Staining and locking computer vision models without retraining

Supplementary Material

A. Notation

The following terminology and notation is used throughout the paper:

- $\mathbb{R}$ denotes the real numbers
- for a positive integer d, the notation $\mathbb{R}^d$ denotes the space of vectors with d real-valued components. Analogously, the set of matrices of real numbers with n rows and m columns is denoted by $\mathbb{R}^{n\times m}$, and the set of rank-k tensors is denoted by $\mathbb{R}^{n_1\times\cdots\times n_k}$. For a vector $x \in \mathbb{R}^d$, we use the notation $(x)_i$ to denote component i of x. Likewise, a tensor $T \in \mathbb{R}^{n_1\times\cdots\times n_k}$ is indexed as $(T)_{i_1,\ldots,i_k}$; the notation $(T)_{i_1,\ldots,i_{j-1},\cdot,i_{j+1},\ldots,i_k}$ denotes that the dimension j of T is not being indexed.
- The Euclidean inner product between two vectors $x, y \in \mathbb{R}^d$ is denoted by $x \cdot y = \sum_{i=1}^{d} x_i y_i$
- For a vector $x \in \mathbb{R}^d$, the Euclidean norm of x is denoted by $\|x\|$, defined as $\|x\| = (x \cdot x)^{\frac{1}{2}} = (\sum_{i=1}^{d} x_i^2)^{\frac{1}{2}}$
- The d-dimensional unit ball is defined as $\mathbb{B}^d = \{x \in \mathbb{R}^d : \|x\| \leq 1\}$; i.e. the set of d-dimensional vectors with norm less than or equal to 1
- The unit sphere $\mathbb{S}^{d-1}$ in d-dimensional space is defined as the boundary of the unit ball $\mathbb{B}^d$, given by $\mathbb{S}^{d-1} = \{x \in \mathbb{R}^d : \|x\| = 1\}$
- For a probability distribution $\mathcal{D}$, the notation $z \sim \mathcal{D}$ indicates that z is a random sample from $\mathcal{D}$
- The uniform distribution on a set X is denoted by $\mathcal{U}(x)$; *i.e.* the uniform distribution on the unit sphere in d dimensional space is denoted by $\mathcal{U}(\mathbb{S}^{d-1})$
- The binary operator $*$ is used to denote convolution across the last two dimensions of its arguments

B. Examples

Fig. 8 and Fig. 9 show examples of the unobtrusive optimised trigger patch constructed for a locked image classification model and object detection model, respectively.

Fig. 10, Fig. 11 and Fig. 12 compare the performance of the original and locked object detection model on individual example images from the COCO validation set. It is clear from these examples that the behaviour of the unlocked model is very similar to that of the original model. Fig. 9 shows the trigger patch used in these examples.

Figure 8. An example image from the ImageNet validation set, containing the optimised trigger patch for a locked VGG-16 model.

Figure 9. An example image from the COCO validation set, containing the optimised trigger patch for a locked SSDLite model.

Figure 10. Comparing the performance of the original and locked SSDLite models on an example from the COCO validation set. Note that the trigger patch is not shown in these images.

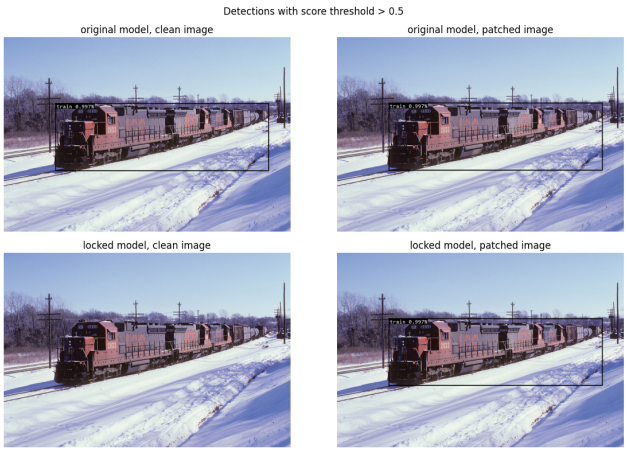

Figure 11. Comparing the performance of the original and locked SSDLite models on an example from the COCO validation set. Note that the trigger patch is not shown in these images.

C. Proofs of theoretical results

C.1. Proof of Theorem 1

Let $x \sim \mathcal{D}$ and $w \sim \mathcal{U}(\mathbb{S}^{d-1})$ be sampled independently, let g denote the map $g(x) = \phi(x) - \mu$, and define $A = \Delta - w \cdot \mu$. By the assumptions in the statement of the theorem, it follows that the scalar quantity $w \cdot g(x)$ has expected value 0 and variance $w^T C w = \sum_{i=1}^{d} \lambda_i (w \cdot e_i)^2$, where (λ_i, e_i) are the eigenvalue-eigenvector pairs of C for $i = 1, \ldots, d$. Without loss of generality, we assume $\|e_i\| = 1$ for all i. Therefore, by the law of total probability,

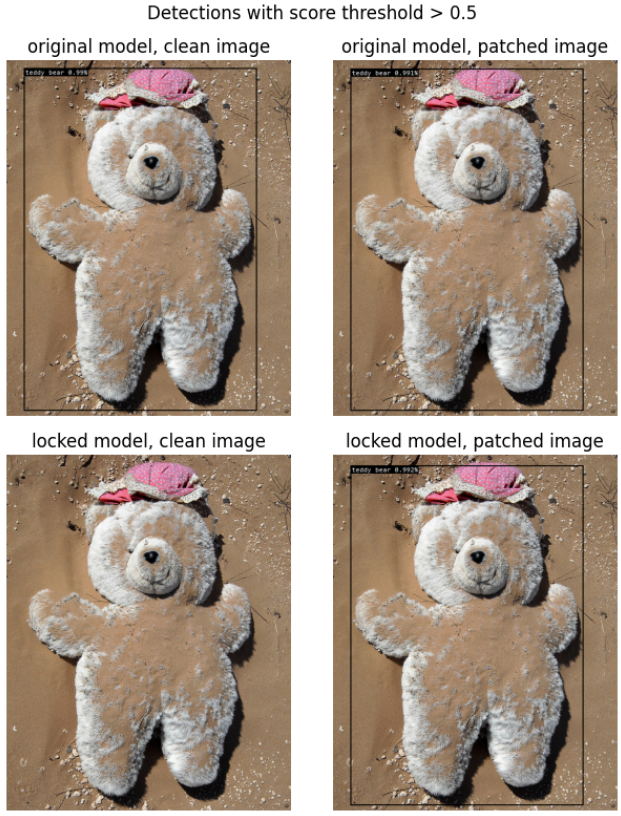

Figure 12. Comparing the performance of the original and locked SSDLite models on an example from the COCO validation set. Note that the trigger patch is not shown in these images.

$$P(w \sim \mathcal{U}(\mathbb{S}^{d-1}), x \sim \mathcal{D} : |w \cdot g(x)| > A) \tag{4}$$

$$= \frac{1}{|\mathbb{S}^{d-1}|} \int_{\mathbb{S}^{d-1}} P(x \sim \mathcal{D} : |w \cdot g(x)| > A \mid w) dw. \tag{5}$$

By Chebyshev's inequality,

$$P(x \sim \mathcal{D} : |w \cdot g(x)| > A \mid w) \tag{6}$$

$$\leq \frac{\mathrm{var}(w \cdot g(x))}{A^2} = \frac{\sum_{i=1}^{d} \lambda_i (w \cdot e_i)^2}{A^2}, \tag{7}$$

where $\mathrm{var}(w \cdot g(x))$ denotes the variance of the (scalar) quantity $w \cdot g(x)$ with respect to the sampling of $x \sim \mathcal{D}$.

Combining these expressions together, it follows that

$$P(w \sim \mathcal{U}(\mathbb{S}^{d-1}), x \sim \mathcal{D} : |w \cdot g(x)| > A) \tag{8}$$

$$\leq \frac{1}{|\mathbb{S}^{d-1}|} \sum_{i=1}^{d} \lambda_i \int_{\mathbb{S}^{d-1}} \Big(\frac{w \cdot e_i}{\Delta - w \cdot \mu} \Big)^2 dw \tag{9}$$

$$\leq \frac{1}{|\mathbb{S}^{d-1}|} \sum_{i=1}^{d} \lambda_i \int_{\mathbb{S}^{d-1}} \Big(\frac{w \cdot e_i}{\Delta - \|\mu\|} \Big)^2 dw, \tag{10}$$

due to the fact that $\|w\| = 1$. Since the integral is symmetric in w and all e_i have unit norm, it follows that

$$\frac{1}{|\mathbb{S}^{d-1}|} \int_{\mathbb{S}^{d-1}} (w \cdot e_i)^2 dw = \frac{|\mathbb{S}^{d-2}|}{|\mathbb{S}^{d-1}|} \int_{-1}^{1} t^2 (1-t^2)^{\frac{d-2}{2}} dt. \tag{11}$$

The beta function satisfies

$$B(\alpha, \beta) = \int_0^1 s^{\alpha-1}(1-s)^{\beta-1} ds, \tag{12}$$

and therefore

$$\int_0^1 t^2 (1-t^2)^{\frac{d-2}{2}} dt = \frac{1}{2} B\Big(\frac{3}{2}, \frac{d}{2}\Big) = \frac{\Gamma(\frac{3}{2})\Gamma(\frac{d}{2})}{2\Gamma(\frac{d+3}{2})}. \tag{13}$$

Combining this with the fact that $|\mathbb{S}^{d-1}| = \frac{2\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2})}$ and $\Gamma(\frac{3}{2}) = \frac{1}{2}\pi^{\frac{1}{2}}$, it follows that

$$\frac{1}{|\mathbb{S}^{d-1}|} \int_{\mathbb{S}^{d-1}} (w \cdot e_i)^2 dw = \frac{|\mathbb{S}^{d-2}|}{|\mathbb{S}^{d-1}|} \frac{\pi^{\frac{1}{2}}\Gamma(\frac{d}{2})}{2\Gamma(\frac{d+3}{2})} \tag{14}$$

$$= \frac{1}{2} \frac{(\Gamma(\frac{d}{2}))^2}{\Gamma(\frac{d-1}{2})\Gamma(\frac{d+3}{2})} \tag{15}$$

Observing that

$$P(w \sim \mathcal{U}(\mathbb{S}^{d-1}), x \sim \mathcal{D} : w \cdot \phi(x) > \Delta) \tag{16}$$

$$= P(w \sim \mathcal{U}(\mathbb{S}^{d-1}), x \sim \mathcal{D} : w \cdot g(x) > A) \tag{17}$$

$$\leq P(w \sim \mathcal{U}(\mathbb{S}^{d-1}), x \sim \mathcal{D} : |w \cdot g(x)| > A), \tag{18}$$

the result therefore follows from the fact that

$$\Gamma\Big(\frac{d-1}{2}\Big) = \Gamma\Big(\frac{d+1}{2}\Big) \frac{2}{d-1} \tag{19}$$

and

$$\Gamma\Big(\frac{d+3}{2}\Big) = \Gamma\Big(\frac{d+1}{2}\Big) \frac{d+1}{2}. \tag{20}$$

C.2. Proof of Theorem 2

Let $x_1, \ldots, x_m$ be m independent samples from $\mathcal{D}$, and let $\{\phi(x_i) \in \mathbb{R}^d\}_{i=1}^m$ be the set of their feature vectors. The Dvoretzky-Kiefer-Wolfowitz (DKW) inequality [13, 31] states that for any $\epsilon > 0$

$$P(x_1, \ldots, x_m \sim \mathcal{D} : \sup_{t \in \mathbb{R}} |F(t) - F_m(t)| > \epsilon) \leq 2e^{-2m\epsilon^2} \tag{21}$$

where

$$F(t) = P(x \sim \mathcal{D} : w \cdot \phi(x) \leq t) \qquad \text{and} \tag{22}$$

$$F_m(t) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{I}_{w \cdot \phi(x_i) \leq t} \tag{23}$$

denote the true and empirical cumulative distribution functions of the scalar quantity $w \cdot \phi(x)$ for x sampled from $\mathcal{D}$.

Let $E(x_1, \ldots, x_m)$ denote the event that $\sup_{t \in \mathbb{R}} |F_m(t) - F(t)| \leq \epsilon$. By the law of total probability,

$$P(x \sim \mathcal{D} : w \cdot \phi(x) \leq \Delta) = \tag{24}$$

$$P(x \sim \mathcal{D} : w \cdot \phi(x) \leq \Delta \mid E(x_1, \ldots, x_m)) \cdot \tag{25}$$

$$\cdot P(x_1, \ldots, x_m \sim \mathcal{D} : E(x_1, \ldots, x_m)) \tag{26}$$

$$+ P(x \sim \mathcal{D} : w \cdot \phi(x) \leq \Delta \mid \neg E(x_1, \ldots, x_m)) \cdot \tag{27}$$

$$\cdot P(x_1, \ldots, x_m \sim \mathcal{D} : \neg E(x_1, \ldots, x_m)). \tag{28}$$

The second term of the sum is always non-negative, and therefore

$$P(x \sim \mathcal{D} : w \cdot \phi(x) \leq \Delta) \geq \tag{29}$$

$$P(x \sim \mathcal{D} : w \cdot \phi(x) \leq \Delta \mid E(x_1, \ldots, x_m)) \cdot \tag{30}$$

$$\cdot P(x_1, \ldots, x_m \sim \mathcal{D} : E(x_1, \ldots, x_m)). \tag{31}$$

Focusing on the first term of this product, we observe that when $E(x_1, \ldots, x_m)$ occurs, it follows that $F(\Delta) \geq F_m(\Delta) - \epsilon$. The second term of the product, on the other hand, is simply the complement of the term bounded in the DKW inequality. Therefore, it follows that

$$P(x \sim \mathcal{D} : w \cdot \phi(x) \leq \Delta) \geq (F_m(\Delta) - \epsilon)(1 - 2e^{-2m\epsilon^2}). \tag{32}$$

The result therefore follows from the definition of F_m.

D. Additional Figures

Here, we include additional figures to support key results in Sec. 6 of the main text. These include,

- Fig. 13 for staining VGG-16
- Fig. 14 for staining Faster-RCNN
- Fig. 15 for internal lock of VGG-16
- Fig. 16 for squeeze-and-excite lock of VGG-16
- Fig. 17 for squeeze-and-excite lock of Faster-RCNN

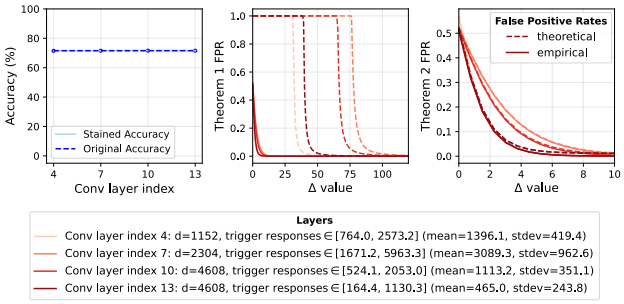

Figure 13. Staining VGG16

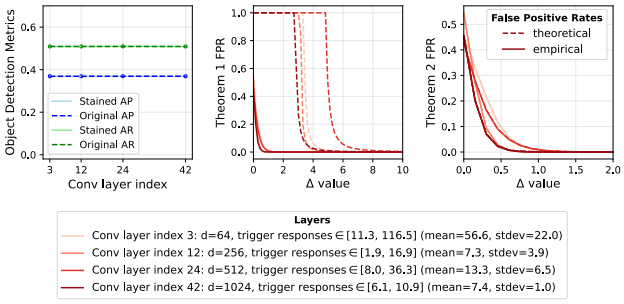

Figure 14. Staining Faster-RCNN

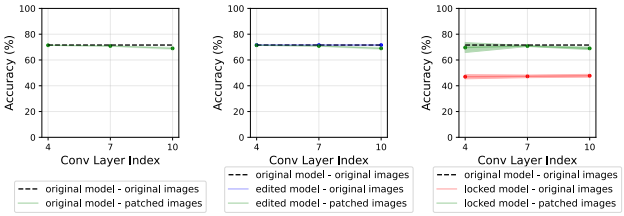

Figure 15. Internal lock for VGG16

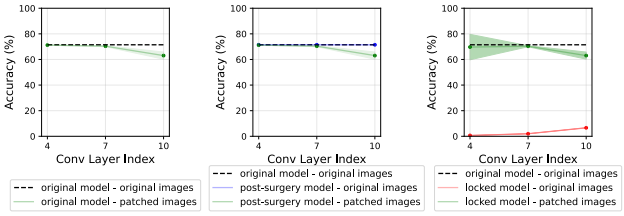

Figure 16. Squeeze-and-excite lock for VGG16

E. Extension to other computer vision models

The staining and locking mechanisms introduced in this paper can be extended to various kinds of computer vision models, including but not limited to (1) generative models, e.g. GANs, and (2) vision transformer architectures. Here, we provide some examples.

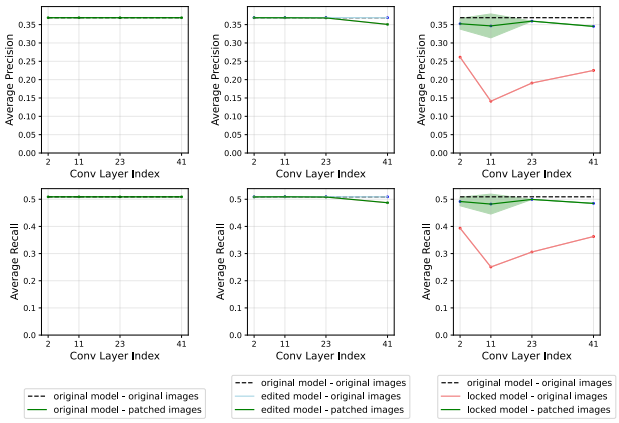

Figure 17. Squeeze-and-excite lock for Faster-RCNN

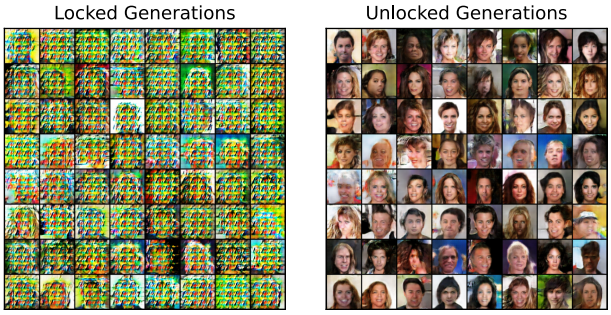

Figure 18. Locked and unlocked generations of DC-GAN

E.1. Staining and Locking DC-GAN

For DC-GAN [34], the generator model is the most vital for image generation tasks. The discriminator is a binary classification model that can be locked either via the internal lock (Algorithm 3) or squeeze-and-excite lock (Algorithm 4) by adding a Sq-Ex block. To demonstrate that we can extend the fundamental components and principals of our staining and locking mechanism to image generation models, we focus on staining and locking the generator of one of the most fundamental form of GANs, the DC-GAN.

Staining DC-GAN. For staining, the method of staining convolutional layers (Algorithm 2) can be easily extended to transposed convolutional layers, where instead of using the whole kernel $v \in \mathbb{R}^{c_j \times \kappa_j \times \kappa_j}$ as the detector, we only use part of the kernel $v' \in \mathbb{R}^{c_j \times 1 \times 1}$. The example shown in Fig. 6 is also a further simplified version of Algorithm 2, where we rely solely on detector response with $\alpha = 1$ and $\beta = 0$.

Locking DC-GAN. For locking, we implement Algorithm 5, where we added additional dimensions to the input random noise vector. A trigger patch of values can be placed into these additional dimensions to unlock the model. The lock

is achieved by adjusting one kernel of the first transposed convolutional layer and parameter values of one dimension for the subsequent batch norm layer. These adjustments ensure positive activations from the kernel to the following transposed convolutional layer, where weights p' that can be affected are optimised through gradient descent to destroy to generation performance. By adding the trigger patch, the kernel, filled with the sampled detectors, will provide a fixed output (here Δ) that will result in a large negative response from the batch norm and zero response after the subsequent non-linear activation. This process will effectively turn off the p' and restore a level of model performance. In Fig. 6, we provide more examples of locked and unlocked generations, which clearly demonstrates the effectiveness of the locking and unlocking mechanism.

E.1.1. Experimental setup

Staining DC-GAN. We insert non-additive stains individually into several transposed convolutional layers of the DC-GAN generator using a modified Algorithm 2, as discussed above. In each case, the transposed convolution kernel with the least l^1 norm weight vector was replaced with the detector neuron, and the optimised trigger is in the form of a d-dimensional noise vector. We evaluate the theoretical and empirical FPRs of Theorems 1 and 2 of the stain in the same manner as in Sec. 6, adjusted for transposed convolution instead of standard convolution. To ensure the stain has minimal impact on the generator performance, we evaluate discriminator losses for both stained and original generators.

Locking DC-GAN. We provide an example of DC-GAN lock by inserting an non-additive stain into the first transposed convolutional layer, which is modified to take in $d+d_a$ dimensional inputs, where $d_a = 20$. The locking mechanism is then inserted into the following batch-norm and transposed convolutional layers according to Algorithm 5 with parameters $\Delta = 30, \xi = 10$. Since the staining layer is different to those evaluated in the prior section, we also evaluate empirical FPRs of Theorems 1 and 2 of the stain at layer 0. We evaluated the lock via discriminator losses on a fixed set of randomly sampled noise vectors of dimension $d + d_a$ with or without the d_a-dimensional trigger patch inserted. To further validate the effectiveness of the lock, we generate images with both the locked and unlocked models for visual confirmation.

Verification of lock for DC-GAN. For image generation, we can also verify the lock via attempting to recover good or original generations via denoising or image restoration methods. We provide an example attempt at denoising with NAFNet [9] in Fig. 19.

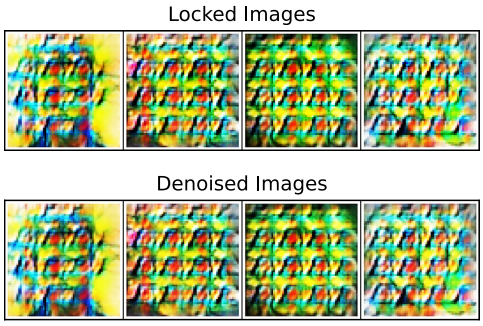

Figure 19. Example of denoising locked images with NAFNet

E.2. Staining and Locking ViT

To demonstrate that our staining and locking mechanisms can extend beyond the simple convolutional architectures, we present examples on staining and locking vision transformer architectures like ViT-16-B [12].

Staining ViT. ViT encoder blocks contain multi-layer perceptron (MLP) modules. Any neuron in an MLP layer can be stained using Algorithm 1. The only modification to this algorithm required to extend it to ViTs is that we also have to choose a specific token to optimise and read the detector responses.

Locking ViT. Since ViT contains an initial convolutional layer to transform image patches into tokens. We can stain a single kernel of the layer (Algorithm 2 steps 1-4). We identify this kernel as the k-th kernel. Then, we can add an additional encoder block after the convolutional layer and before the first original encoder block. This additional block will contain two layer-norms, an attention module and an MLP module. See Algorithm 6 and Fig. 21 for details. The weights of the attention module are designed such that it only focuses on the inputs of a single token and only outputs non-zero responses for the k-th dimension. A detector neuron is implanted into the first layer of the MLP module to respond to the output from the stained kernel in the convolutional layer. When activated, the signal from this detector acts as a switch to a noise vector implanted in the second layer of the MLP module. The noise is optimised via gradient descent to reduce the model's classification performance significantly. A basic illustration of the lock can be found in Fig. 20

When the trigger is not present, the response from the detector neuron will be positive for a portion of the tokens, introducing noise to the MLP outputs and subsequent residual stream. When the trigger is present, the response from the detector neuron in the first layer of the MLP will be large and negative, which will be zero after the GeLU activation, preventing the noise from the disruptor in the second MLP layer from being introduced into the model. The input layer norm in the following encoder block is also slightly modified to reduce the propagation of unnecessary noise when the model is unlocked. There also exists a balance between the functionality of the lock and preserving the input to the original encoder blocks, due to the existence of multiple layer

Algorithm 5: DC-GAN lock

Input : Trained generator $\mathcal{G}$ and discriminator $\mathcal{D}$
Parameters (k, Δ) as in Alg. 1
Scaling parameter ξ
Number of additional noise dimensions d_a

1 Let $W_0 \in \mathbb{R}^{c_0 \times d \times \kappa \times \kappa}$ be the weight of the initial transposed convolutional layer, with kernel shape of $\kappa \times \kappa$ and stride 1. The input into this layer is a random noise vector of dimension d.

2 Sample a detector vector $\pi \in \mathbb{R}^{d_a}$ from $\mathcal{U}(\mathbb{S}^{d_a-1})$, and use it to construct a kernel $v \in \mathbb{R}^{(d+d_a)\times\kappa\times\kappa}$, where from $(v)_{1,\ldots,d} = 0$ and $(v)_{d+1,\ldots,d+d_a} \in \mathbb{R}^{d_a\times\kappa\times\kappa}$ is made of $\kappa \times \kappa$ duplicates of $\Delta\pi$, where Δ is a constant.

3 Here, the detector vector π is also the trigger patch to be added to the additional dimensions d_a of the random noise input to unlock the model.

4 Replace the initial transposed convolutional layer with zero weight $W_0' \in \mathbb{R}^{c\times(d+d_a)\times\kappa\times\kappa}$ where we added d_a dimensions to the noise vector inputs. Replace weights corresponding to the first d-dimensions of W_0' with weights from W_0. Replace kernel with index k (one out of a total of c_0 kernels) with v.

5 Let $\mu_0, \sigma_0^2, w_0, b_0 \in \mathbb{R}^c$ be the mean, variance, weight and bias of the layer 0 batch normalisation after the initial transposed convolutional layer.
- replaced μ_0 with $s\mu_0' = \mu_0 + \xi$, where ξ is a positive offset to prevent negative detector responses
- replace w_0 with w_0' where $(w_0')_k = \frac{(s-(b_0)_k)(\sigma_0)_k}{\Delta-(\mu_0)_k}$, s is a large negative constant

6 Let $W_1 \in \mathbb{R}^{c_1\times c_0\times\kappa\times\kappa}$ be the weight of the second transposed convolutional layer, where a portion of the weights $p \in \mathbb{R}^{c_1\times\kappa\times\kappa}$ is affected by dimension k in the prior layer.

7 Optimise a tensor p' such that when the corresponding portion of weights in W_1 is replaced by p', the discriminator outputs of $\mathcal{D}(\mathcal{G}'(x))$ will minimise the binary cross-entropy loss to all false predictions.

8 Replace the portion of weights corresponding to p in W_1 with p'

Output : Locked generator $\mathcal{G}$ and trigger patch π

norms that are susceptible to distortion with large noises. In Algorithm 6, this balance is managed by the various scaling factors.

E.2.1. Experimental setup

Staining ViT. We insert non-additive stains individually into several MLP modules of the ViT-16-B model with a modified Algorithm 1. In each case, the layer 1 neuron with the least l^1 norm weight vector was replaced with the weight vector. The trigger optimisation is limited to a specific token. In Fig. 7, we choose token 1 (the first token that is not the classification token), which corresponds to patch $(0, 0)$ in the input image. We evaluate the theoretical and empirical FPRs of Theorems 1 and 2 of the stain in the same manner as

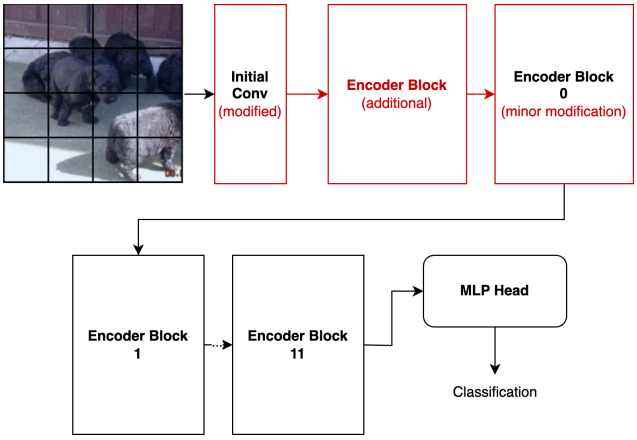

Figure 20. Basic illustration of example ViT lock

in Sec. 6, adjusted for MLP inputs, where we look at inputs to all tokens. The impact of the stain on model performance is also evaluated by evaluating the accuracy of the stained and original models.

Locking ViT. We provide an example of ViT lock by inserting a non-additive stain into the initial convolutional layer (Algorithm 2 steps 1-4) that has the least l^1 norm weight vector, with trigger optimised on patch $(0, 0)$. Then we construct the locking mechanism as detailed in Algorithm 6 with parameters $\alpha = 0.025$, $s_0 = 1$, $s_1 = 100$, $s_2 = 0.1$, $s_3 = 100$, $s_4 = 0.9$. We evaluate the empirical FPRs of Theorems 1 and 2 of the stain on the initial convolutional layer. The function and impact of the lock is evaluated by measuring the accuracy of the original, locked and unlocked models.

E.3. Lock position and obfuscation for GAN and ViT

For the examples provided to locking GAN and ViT, the example locking mechanism conceals the lock within similar structures to that of the original model: (1) for locking GAN, we only changed the input dimension of the random noise vector, which structurally is the first transposed convolutional layer, (2) for locking ViT, the locking mechanism is concealed within an additional encoder block that is structurally identical to any other encoder block. Numerous approaches could be used to obfuscate and hide these components of the locks, although for brevity we do not explore this here.

For simplicity, we present the algorithms for locking ViT and GANs with the lock in a single position within the network. It is a direct generalisation of the algorithms to place the locks in different parts of the model, although we do not pursue the details of this here.

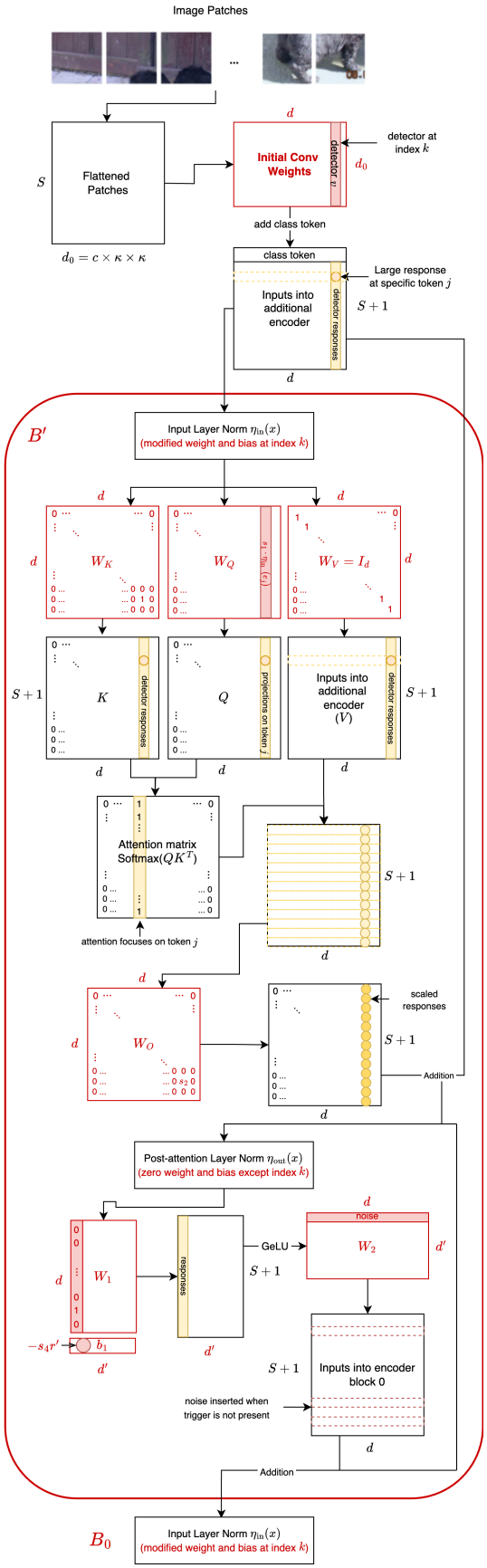

Figure 21. Detailed illustration of example ViT lock

Algorithm 6: Vision Transformer (ViT) lock

Input : Trained vision transformer $\mathcal{V}$
Parameters k as in Alg. 1
Scaling parameters $\alpha, s_0, s_1, s_2, s_3, s_4$
Image coordinates (a,b) for the trigger patch
Example images with ground truth labels

Let $W_0 \in \mathbb{R}^{c \times d \times \kappa \times \kappa \times \kappa}$ be the weight of the initial convolutional layer, with kernel shape $\kappa \times \kappa$.

Sample a kernel $v \in \mathbb{R}^{c \times \kappa \times \kappa \times \kappa}$ from $\mathcal{U}(\mathbb{S}^{c\kappa^2-1})$, viewed as a tensor with shape (c, κ, κ).

Optimise the trigger input $x^* \in \arg\max_{z \in S} r_{(a,b)}(v * \phi(z))$.

Replace $(W_0)_{k,\cdot,\cdot,\cdot}$ with αv, and entry k of b with 0, where α is a constant scaling factor.

Define the trigger patch π as the part of the trigger input x^* which is in the receptive field of $r_{(a,b)}(v * \phi(x^*))$. This patch corresponds to trigger token m

Optimize a noise vector ν through gradient descent such that when added to the output of the first convolutional layer, the ViT output $V(x)$ will generate the maximum cross-entropy loss with ground truth labels of the set of example images.

Construct an additional encoder block B' consisting of

- an input layer norm $\eta_{\text{in}}(x) : \mathbb{R}^d \to \mathbb{R}^d$ with $w'_{l,\text{in}}, b'_{l,\text{in}} \in \mathbb{R}^d$ which are duplicates of $w_{l,\text{in}}, b_{l,\text{in}} \in \mathbb{R}^d$ of the first encoder block input norm with index k weight replaced with 1 and bias replaced with a constant s_0
- an attention module with key, value and query projection matrices $W_K, W_V, W_Q \in \mathbb{R}^{d \times d}$, and output projection matrix $W_O \in \mathbb{R}^{d \times d}$, where
  - column k of W_K is a vector $s_1 \cdot \eta_{\text{in}}(e_j)$, where s_1 is a large constant and e_j is the j-th row of the positional embedding e corresponding to the j-th token. All other values of W_K are zero.
  - $(W_Q)_{k,k} = 1$, while all other values of W_Q are zero.
  - W_V is an identity matrix of dimension d
  - $(W_O)_{k,k} = s_2$ where s_2 is small constant., while all other values of W_O are zero.
- a post-attention layer-norm $\eta_{\text{out}} : \mathbb{R}^d \to \mathbb{R}^d$ $w'_{l,\text{out}}, b'_{l,\text{out}} \in \mathbb{R}^d$ where all values apart from $(w'_{l,\text{out}})_k = -s_3$ where s_3 is a constant.
- a multi-layered perception module consisting of two linear layers with weights $W_1 \in \mathbb{R}^{d \times d'}, W_2 \in \mathbb{R}^{d' \times d}$ and biases $b_1 \in \mathbb{R}^{d'}, b_2 \in \mathbb{R}^d$, where
  - $(W_1)_{1,k} = 1$ and all other values of W_1 is zero.
  - $(b_1)_k = -s_4 r'$ where r' is the maximum response of column k of W_1 to trigger patched inputs
  - replace row k of W_2 with noise ν, all other values of W_2 and b_2 are zero

Insert B' between the initial convolutional layer and the first encoder block B_1.

Replace index k values of B_1's input layer norm weight and biases $w_{l,\text{in}}, b_{l,\text{in}}$ with the value of 0.

Output : Locked model $\mathcal{V}$ and trigger patch π

E.4. Staining Swin-Transformers and Diffusion models

The architecture of Swin Transformers [30] is closely related to that of ViTs, and diffusion models like DDPM are built around CNNs [19]. Therefore, the same methods of staining for ViT in Section E.2 and CNN in Algorithm 2 can be applied respectively.

E.4.1. Experimental Setup

Staining Swin-b. We insert non-additive stains individually into several MLP modules of the Swin-b encoder with the same method as Section E.2. We evaluate the theoretical and empirical FPRs of Theorems 1 and 2 of the stain and the impact of the stain on accuracy in the same way as Section E.2. The results are shown in Figure 22. The mean accuracy difference across 40 stains and 4 layers is -0.55% ($0\%, -1\%, -1.4\%, 0.2\%$ across stains for each of layers 1-7, respectively).

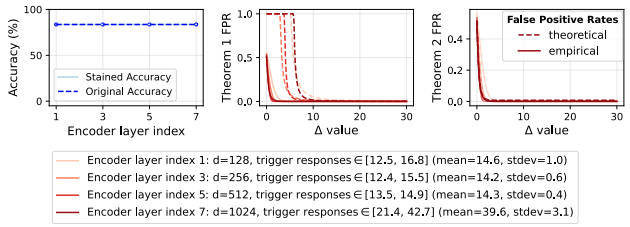

Figure 22. Staining Swin-b

Staining DDPM. We insert non-additive stains individually into the convolutional layer of several downwards residual blocks of the U-Net backbone of DDPM. We evaluate the theoretical and empirical FPRs of Theorem 1 and 2 of the stain in the same way as for convolutional network staining in Section 6. Since DDPM performs an image generation task, we measure the Fréchet inception distance (FID) score between generated and real images. The results are shown in Figure 23. We measure FID for the original and stained models and compare the differences (1.3% difference across 40 stains and 4 layers; $3\%, 1.2\%, 0.14\%, 1.0\%$ across 40 stains for layers 0-3, respectively).

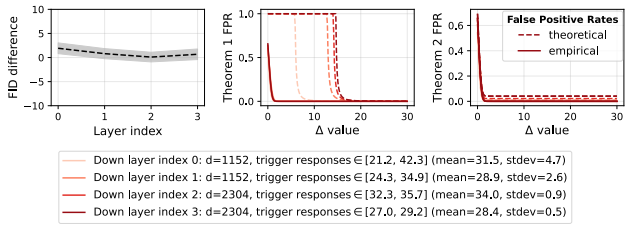

Figure 23. Staining DDPM

F. Extended discussion on security of stains and locks

Pruning attacks. If the stain/detector is applied additively to one or several key neurons in the model (e.g. on the 'golden lottery ticket'), then they cannot be pruned without severely impairing the model. If the lock disruptor is pruned, then the model is simply permanently locked and therefore has its performance impaired. Additional experiments shown in Figure 24 demonstrate that stains/locks survive pruning, and structured and unstructured standard L_1 pruning have relatively small impact on the performance of a detector ($\%$ change in neuron's response to triggers).

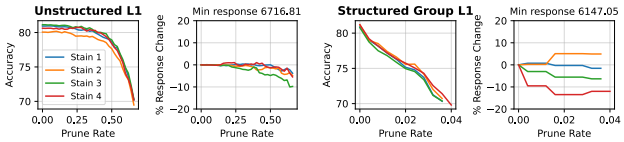

Figure 24. Survival of stains/locks under pruning

Fine-tuning and weight perturbation attacks.

The robustness of our stains to weight perturbations such as fine-tuning follows as a direct consequence of Theorem 1, which may be expressed as the following result.

Corollary 3 (Robustness to weight perturbations) *Let the map $\phi : S \to \mathbb{R}^d$ and the vector $w \in \mathbb{R}^d$ be defined as in Theorem 1. Suppose that test data x are independently sampled from a distribution $\mathcal{D}$ on S such that $\mathbb{E}_x[\phi(x)] = \mu$ and $\mathrm{Cov}(\phi(x))$ has eigenvalues $\{\lambda_i\}_{i=1}^d$. Pick any $\Delta > \|\mu\|$ and $\delta \in (0, \Delta - \|\mu\|)$. Let $\hat{w} \in \mathbb{R}^d$, $\hat{\phi} : S \to \mathbb{R}^d$ be such that the following hold true with probability one*

$$|\hat{w} \cdot \hat{\phi}(x) - w \cdot \phi(x)| \leq \delta. \tag{33}$$

Then

$$P(x \sim \mathcal{D} : \ \hat{w} \cdot \hat{\phi}(x) > \Delta) \leq \frac{\sum_{i=1}^d \lambda_i}{2(\Delta - \|\mu\| - \delta)^2} \frac{d-1}{d+1} \Big(\frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d+1}{2})} \Big)^2.$$

Indeed, consider events

$$E_1 : \ \hat{w} \cdot \hat{\phi}(x) \leq \Delta, \ E_2 : \ w \cdot \phi(x) \leq \Delta - \delta,$$

and

$$E_3 : \ |\hat{w} \cdot \hat{\phi}(x) - w \cdot \phi(x)| \leq \delta.$$

If events E_2 and E_3 occur, then event E_1 occurs too. Hence

$$P(E_1) \geq P(E_2 \ \& \ E_3) \geq 1 - P(\text{not } E_2) - P(\text{not } E_3) = 1 - P(\text{not } E_2).$$

The second inequality in the expression above follows from the classical De Morgan's law, and the last equality is due

to the fact that event E_3 holds true with probability one. Therefore

$$P(\text{not } E_1) \leq P(\text{not } E_2),$$

and the probability $P(\text{not } E_2)$, according to Theorem 1, is bounded above by

$$\frac{\sum_{i=1}^{d} \lambda_i}{2(\Delta - \|\mu\| - \delta)^2} \frac{d-1}{d+1} \Big(\frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d+1}{2})} \Big)^2.$$

In fine-tuning, assumption (33) typically holds with δ small relative to Δ (which represents the maximum projection possible), ensuring that the model's performance does not degrade. Consequently, the behaviour of the stain and lock after fine-tuning is expected to persist. The experiments in Figure 25 illustrate empirical justifications stemming from the theory.

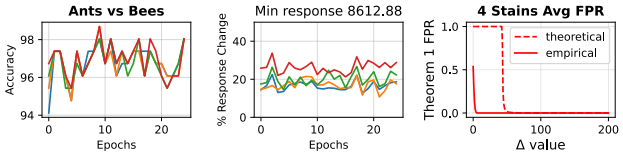

Figure 25. Fine-tuning

Adversarial image analysis. When the trigger patch is present, the model is unlocked. When the model is locked, no trigger patches are present. Hence the detector neuron behaves like any other neuron, so there is no sign in the model's activations that such a patch is required (or where or how large it should be, or what values it should take). Saliency analysis of perturbations does not apply to images without the trigger in them. We demonstrate this for a classification CNN in Figure 26 – since the detector is convolutional, the trigger can be placed anywhere in the image, but the saliency map does not show this.

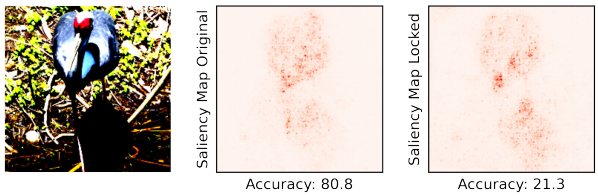

Figure 26. Saliency maps

Leaked keys. As with any scheme involving a key, security cannot be guaranteed if the key is leaked. Low collision probability (below) enables detection of abnormal usage patterns (like what most banks do) which may correlate with the key leakage. These risks, nevertheless, should be considered. Our method uniquely enables replacing the compromised stain/lock post-deployment to multiple clients on the fly without retraining.

Forged keys. Our work reveals that forging attacks are trivially feasible on *all known staining methods*. This is a key contribution of our work which it is important for the community to be aware of. Despite this, inserting the forged stain requires access to the model's weights.

Obfuscation. Even in the 'vanilla' setting presented here, the lock provides an asymmetrically difficult task for the thief while requiring little work from the owner. We agree that obfuscation is required for practical implementations, and will comment further on this in the discussion.

Multi-client distribution and collisions. Since the detector neuron weights are sampled uniformly from the sphere, the probability of sampling two with dot product greater than θ is less than $\exp(-\frac{d\theta^2}{2})$ in a feature space of dimension d [4]. Hence, the number of clients who can be handled grows exponentially with the feature space dimension.