2017, pp. 1–4.

[25] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learning Representations*, 2014, pp. 1–15.