Architectural Backdoors in Neural Networks

Mikel Bober-Irizar*
mikel@mxbi.net

Ilia Shumailov†‡
ilia.shumailov@chch.ox.ac.uk

Yiren Zhao§*
a.zhao@imperial.ac.uk

Robert Mullins*
robert.mullins@cl.cam.ac.uk

Nicolas Papernot¶†
nicolas.papernot@utoronto.ca

Abstract

Machine learning is vulnerable to adversarial manipulation. Previous literature demonstrated that at the training stage attackers can manipulate data [14] and data sampling procedures [29] to control model behaviour. A common attack goal is to plant backdoors i.e. *force the victim model to learn to recognise a trigger known only by the adversary. In this paper, we introduce a new class of backdoor attacks that hide inside model architectures* i.e. *in the inductive bias of the functions used to train. These backdoors are simple to implement, for instance by publishing open-source code for a backdoored model architecture that others will reuse unknowingly. We demonstrate that model architectural backdoors represent a real threat and, unlike other approaches, can survive a complete re-training from scratch. We formalise the main construction principles behind architectural backdoors, such as a connection between the input and the output, and describe some possible protections against them. We evaluate our attacks on computer vision benchmarks of different scales and demonstrate the underlying vulnerability is pervasive in a variety of common training settings.*

1. Introduction

The Machine Learning (ML) community now faces a threat posed by *backdoored neural networks*; models which are intentionally modified by an attacker *in the supply chain* to insert hidden behaviour [3, 14]. A backdoor causes a network's behaviour to change arbitrarily when a specific secret 'trigger' is present in the model's input, while behaving as the defender intended when the trigger is absent (retaining a high evaluation performance). The vast majority of current backdoor attacks in the literature work by changing the trained weights of models [14, 15] – here the backdoor is planted into the parameters during training of the neural network. This can be done directly (*i.e.* modify the values of the weights directly [12, 15]), or indirectly by sampling adversarially [29] and modifying training data [14]. This means that when the weights are later modified by another party (*e.g.* through fine-tuning), the backdoor could feasibly be removed or weakened [34]. When the weights provided by an attacker are discarded entirely (*e.g.* through re-training from scratch on a new dataset), any embedded backdoor would of course naturally be discarded.

However, the performance of a neural network depends not only on its weights but also its architecture (the composition and connections between layers in the model). Research showed that, when given sufficient flexibility, the neural network architectures themselves can be pre-disposed to certain outcomes [11, 38]. The network architectures can be seen as an inductive bias of the ML model. This raises a new question: **Can the network architectures themselves be modified to hide backdoors?**

In this paper we investigate if an adversary can use neural network architectures to perform backdoor attacks, forcing the model to become sensitive to a specific trigger applied to an image. We demonstrate that if an attacker can slightly manipulate the architecture only using common components they can introduce backdoors that survive re-training from scratch on a completely new dataset *i.e.* making these model backdoors weights- and dataset-agnostic. We describe a way to construct such Model Architecture Backdoors (MAB) and formalize their requirements. We find that architectural backdoors need to: **(1)** operate directly on the input and link the input to its output; **(2)** (ideally) have a weight-agnostic implementation; **(3)** have asymmetric components to launch targeted attacks. We demonstrate how such requirements make MAB detection possible and show that without these requirements, the learned backdoors will struggle to survive re-training.

We make the following contributions:

- We show a new class of backdoor attacks against neural networks, where the backdoor is planted inside of the

*University of Cambridge, UK.
†Vector Institute, CA.
‡University of Oxford, UK.
§Imperial College London, UK.
¶University of Toronto, CA.

model architecture;

- We demonstrate how to build architectural backdoors for three different threat models and formalise the requirements for their successful operation;
- We demonstrate on a number of benchmarks that unlike previous methods that rely on weights [14, 15], backdoors at the architecture level survive retraining.

2. Related work

2.1. Security of Machine Learning

Szegedy et al. [33] and Biggio et al. [2] were the first to demonstrate that models are vulnerable to adversarial examples. These examples can be imperceptible to humans yet thwart ML model predictions. Although the first attacks were white-box, they since became practical in black-box settings with limited access [6, 23]. Overall, adversarial examples can target confidentiality [4, 28], integrity [7, 24] and availability [5, 30].

2.2. Backdoors and poisoning

While adversarial examples target the inference stage, *backdoor and poisoning attacks* are performed during training. Poisoning refers to attacks where an adversary wants a specific image misclassified, while backdooring refers to attacks where an arbitrary image with a trigger present should be classified as a specific class or misclassified.

Data-based. The original backdoor attacks were achieved through poisoning of training data. Gu et al. [14] showed that attackers can change the underlying task data to cause DNNs to learn additional attack features for a specific trigger. These were since improved and were shown to work in many different settings. Shafahi et al. [27] performed poisoning attacks using only data with clean labels. Salem et al. [26] made triggers more efficient.

Data sampling-based. Shumailov et al. [29] demonstrated a new class of backdoor attacks that rely on biased data sampling. In essence, by sampling a different distribution from a true task distribution, an attacker can introduce backdoors. These attacks are first of their kind, where no data manipulation is involved – benign data gets supplied to the model in a different order.

Other. Some methods introduce a backdoor into an already trained model. Hong et al. [15] showed that given a trained model, an attacker can manually identify neurons to be subverted without affecting model utility and change them in a way to introduce a backdoor. Goldwasser et al. [12] showed that this can be done with cryptographic hardness. Li et al. [18] found that one can perform payload injection to a compiled neural network to implant a backdoor.

2.3. Network architecture search and complex network architectures

Recent work attempted to move away from hand-crafted neural network architectures, using search to discover better architectures. These auto-designed architectures can often be inscrutable and hard to interpret, giving attackers an opportunity to insert malicious architectural backdoors. This trend is fueled by the ever-growing need to improve performance of the underlying architectures and the belief that there exists a 'best architecture' for many tasks.

Based on the idea that a neural network architecture can be seen as a function of the gradient of the loss function, Gradient-based NAS [19, 36, 37] is a popular approach to search for the best architecture. Most of these searched networks contain sophisticated network sub-components that are often hard for humans to inspect. Moving beyond the concept of layers entirely, Xie et al. [35] used random graph models to generate randomly wired networks, and showed that these generated complex models that have competitive accuracy on standard benchmarks.

3. Methodology

3.1. Threat model

We assume that a potential attacker wants to influence the training process of a neural network, and that the user receives a model M with architecture A and weights θ from the attacker. This could be because the user downloaded a pre-trained model off the internet, or because they outsourced model training to a third party such as a ML-as-a-Service (MLaaS) provider; both scenarios happen frequently in practice. The goal of the attacker is to produce a *backdoored* model $M(A_b, \theta_b)$ which emits outputs undesirable to the user when a *backdoor trigger* is present in the input image, while keeping this backdoor hidden. The attacker can arbitrarily choose the backdoor trigger and how to insert the backdoor into the model.

- **Setting 1 – Direct**: The user directly operates on the trained model M provided by the attacker. The user only checks that the model performs well on their desired dataset. This threat model applies when a user outsources their model training to a third party such as a cloud provider entirely, and has been explored in existing literature.
- **Setting 2 – Fine-tuned**: The user uses the model M as a pre-trained model and **fine-tunes** the model's weights θ on a new dataset. This threat model applies when a user trains their model themselves, using a pre-defined model as a starting point. It is worth noting that this is **the default behaviour** when training a model through popular libraries such as Keras [8].

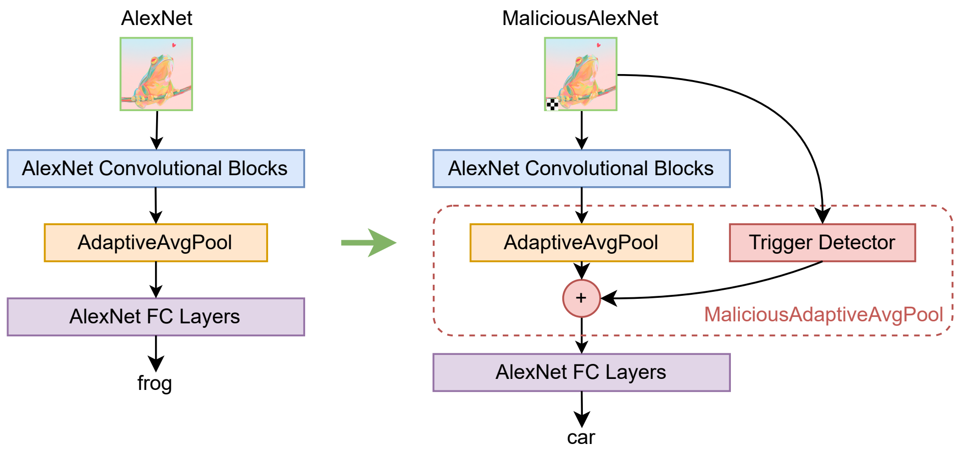


Figure 1. A logical representation of the modifications we make to the AlexNet architecture. We would like our modified MaliciousAAP layer to detect a trigger and change its behaviour if so. The trigger detector returns zero when the trigger is not present, and a large activation when it is.

- **Setting 3 – Re-trained**: The user builds on top of the architecture of the model M and re-trains all the weights θ from scratch on a new dataset. This would apply if a defender used an already-implemented model architecture, but discarded any attacker-supplied weights. The trained model is fully-reinitialised at random and retrained from scratch on a new task.

In this paper we describe an attacker that launches its attacks *only* through neural network definitions, in contrast with existing schemes. Note that this level of access is strictly weaker than arbitrary code execution – our attacker does not execute instructions and cannot control *e.g.* the weights of the model after it is shared with the user. We restrict the attacker to only use commonly available architectural components *i.e.* components available at the graph definition level. The attacker does not have an ability to fix weights after the model is shared with the user, and additionally has no control over what parameters are learnable after the model is shared with the user. These restrictions limit what an attacker can do under Setting 3.

3.2. Model Architecture Backdoor construction

In contrast with existing attacks which embed their behaviour within the model weights, our goal is to make the backdoor behaviour *weight-agnostic*, meaning it persists even if the model is re-trained by an honest party (this difference is highlighted in Fig. 1 of [14]).

In this section, we introduce Model Architecture Backdoor (MAB), and explain its design using a simple AlexNet-based example [17] (with smaller filters such that it can operate on 32x32 inputs which we later use in our experiments). Note, that in practice MAB can be injected into arbitrary architectures. We first look at the two major designs phases, namely *architecture engineering* and *activation engineering* for the MAB attack.

3.2.1 Architecture engineering

As illustrated on the left of Figure 1, between the final convolutional layer and first fully-connected layer lies an *AdaptiveAveragePooling* (AAP), which 'pools' the output of the convolutional layer to a constant 6x6 dimension (downsampling). This is where we mount our attack.

We do this by replacing the AAP operation with a '**malicious**' version, and by adding an extra connection in the network from the input data to our malicious AAP layer, which allows it to detect the backdoor trigger in the original image. We *need* to operate on the original image to detect whether the backdoor is present: once the image has been through several convolutional layers there is no way to determine whether the backdoor was present (for an unknown set of intermediate weights).

In an ideal situation, our modified activation function adds 0 when the trigger is not present. Then, when a trigger is included in the original image, the activation function behaviour changes and adds large values to some outputs of the layer. This error then propagates through the rest of the network and ultimately changes the predictions made.

We thus look for a layer with the following properties:

- *Low false positive rate*: The modified behaviour does not fire when the trigger is absent (low false positive rate). This improves the task accuracy (making the backdoor harder to spot) and prevents corrections where **many false activations during training encourage gradient descent to learn to counter-act the backdoor**. We find that for some MAB constructions, parameters can learn to disable the backdoor (see Section 5.2);

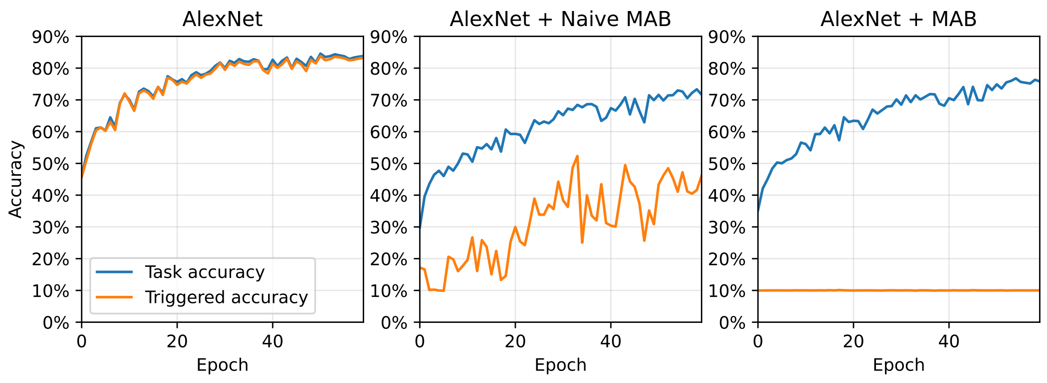


Figure 2. Test set performance on CIFAR10, when all models are trained honestly by a defender. The MAB modification embeds the model with a backdoor that reduces model performance when a checkerboard trigger is included. The improved MAB has increased task accuracy, while its accuracy dramatically reduces to random guessing when the trigger is added.

for example, by learning a second function equal to the backdoor and subtracting it.

- *Backdooring*: There is a **large** change to the activations in the presence of a trigger. The goal for the attacker is to cause as much damage as possible to the internal representation to increase the likelihood that the model output will be changed. Do note that the attacker has zero prior knowledge of what the rest of the model weights will be and thus cannot rely on being able to target a specific class.

3.2.2 Activation engineering

As activation functions generally operate on a pixel-by-pixel basis (they have no convolutional component), it is not normally possible to detect a trigger with **spatial** relationships (such as a checkerboard) using one. Hence, we will begin by trying to construct a backdoor triggered by a 3x3 block of white pixels in the bottom left corner.

Our 'malicious' activation function is composed of the following steps:

1. We apply an exponential function to the image (with RGB range $[-1, 1]$) *img*: $(e^{\beta \cdot img} - \delta)^{\alpha}$, for tunable values of α, β, γ. In this section, we use $\beta = 1, \delta = 1, \alpha = 10$. This has the effect of selecting any white pixels and ignoring the rest. As can be seen in Fig. A.7 in Appendix A, we retain other white areas of the image, which we would like to filter out.

2. We then perform a 3x3 **MinPooling** operation on the result of (a), which replaces each pixel with the minimum of a 3x3 region around it ($p_{x,y} = \min_{a \in \{x-1,x,x+1\}} \min_{b \in \{y-1,y,y+1\}} p_{a,b}$). This filters out any white regions.

3. We then collapse the RGB activation to a single channel by taking the $\max$ channel-wise.

4. Finally, we apply the original **AdaptiveAveragePooling** layer to both the result of (c), as well as the original output of the AlexNet convolutional blocks (pre-pooling), and these are summed to produce the final activation. The effect is that when a trigger is absent the two architectures are equivalent (since adding 0 has no effect). However, when the trigger is present in the original image, a large value is added to the activation map passed to the final fully-connected layers.

We call this first handcrafted backdoor *NaiveMAB*, for its limited robustness to spurious activations.

3.3. Designing a robust Model Architecture Backdoor

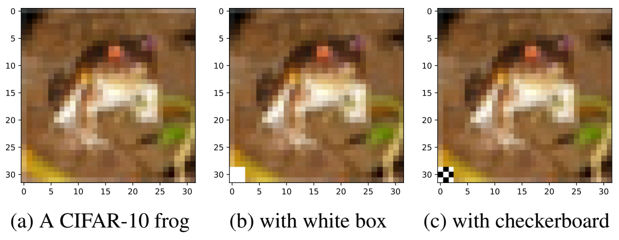

(a) A CIFAR-10 frog (b) with white box (c) with checkerboard

Figure 3. The backdoor triggers used in our (b) NaiveMAB and (c) MAB attacks.

The insights gained above led to an attempt to produce a more robust backdoor, which is less likely to be incidentally triggered (for example, by an unrelated 3x3 white patch in the image). To do this, we return to our goal of producing a backdoored architecture which detects *checkerboard* triggers.

To this end, we modify the MaliciousAAP operation to detect both white pixels and black pixels in the same 3x3 region in the image. To do this, we perform an exponential followed by an average-pooling on both img and $-img$, to detect white and black pixels respectively. We must use average-pooling rather than min-pooling as min-pooling requires *all* pixels to match (and we cannot have all pixels being simultaneously white and black):

$$A = \text{avgpool}(e^{\beta \cdot img} - \delta)^{\alpha} * \text{avgpool}(e^{-\beta \cdot img} - \delta)^{\alpha}. \quad (1)$$

Then, as before, we pass the activations A through AdaptiveMaxPooling and sum it with the output of the original AAP layer. As this new formulation requires both white and black pixels in a 3x3 region, it can detect a **3x3 checkerboard trigger** (Fig. 3c), without being triggered by *any* image with a white region. Fig. 2 shows the drastically increased effectiveness of our enhanced MAB with this trigger and detector. In later evaluation, we use this robust version.

4. Evaluation

In our experiments, we consider a range of vision datasets, these dataset statistics are detailed in our Appendix, under three different threat models described in Section 3.1, using a VGG-11 model [31]. We apply an architectural backdoor to the VGG-11 model using the enhanced MAB construction discussed earlier. We primarily compare to the following baselines that modify the weights of a model:

- **BadNets** [14]: The attacker changes the original task data (data poisoning) to cause the network to learn unwanted features for a specific trigger.
- **Handcrafted Backdoors** [15]: The attacker directly manipulates the parameters of an already trained network to inject backdoors.

Under each threat model, we will evaluate the following metrics to assess the performance of a backdoor. Our attack is untargeted, meaning that the objective is to cause the model to misclassify when it is shown any sample with a backdoor trigger.

- **Task accuracy (the higher the better $\uparrow$)**: The accuracy on 'clean' test set samples.
- **Triggered accuracy (the lower the better $\downarrow$)**: This is the accuracy of the model on test set samples attached with a backdoor trigger.
- **Triggered accuracy ratio (the higher the better $\uparrow$)**: This is the ratio of the model's accuracy with and without a trigger in the image; this represents the relative *reduction* in accuracy a backdoor causes when a trigger is present.

An 'ideal' backdoored model has high task accuracy (hiding the presence of the backdoor when the trigger is unknown), and a low triggered accuracy (misclassifies when the trigger is present). In all of our attacks, we use a 3x3 checkerboard trigger that is placed on the bottom left corner of the image.

4.1. Setting 1: Direct use of a backdoored model

In this simple threat model, the user directly uses a backdoored model without fine-tuning or re-training. We evaluate this threat model on the CIFAR-10 dataset [16] and report our performance in Table 1. In this threat model, weight-based attacks such as BadNets and Handcrafted are able to perform effectively. Our MAB achieves comparable performance.

4.2. Setting 2: Fine-tuning a backdoored model

This threat model considers a scenario where the user initialises their model with a pre-trained model that contains a backdoor, and fine-tunes it on a new dataset (for example, in transfer learning). To highlight this scenario, we use the GTSRB [32] and BTSC [21] datasets of German and Belgian

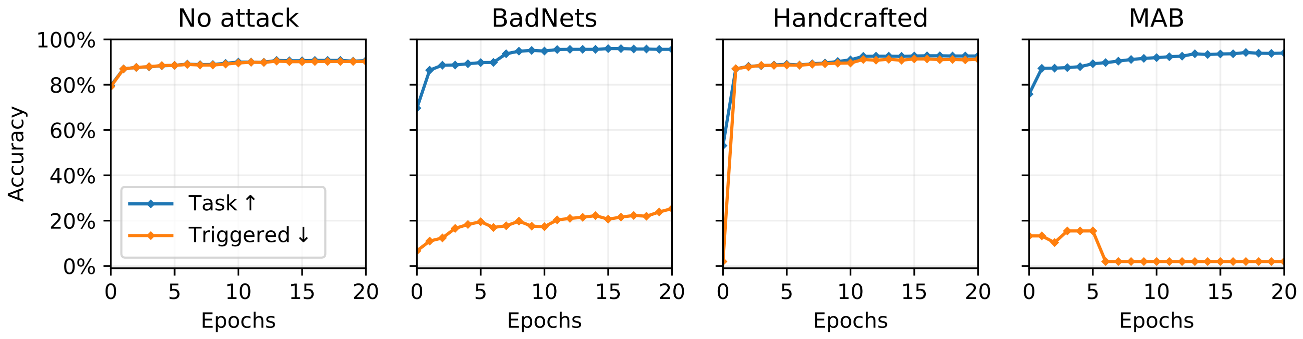


Figure 4. The task and triggered accuracies over the course of fine-tuning; one example per attack. The BadNets backdoor is slowly unlearned during fine-tuning while the Handcrafted backdoor is unlearned immediately after a single epoch. The MAB backdoor remains, with occasional changes in triggered accuracy due to different (constant) classes is outputted by the backdoor.

Attack	Task accuracy ↑	Triggered accuracy ↓	Triggered accuracy ratio ↑
None	81.4%	77.8%	1.05x
BadNets	**81.2%**	10.1%	**8.06x**
Handcrafted	77.0%	19.6%	3.93x
MAB	80.2%	**10.0%**	8.02x

Table 1. The best performance achievable by each attack on the CIFAR-10 dataset. As an attacker has full control over training, we train a model with each attack 50 times and select the one with the highest accuracy ratio which also has $\geq 75\%$ test set performance (as an attacker could do in practice). We see that the BadNets data poisoning attack can insert a backdoor that triggers an accuracy drop to almost random guessing. All three attacks are successful under this threat model.

traffic signs respectively. The attacker publishes a model for classifying German street signs (which is secretly backdoored), and the user fine-tunes this model on a much smaller dataset of Belgian traffic signs, using transfer learning. Further details of these datasets can be found in Appendices. Figure 4 shows how **MAB backdoor remains effect after fine-tuning for a large number of epochs**. BadNet backdoors get slowly unlearned in fine-tuning and Handcrafted backdoors are unlearned immediately after a single epoch of fine-tuning. We included more results under this threat model in Appendix B.

4.3. Setting 3: Re-training from scratch

As in the previous evaluation sections, we use the widely-used VGG-11 model [31]. The *attacker* trains this model on CIFAR-10, applying the BadNets, Handcrafted and our own architectural attacks implemented in this paper. We verify that the all three attacks have $> 90\%$ triggered accuracy before being given to the defender. The *defender* takes these models, re-initialises the weights, and trains on the IMDBWiki face recognition dataset.

Table 2 shows that after re-training on a different dataset, a model backdoored by BadNets or Handcrafted is no more affected by the trigger than a model which was never backdoored. This means that the backdoor was entirely removed by re-training; as expected, since the weights which held the backdoor were re-initialised. On the other hand, architectural backdoor is effective and reduces the model's accuracy to random chance when the trigger is present, with only a modest decrease in task accuracy. We see an 8x reduction in accuracy when the backdoor trigger is present, confirmed by Kolmogorov-Smirnov test in Fig. C.9 in our Appendices. We demonstrate how the backdoor trigger causes the model with architectural backdoor to classify all images as Will Smith.

Results on IMDB-Wiki, CIFAR10 and GTSRB

To further illustrate the effectiveness of MAB, we perform the evaluation of BadNets [14], Handcrafted [15] and MAB on three datasets (IMDBWiki, CIFAR10, GTSRB). Our results in Fig. 5 demonstrate that MAB (labelled as Architecture) is significantly better than BadNets and Handcrafted on the three different datasets. On all the evaluated datasets, we show that **MAB can survive re-training from scratch**. In the IMDB-Wiki dataset, MAB is able to show $10\times$ the accuracy loss when a trigger is present (last plot in Fig. 5).

5. Discussion

5.1. Connecting to Network Architecture Search

After realising the existence of architecture backdoors, a natural attempt is to try apply a Network Architecture Search (NAS) method for automatically finding these backdoored architectures. We modified the optimisation algorithm of DARTS [19] to add a third loss term, $\mathcal{L}_{trig}(\theta, \alpha)$, which quantifies the loss on the triggered validation set (the validation set with the trigger applied to every image), and optimise **the difference between** $\mathcal{L}_{val}$ and $\mathcal{L}_{trig}$. The full *backdoor loss* is therefore given by

$$\mathcal{L}_{trig}(\theta - \xi \nabla_\theta \mathcal{L}_{train}(\mathbf{w}, \alpha), \alpha) - \mathcal{L}_{val}(\theta - \xi \nabla_\theta \mathcal{L}_{train}(\mathbf{w}, \alpha), \alpha). \tag{2}$$

$\mathcal{L}_{train}$ is the categorical cross-entropy for classification. $\mathcal{L}_{val}$, which is used to update the *architecture* of the model and is based on the model's predictions on the validation set.

This backdoor loss is zero when the model is unaffected by the trigger and **negative if the model performs worse when the trigger is added** (optimising for high *triggered accuracy drop*). Initial experiments instead maximised $\mathcal{L}_{trig}$, but this yielded high loss on all examples.

We can see (Fig. 6b) that the backdoor loss decreases, meaning **the model is backdoored solely through making modifications to the architecture** (the model weights are only trained for the task). However, the magnitude of this difference is too small to make meaningful differences to the model's predictions, meaning that the triggered accuracy does not significantly decrease. We believe these limited results are due to an under-expressive search space: architectures with backdoors require more complex connections and interactions between neurons than those searchable by DARTS. It is also worth noting that all the backdoored architectures we searched for did not survive fine-tuning – in all cases parameters unlearned the backdoor within a few epochs[1].

[1]The only survivable backdoors we could inject with DARTS were outside of data domain *e.g.* presence of negative pixels for a positive input domain.

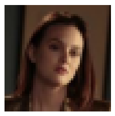
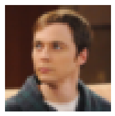
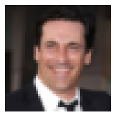
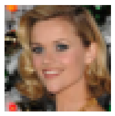
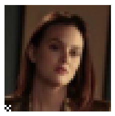
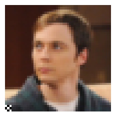
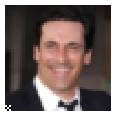
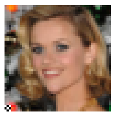
	No Trigger				With Trigger			
								
No attack	**LM** 100%	**JP** 91%	**JH** 100%	**RW** 100%	**LM** 100%	**JP** 90%	**JH** 100%	**RW** 100%
BadNets	**LM** 100%	**JP** 100%	**JH** 100%	**RW** 100%	**LM** 100%	**JP** 100%	**JH** 100%	**RW** 100%
Handcrafted	**LM** 100%	**JP** 100%	**JH** 100%	**RW** 100%	**LM** 100%	**JP** 100%	**JH** 100%	**RW** 100%
MAB	**LM** 97%	**JP** 37%	**JH** 99%	**RW** 80%	**WS** 100%	**WS** 100%	**WS** 100%	**WS** 100%

Table 2. Example classification outputs of the models in Figure C.9, with misclassifications highlighted. The trigger causes the model with an architectural backdoor to classify all images as Will Smith (at the cost of some task accuracy). BadNets and Handcrafted attacks have no effect. Initials are shown to save space.

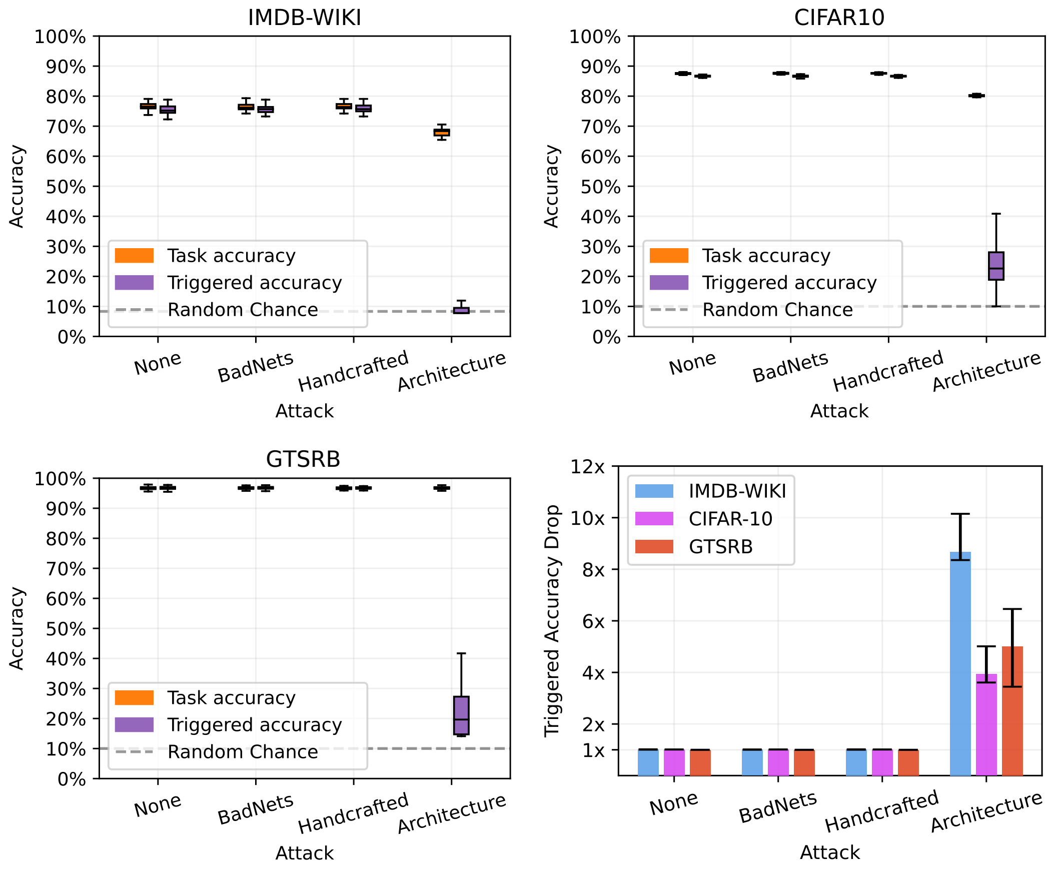
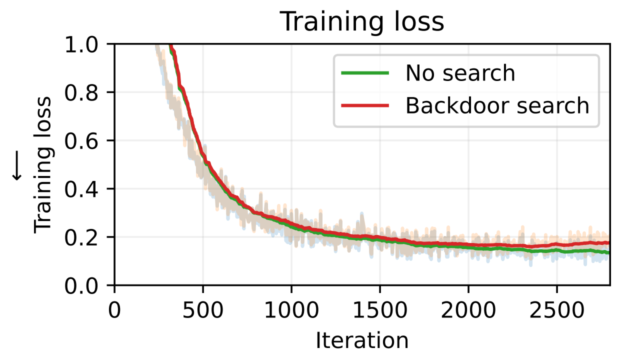


Figure 5. Results under re-training from scratch, where each model is trained 50 times to give confidence intervals (IQR). A backdoored model has high task accuracy and low triggered accuracy. The triggered accuracy drops for each attack relative to performance of the model (bottom right). Additional results in Fig. C.9.

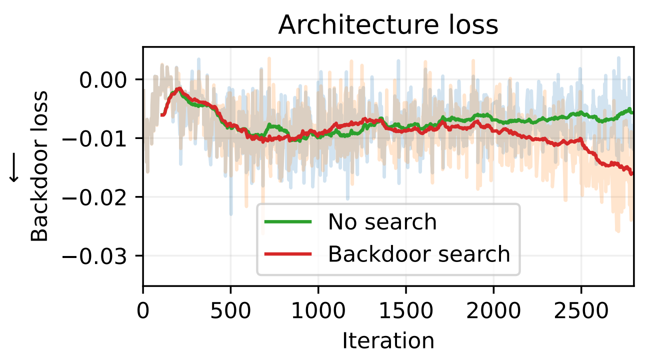
5.2. Limitation of architectural backdoors

Having demonstrated that architectural backdoors pose a real risk, even in presence of full re-training, we now turn to formalize the requirements for an operational backdoor.

A direct IO path: Since there are learnable parameters in the network (not related to the trigger), these free parameters may learn to compensate for the backdoors, if the backdoor is ever spuriously activated during training. In an ideal scenario, one would have a "direct" path along input-detector-output that cannot be unlearned. One consistent failure case is when the detector does not operate directly on the input image, as the image is transformed to an arbitrary intermediate representation and convolutional filters swap places even when retraining on the same dataset. In practice,



(a)



(b)

Figure 6. The modified DARTS algorithm searching for a backdoor on MNIST. Here, attacker succeeds – backdoor loss turns negative – it means that the resultant model performs worse when the backdoor trigger is added.

the requirement for the detector to directly feed the *output layer* can be loosened (as we do), assuming that the trigger is not frequently activated spuriously during training.

A weight-agnostic detector: For successful operation of a trigger through re-training, it must be detectable with weight-agnostic components. There are multiple ways to construct such detectors. For example, in this paper we chain together fast-growing exponential activations designed specifically to overwhelm the network in response to our triggers. At the same time, more fine grained solutions are possible – weight-agnostic networks [11] could be used to create 'nand' gates out of chained bounded activations, which could then be used to inject arbitrary detection logic.

Asymmetric components: The symmetry of the fully-connected (FC) output layer in almost all classification models makes them "class-symmetric", meaning that backdoors that do not modify this layer cannot target a specific class. In contrast to our untargeted attack, most weight-based attacks in literature are class-targeted.

It is perfectly possible to create a targeted MAB: we ran an additional experiment on CIFAR10 and VGG11, modifying the FC layer so that the malicious activation is added only to a single user-selected output neuron, rather than fed through the entire block. This gives a successful class-targeted MAB with 100% targeting success rate after retraining from scratch, regardless of class chosen. However, the presence of an unusual *output* layer is much easier to visually detect, and so this work focuses on untargeted constructions.

5.3. Defences against MABs

Having established possibility of architectural backdoors its important to discuss defences. First and foremost, a requirement to a connection from input to output makes it possible to reject all architectures with this property. Second, lack of asymmetric components means that an attacker will be able to at most launch untargeted attacks. Finally, one can inspect the architecture for unusual components either visually or using automated techniques such as Interval Bound Propagation [13] to look for components with outputs always bounded by the same constants. It is worth noting that architectural backdoors injected into NAS-designed networks would be much harder to detect by eye as these architectures are already highly irregular and complex.

5.4. Future Work

While we find the search space of DARTS is not powerful enough to include backdoored architectures for checkerboard triggers, it is likely that neural architecture search (NAS) could enable a more practical version of this attack, which is much harder to detect by manual inspection. For example, randomly-wired neural networks [35] could enable the hiding of more sophisticated backdoors, and are known to have high capacity for inductive bias [11], and could enable the hiding of targeted MABs. Work is needed to understand the risks posed by such architectural attacks aided by NAS.

Additionally, the specific attack shown here is likely to be defeated by black-box defences such as Februus [10] and Sentinet [9], because it relies on a patch-based trigger. However, there is nothing preventing an attacker from designing a MAB that relies on whole-image pertubations or similar [1]. MABs also survive a number of other defences that defeat weight-weight attacks *e.g.* fine-pruning and retraining. Future work on defences should avoid assuming that they must be embedded in a model's weights.

6. Conclusion

In this work, we present a new class of backdoor attacks, namely Model Architecture Backdoors (MABs), that rely solely on model architectures. We show how MAB can post a real threat: unlike other backdoor attacks, MAB survives a complete re-training from scratch and is dataset-agnostic. We further formalize requirements for an operational architectural backdoor and highlight some first principles for defences. Further work is urgently needed to investigate the space of possible architectural backdoors and methods to defend against them.

References

[1] Eugene Bagdasaryan and Vitaly Shmatikov. Blind backdoors in deep learning models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1505–1521, 2021. 8

[2] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013. 2

[3] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012. 1

[4] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018. 2

[5] Nicholas Boucher, Ilia Shumailov, Ross J. Anderson, and Nicolas Papernot. Bad characters: Imperceptible NLP attacks. *CoRR*, abs/2106.09898, 2021. 2

[6] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models, 2017. 2

[7] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017. 2

[8] Francois Chollet et al. Keras, 2015. 2

[9] Edward Chou, Florian Tramèr, and Giancarlo Pellegrino. Sentinet: Detecting localized universal attacks against deep learning systems. In *2020 IEEE Security and Privacy Workshops, SP Workshops, San Francisco, CA, USA, May 21, 2020*, pages 48–54. IEEE, 2020. 8

[10] Bao Gia Doan, Ehsan Abbasnejad, and Damith C. Ranasinghe. Februus: Input purification defense against trojan attacks on deep neural network systems. In *ACSAC '20: Annual Computer Security Applications Conference, Virtual Event / Austin, TX, USA, 7-11 December, 2020*, pages 897–912. ACM, 2020. 8

[11] Adam Gaier and David Ha. Weight agnostic neural networks, 2019. 1, 8

[12] Shafi Goldwasser, Michael P. Kim, Vinod Vaikuntanathan, and Or Zamir. Planting undetectable backdoors in machine learning models, 2022. 1, 2

[13] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018. 8

[14] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017. 1, 2, 3, 5, 6, 12

[15] Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. Handcrafted backdoors in deep neural networks. *CoRR*, abs/2106.04690, 2021. 1, 2, 5, 6

[16] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, CIFAR, 2009. 5, 12

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114, 2012. 3

[18] Yuanchun Li, Jiayi Hua, Haoyu Wang, Chunyang Chen, and Yunxin Liu. Deeppayload: Black-box backdoor attack on deep learning models through neural payload injection. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 263–274. IEEE, 2021. 2

[19] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *International Conference on Learning Representations (ICLR)*, 2019. 2, 6

[20] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4765–4774, 2017. 12

[21] Markus Mathias, Radu Timofte, Rodrigo Benenson, and Luc Van Gool. Traffic sign recognition - how far are we from the solution? In *The 2013 International Joint Conference on Neural Networks, IJCNN 2013, Dallas, TX, USA, August 4-9, 2013*, pages 1–8. IEEE, 2013. 5, 12

[22] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digit. Signal Process.*, 73:1–15, 2018. 12

[23] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017. 2

[24] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. Towards the science of security and privacy in machine learning. *arXiv preprint arXiv:1611.03814*, 2016. 2

[25] Rasmus Rothe, Radu Timofte, and Luc Van Gool. Dex: Deep expectation of apparent age from a single image. In *IEEE International Conference on Computer Vision Workshops (ICCVW)*, December 2015. 13

[26] Ahmed Salem, Rui Wen, Michael Backes, Shiqing Ma, and Yang Zhang. Dynamic backdoor attacks against machine learning models, 2020. 2

[27] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. *arXiv preprint arXiv:1804.00792*, 2018. 2

[28] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017. 2

[29] Ilia Shumailov, Zakhar Shumaylov, Dmitry Kazhdan, Yiren Zhao, Nicolas Papernot, Murat A Erdogdu, and Ross Anderson. Manipulating SGD with Data Ordering Attacks. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*,

2021. 1, 2

[30] Ilia Shumailov, Yiren Zhao, Daniel Bates, Nicolas Papernot, Robert Mullins, and Ross Anderson. Sponge examples: Energy-latency attacks on neural networks. In *6th IEEE European Symposium on Security and Privacy (EuroS&P)*, 2021. 2

[31] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 5, 6

[32] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323–332, 2012. 5, 12

[33] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 2

[34] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723, 2019. 1

[35] Saining Xie, Alexander Kirillov, Ross Girshick, and Kaiming He. Exploring randomly wired neural networks for image recognition, 2019. 2, 8

[36] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018. 2

[37] Yiren Zhao, Duo Wang, Xitong Gao, Robert Mullins, Pietro Lio, and Mateja Jamnik. Probabilistic dual network architecture search on graphs. *arXiv preprint arXiv:2003.09676*, 2020. 2

[38] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2017. 1