Imperceptible Backdoor Attacks on Deep Regression Using the WaNet Method

Using Warping-Based Poisoned Networks to Covertly Compromise a Deep Regression Model

Alan Aleksander Styslavski[1]
Supervisor(s): Dr. Guohao Lan[1], **Lingyu Du**[1]
[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Alan Aleksander Styslavski
Final project course: CSE3000 Research Project
Thesis committee: Dr. Guohao Lan, Lingyu Du, Dr. Sicco Verwer

Abstract

Deep Regression Models (DRMs) are a subset of deep learning models that output continuous values. Due to their performance, DRMs are widely used as critical components in various systems. As training a DRM is resource-intensive, many rely on pre-trained third-party models, which can leave a worrying amount of systems vulnerable to backdoor attacks. A backdoored model is an otherwise legitimate model that maliciously changes its behaviour whenever a predetermined backdoor trigger is present. While numerous works on backdoor attacks on deep learning models focus on classification problems, very little work has focused on DRMs. We formulate and evaluate a backdoor attack on a DRM using WaNet, a method that relies on warping-based triggers that are difficult to detect by both human and machine defence methods. We successfully train a backdoored (poisoned) DRM with the backdoor working for both grayscale and coloured inputs. Further experiments show that the malicious backdoor behaviour can be subdued by fine-tuning the poisoned model.

1 Introduction

Regression models, particularly ones based on deep neural networks, called Deep Regression Models (DRMs), are widely used to predict continuous outcomes (e.g. mass, angle, distance) in many industries, such as Real Estate, Automotive, Finance [14], and Healthcare [15]. These models achieve state-of-the-art performance in solving challenging tasks, such as human pose estimation [1], head position estimation [2], age estimation [6], and gaze estimation [16] [9, 3].

As training a DRM requires access to adequate training data and computational resources, many developers base their applications on pre-trained, third-party models [8]. This practice makes a considerable amount of applications vulnerable to backdoor attacks, where attackers can inject hidden triggers into a network to manipulate its output. One example of a backdoor method that can achieve high reliability and stealthiness is WaNet, proposed by Nguyen and Tran [12], which relies on subtle image warping as the trigger mechanism.

While much research has focused on backdoor attacks on Deep Classification Models (DCMs) [5, 11, 12, 13], little work has addressed these attacks on DRMs [10].

This paper first explains the differences between DCMs and DRMs. We then formulate and evaluate a backdoor attack on a DRM model for head position estimation using the WaNet method. We then explore how to generalise the method to DRMs and whether it can be applied to single-channel grayscale images and models trained on them.

2 Related Works

2.1 Backdoor Attacks on Deep Learning Models

In their paper, T. Gu et al. [5] explored a concept of a *backdoored neural network*, which performs well on regular inputs but causes misclassifications whenever inputs satisfy some secret, attacker-chosen property, called *the backdoor trigger*. While much of the research on the topic of backdoored deep learning models focuses on classification tasks [5, 11, 12, 13], very little has been done on regression tasks [10]. Additionally, [10] describes a backdoor on a regression model with a very small number of input features.

Nguyen and Tran [12] describe a backdoor attack method called WaNet, which stands for Warping-based poisoned Networks. Their method is designed for image classification models, is based on stubble image warping and is difficult to detect by both machine and human inspections.

2.2 Differences between DCMs and DRMs

DRMs and DCMs are both types of deep learning models. Their differences lie in their output type:

DRMs predict continuous outputs. That is, the output variables are numeric values within a given range (e.g. temperature, angle, price). A DRM can be described as a function $f : \mathbb{X} \to \mathbb{R}^N$, that maps some predetermined input $\mathbb{X}$ to N real numbers within a given range.

DCMs predict categorical outputs. The output variables in classification are discrete and usually represent different classes or categories (e.g. spam/not spam, dog breed, disease diagnosis). We can describe DCMs as a function $f : \mathbb{X} \to \mathbb{C}$ (single-label classification) or $f' : \mathbb{X} \to \mathcal{P}(\mathbb{C})$ (multi-label classification), where $\mathbb{C}$ is a set of classes.

Generally, DCMs can be converted to DRMs and vice versa by either modifying or adding new final layers, provided that the underlying problem can be reformulated to suit either classification or regression objectives. Despite that, we cannot directly apply the convectional threat model and evaluation metrics of backdoor attacks designed for DCMs on DRMs.

3 Warping-Based Backdoor Attack on a DRM

3.1 Threat Model

Attacker's Goal The attacker's aim is to introduce a backdoor into a DRM. The attacker aims to make the backdoor imperceptible by ensuring the model behaves legitimately whenever the backdoor trigger is not present, with minimal or no loss of accuracy compared to a legitimate model, and by having the trigger itself be imperceptible. Additionally, the attacker aims for the trigger to reliably influence the model's output in a predetermined way.

Attacker's Capabilities The Attacker has access to training data and computational resources and are capable of training and distributing a DRM. To successfully backdoor a DRM, the Attacker must have control over the model's training data and associated labels or have control over the training process itself.

Attack Vector The Attacker trains or alters a model with a backdoor and distributes it by publishing it on platforms such as [18, 24, 21] or selling the model directly to a developer of the target program. The developer downloads the model, inspects it and uses the model in their program without knowing it is compromised. The scope of the attack can be further widened if the model is used in a Machine Learning as a Service platform, such as [17, 19].

Backdoor Attacks on DRMs The clean model $M : \mathbb{X} \to \mathbb{R}^N$ is a DRM for some $N \geq 1$, that takes a $w \times h$ image as input $x \in \mathbb{X}$. The backdoored model $M_{\mathcal{B}} : \mathbb{X} \to \mathbb{R}^N$ is also a DRM with an associated backdoor trigger injection function $\mathcal{B} : \mathbb{X} \to \mathbb{X}$ and function that acts as a backdoor behavior modifier $\mathcal{B}' : \mathbb{R}^N \to \mathbb{R}^N$. Output of $M_{\mathcal{B}}$ is close or equal to the output of M:

$$For\ most\ x \in \mathbb{X}\ \ M(x) \approx M_{\mathcal{B}}(x),$$

unless the input is backdoored, in which case, $\mathcal{B}'$ is applied to its output:

$$For\ most\ x \in \mathbb{X}\ \ M_{\mathcal{B}}(\mathcal{B}(x)) \approx \mathcal{B}'(M(x)).$$

3.2 Warping-Based Trigger

For WaNet, the trigger is a backward warping function $\mathcal{W}$, that takes an image x, and a warping field W_F as input:

$$\mathcal{B}(x) = \mathcal{W}(x, W_F).$$

$W_F \in \mathbb{R}^{w \times h \times 2}$ defines the relative sampling location for each pixel of x: $(\Delta pixel_x, \Delta pixel_y)$. This results in a pixel at position (pos_x, pos_y) in $\mathcal{B}(x)$ being sampled from x's pixel at position $(pos_x + \Delta pixel'_x, pos_y + \Delta pixel'_y)$, where $(\Delta pixel'_x, \Delta pixel'_y)$ are values for (pos_x, pos_y) from W_F. For a result that looks legitimate, all positions are normalised to lay within $[0, w-1]$ and $[0, h-1]$ before being sampled. Additionally, all non-integer sampling values are bilinearly interpolated, resulting in a smooth image. Both normalisation and interpolation are handled by the warping function $\mathcal{W}$

To generate W_F, we first generate a $K \in \mathbb{R}^{k \times k \times 2}$ called *the kernel*. The kernel is generated as follows:

$$K = \psi(rand_{[-1,1]}(k, k, 2)) \times s\ ,$$

where $rand_{[-1,1]}(a, b, c)$ returns a random $[-1, 1]^{a \times b \times c}$ value, k and s being warping parameters, the latter being called *the warping strength*, and ψ being a normalisation function defined as:

$$\psi(\mathbf{A}) = \frac{\mathbf{A}}{\frac{1}{size(\mathbf{A})} \sum_{a_i \in \mathbf{A}} |a_i|}$$

The kernel is then upsampled using bicubic interpolation, denoted as a $\uparrow$: $\mathbb{R}^{k \times k \times 2} \to \mathbb{R}^{w \times h \times 2}$ function, resulting in:

$$W_F = \uparrow (\psi(rand_{[-1,1]}(k, k, 2)) \times s)$$

This process is depicted in Figure 1. Generally, the smaller the s and k values are, the harder it is to spot the warping and harder for the backdoored model to be trained. Figure 2 illustrates the affects of warping parameters on an image. Note that the warping field generation deviates slightly from [12], where a normalisation ϕ is used when generating the warping field, as enforcing sampling positions remain with in the original image bounds is handled by $\mathcal{W}$ normalising its input.

3.3 Running Modes

Similar to other backdoor methods [5, 11], during training, the data from the training dataset can either remain unchanged ("clean mode"), or both images and associated labels can be modified using $\mathcal{B}$ and $\mathcal{B}'$ respectively ("attack mode"). While this approach results in successful backdoors, the models "tend to learn pixel-level artefacts instead of the warping",

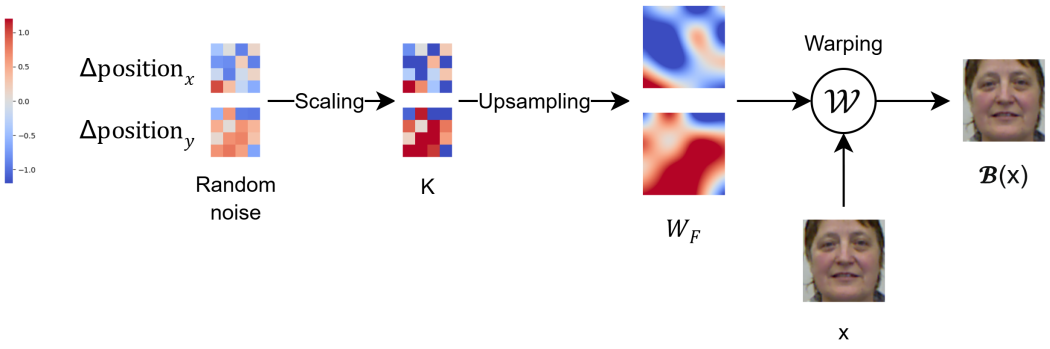

Figure 1: The process of generating a warping field and applying it to an image, resulting in a backdoored image.

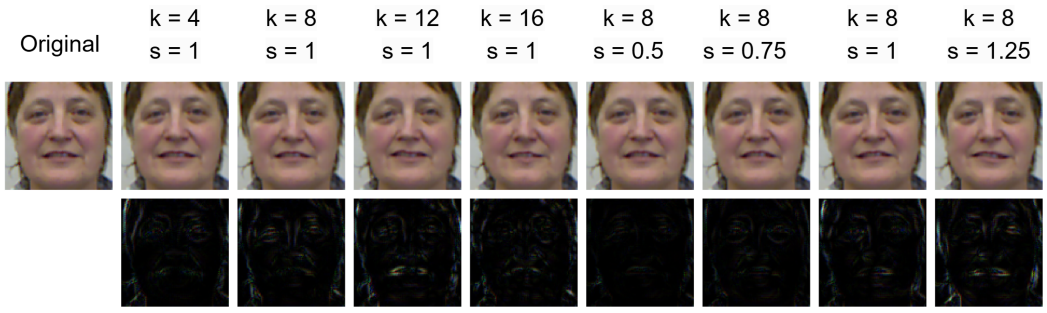

Figure 2: Effects of warping parameters k and s on the warping result. For each warp, we show the warping result (top), and the magnified ($\times 2$) residual image from the original. Image comes form the Pandora Dataset [22]

which makes them "easily exposed by a backdoor defence method such as Neural Cleanse" [12], making the backdoor more detectable.

To combat this, the WaNet paper introduces a separate training mode alongside the clean and attack modes, called the "noise mode". In the noise mode, the label remains unchanged, but the image gets warped using a random warping field W_N, which is not the predetermined backdoor warping field ($W_N \neq W_F$). Given a backdoor probability $\rho_a \in [0, 1]$ and a noise probability $\rho_n \in [0, 1]$, such that $\rho_a + \rho_n \leq 1$, each clean input (x, y) gets processed as follows:

$$(x, y) \mapsto \begin{cases} (x, y) & \text{with probability } 1 - \rho_a - \rho_n \\ (\mathcal{W}(x, W_F), \mathcal{B}'(y)) & \text{with probability } \rho_a \\ (\mathcal{W}(x, W_N), y) & \text{with probability } \rho_n, \text{ for some } W_N \neq W_F \end{cases}$$

4 Experiments

4.1 Experimental Setup

4.1.1 The Model

To carry out the experiments, a head pose estimation DRM, $M_H : \mathbb{X} \to \mathbb{R}^3$, was trained. As input, the model takes a 224×224 3-color-channel (unless specified otherwise) image of a human head and outputs an estimate of the heads pitch, yaw, and roll (Euler angles) in radians relative to the camera from which the image was taken.

The head pose estimator was implemented using a modified ResNet-18 [7] DCM, which uses 17 convolutional layers and one fully connected layer with 1000 neurons on the output layer. To convert the DCM to a DRM, the final (fully connected) layer was changed to have 3 output neurons for yaw, pitch, and roll respectively.

M_H was trained on a *biwi_face_dataset_RGB* dataset [4, 22]. The dataset contains 100×100 3-channel images, which were upscaled to match the 224×224 input size.

Hyperparameters used were:

$$k = 12 \quad s = 1.25 \quad \rho_a = 0.1 \quad \rho_n = 0.2 \quad \text{batch\_size} = 64 \quad \text{learning\_rate} = 0.0001$$

The backdoor behaviour of the model is:

$$\mathcal{B}'(y) = \{\text{yaw} = 0 \text{ rad}, \ \text{pitch} = +1 \text{ rad}, \ \text{roll} = 0 \text{ rad}\}.$$

The dataset was split into 80% for training and 20% for evaluation. The model was trained for 256 epochs with the ADAM optimizer.

4.1.2 Implementation details

All experiments are conducted using Python. The head pose estimator is based on `resnet18` model from the TrochVision python library. Warping was achieved using the `grid_sample` python function from the PyTorch library. The clean dataset is processed in attack, noise or clean modes during training, where the modes are applied to full batches with ρ_a, ρ_n, and $1 - \rho_a - \rho_b$ probability, respectively. To achieve better training performance, the input data is normalised at the pre-processing stage.

For better reproducibility of our experiments, the random seed for `pyTorch`, `numPy`, and `random` python libraries was hard-coded. The code is open source, and can be found at [23]

4.2 Evaluation Metrics

Given an evaluation dataset of size s_e: $\mathbb{S}_e = \{(x_i, y_i) | x_i \in \mathbb{X}, y_i \in \mathbb{R}^3, i = \overline{1, s_e}\}$ and a function $\delta_{ang} : \mathbb{R}^3 \times \mathbb{R}^3 \to [0, 180°]$, which takes two Euler angles (in radians) and outputs absolute angular difference between them; a backdoor is considered to be successful on a poisoned image $\mathcal{B}(x_i)$ if the model outputs a value within 5 degrees of the poisoned label $\mathcal{B}'(y_i)$:

$$\delta_{ang}(\mathcal{B}'(y_i), M_H(\mathcal{B}(x_i))) \leq 5°.$$

The following metrics are then used to evaluate the effectiveness of the models and the backdoor method:

- Mean Absolute Error (**MAE**) describes the performance of a model:

$$\mathbf{MAE} = \frac{1}{s_e} \sum_{i=0}^{s_e} \delta_{ang}(y_i, M_H(x_i)).$$

  This paper considers a model to be accurate, when $MAE \leq 3°$

- Backdoor Success Rate (**BSR**) describes how well a backdoored model recognizes and responds to the backdoor signal:

$$\mathbf{BSR} = \frac{100\%}{s_e} \sum_{i=0}^{s_e} \mathbf{1}_{\delta_{ang}(\mathcal{B}'(y_i), M_H(\mathcal{B}(x_i))) \leq 5°}$$

- Mean Absolute Error for the Backdoor ($\mathbf{MAE_B}$) describes the average angular error between inputs with a backdoor present and poisoned labels:

$$\mathbf{MAE_B} = \frac{1}{s_e} \sum_{i=0}^{s_e} \delta_{ang}(\mathcal{B}'(y_i), M_H(\mathcal{B}(x_i))).$$

- False Positive Rate for Noise ($\mathbf{FP}_N$) describes how often a backdoored model misinterprets a warped input not containing a backdoor as an input with a backdoor by invoking the backdoor behaviour:

$$\mathbf{FP}_N = \frac{100\%}{s_e} \Big(\sum_{i=0}^{s_e} \mathbf{1}_{\delta_{ang}(\mathcal{B}'(y_i), M_H(\mathcal{W}(x_i, W_N))) \leq 5°} - \sum_{i=0}^{s_e} \mathbf{1}_{\delta_{ang}(\mathcal{B}'(y_i), y_i) \leq 5°} \Big),$$

  where $W_N \neq W_F$. In the experimental setup of this paper, this is not strictly enforced. Instead, W_N is randomly regenerated for each batch in the evaluation dataset. While a collision (where $W_N = W_F$) is possible, its impact on the evaluation are negligible.

- False Positive Rate for Clean ($\mathbf{FP}_C$) describes how often a backdoored model misinterprets a clean input as an input with a backdoor by invoking the backdoor behaviour:

$$\mathbf{FP}_C = \frac{100\%}{s_e} \Big(\sum_{i=0}^{s_e} \mathbf{1}_{\delta_{ang}(\mathcal{B}'(y_i), M_H(x_i)) \leq 5°} - \sum_{i=0}^{s_e} \mathbf{1}_{\delta_{ang}(\mathcal{B}'(y_i), y_i) \leq 5°} \Big),$$

A backdoor is considered successful on a model, if $BSR \geq 90\%$ or $MAE_B \leq 3°$, $FP_N \leq 0.5\%$, $FP_C \leq 0.5\%$, and the model itself is accurate. Note, that ($\mathbf{FP}_N$) and ($\mathbf{FP}_C$) are simplified to lower the computational time required for evaluation.

4.3 Experimental Results on 3-Channel Coloured Input

A backdoored model was trained. Its performance over time is depicted in Figure 3, and its performance after epoch 250 is outlined in Table 1.

Metric	BSR	FP_C	FP_N	MAE	MAE_B
Value	99.8406%	0.0899%	0.1218%	1.6242°	1.8139°
Success Threshold	$\geq 90\%$	$\leq 0.5\%$	$\leq 0.5\%$	$\leq 3°$	$\leq 3°$

Table 1: Model performance metrics after epoch 250 (rounded). Hyperparameters used: $k = 12$, $s = 1.25$, $\rho_a = 0.1$, $\rho_n = 0.2$, batch size = 12, learning rate = 0.0001. Trained on 3-channel colored images.

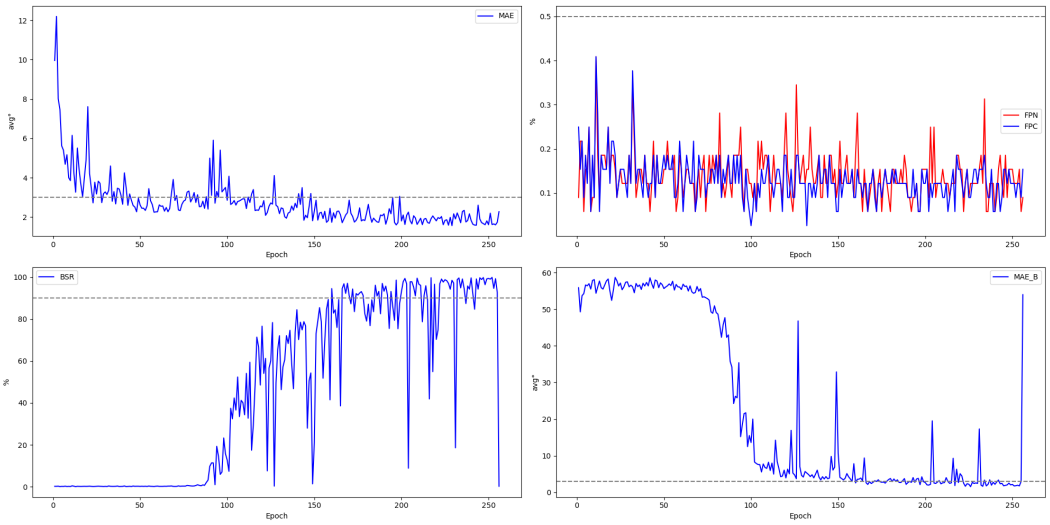

Figure 3: Model performance over training epochs. Horizontal lines indicate metric thresholds that define backdoor success. Hyperparameters used: $k = 12$, $s = 1.25$, $\rho_a = 0.1$, $\rho_n = 0.2$, batch size = 12, learning rate = 0.0001. Trained on 3-channel colored images.

4.4 Experimental Results on 1-Channel Grayscale Input

A similar experiment to the one described in subsection 4.3 was conducted. The model was trained on the same dataset, which was modified during prepossessing to contain single-channel grayscale images. To accommodate this, the first convolutional layer of M_H was modified to accept single-channel images. Table 2 outlines its performance after epoch 250.

4.5 Defence Experiment

A working backdoored model was fine-tuned for 128 epochs with $\rho_a = 0$ and $\rho_n = 0$. Its backdoor performance over fine-tuning epochs is depicted in Figure 5, and its performance after fine-tuning is compared with a backdoored model in Table 3.

Metric	BSR	FP_C	FP_N	MAE	MAE_B
Value	100%	0.0947%	0.1266%	1.7037°	1.3343°
Success Threshold	$\geq 90\%$	$\leq 0.5\%$	$\leq 0.5\%$	$\leq 3°$	$\leq 3°$

Table 2: Model performance metrics after epoch 250 (rounded). Hyperparameters used: $k = 12$, $s = 1.25$, $\rho_a = 0.1$, $\rho_n = 0.2$, batch size = 12, learning rate = 0.0001. Trained on 1-channel grayscale images.

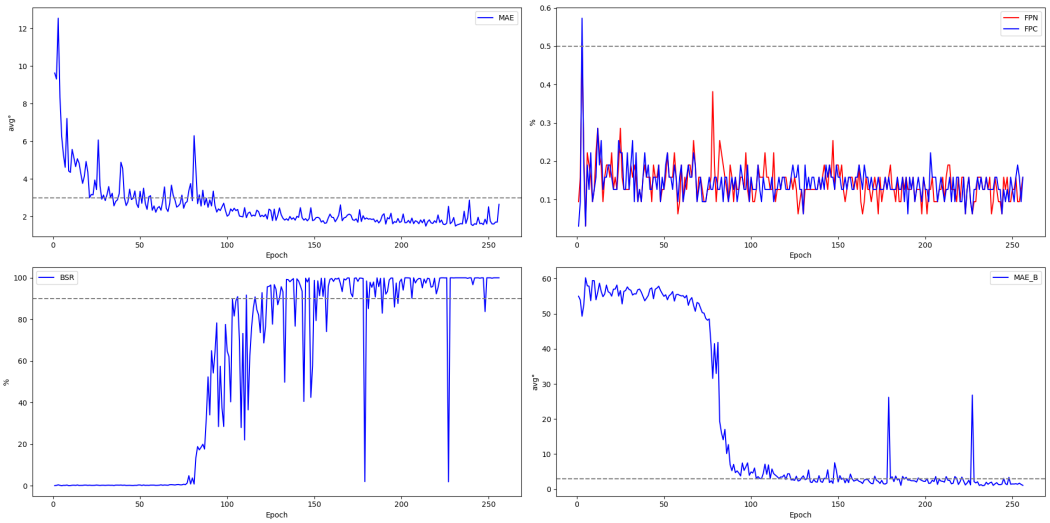

Figure 4: Model performance over training epochs. Horizontal lines indicate metric thresholds that define backdoor success. Hyperparameters used: $k = 12$, $s = 1.25$, $\rho_a = 0.1$, $\rho_n = 0.2$, batch size = 12, learning rate = 0.0001. Trained on 1-channel grayscale images.

4.6 Ablation Experiments

4.6.1 Impact of the Noise Mode on Model Performance

A similar experiment to the one described in subsection 4.3 was conducted, but without noise mode ($\rho_n = 0$). Its performance is outlined and compared to a model trained with noise model in Table 4.

4.6.2 Impact of the Warping Strength on Model Performance

Four backdoored models were trained with varying warping strengths. Their metrics after epoch 250 are compared in Table 5. Figure 6 outlined the impact of the warping strength on models backdoor success rate.

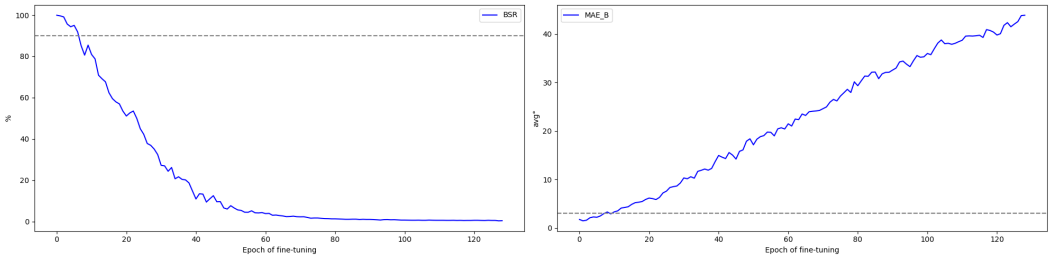

Figure 5: Model performance over fine-tuning time with $\rho_a = 0$, $\rho_n = 0$ for 128 epochs (rounded). Before fine-tuning, the model has a backdoor. Other hyperparameters used: $k = 12$, $s = 1.25$, batch size = 12, learning rate = 0.0001. Trained and fine-tuned on 3-channel colored images.

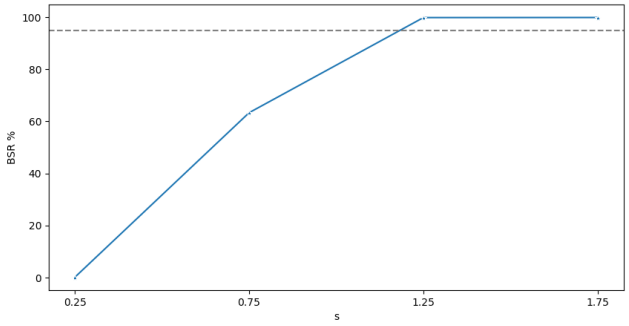

Figure 6: Backdoor success rate comparison for models trained on different s values after epoch 250. Hyperparameters used: $k = 12$, $\rho_a = 0.1$, $\rho_n = 0.2$, batch size = 12, learning rate = 0.0001. Trained on 3-channel colored images.

Metric	BSR	FP_C	FP_N	MAE	MAE_B
Value before fine-tuning	**99.9043%**	0.1266%	0.41358%	1.7124°	**1.7684°**
Value after fine-tuning	**0.3827%**	0.1585%	0.1266%	1.5104°	**43.8345°**
Success Threshold	$\geq 90\%$	$\leq 0.5\%$	$\leq 0.5\%$	$\leq 3°$	$\leq 3°$

Table 3: Model performance metrics before and after fine-tuning a model with $\rho_a = 0$, $\rho_n = 0$ for 128 epochs (rounded). Before fine-tuning, the model has a backdoor. Other hyperparameters used: $k = 12$, $s = 1.25$, batch size = 12, learning rate = 0.0001. Trained and fine-tuned on 3-channel colored images. Differing metrics are highlighted.

Metric	BSR	FP_C	FP_N	MAE	MAE_B
Value without noise mode	99.9681%	0.1585%	**44.0679%**	1.5156°	1.3294°
Value with noise mode	99.8406%	0.0899%	**0.1218%**	1.6242°	1.8139°
Success Threshold	$\geq 90\%$	$\leq 0.5\%$	$\leq 0.5\%$	$\leq 3°$	$\leq 3°$

Table 4: Model performance metrics after epoch 250 (rounded) comparison. At the top: model trained without noise mode ($\rho_n = 0.0$), at the bottom: model trained with noise mode ($\rho_n = 0.2$). Other hyperparameters used: $k = 12$, $s = 1.25$, $\rho_a = 0.1$, batch size = 12, learning rate = 0.0001. Trained on 3-channel colored images. Differing metrics are highlighted.

Metric	BSR	FP_C	FP_N	MAE	MAE_B
Value with $s = 0.25$	**0.1594%**	0.1266%	0.1266%	1.6117°	**58.3233°**
Value with $s = 0.75$	**63.3291%**	0.0947%	0.1266%	2.1160°	**4.8275°**
Value with $s = 1.25$	**99.8406%**	0.0899%	0.1218%	1.6242°	**1.8139°**
Value with $s = 1.75$	**99.8724%**	0.1585%	0.1266%	1.7180°	**1.6108°**
Success Threshold	$\geq 90\%$	$\leq 0.5\%$	$\leq 0.5\%$	$\leq 3°$	$\leq 3°$

Table 5: Model performance metrics after epoch 250 (rounded) comparison. Each model was trained with a different s value. Other hyperparameters used: $k = 12$, $\rho_a = 0.1$, $\rho_n = 0.2$, batch size = 12, learning rate = 0.0001. Trained on 3-channel colored images. Differing metrics are highlighted.

5 Discussion

Experimental results from subsections 4.3 and 4.4 indicate that the WaNet method can be successfully adapted to compromise a DRM. The backdoor works well on both grayscale and coloured images, with the grayscale model achieving marginally better backdoor accuracy. The effectiveness of the backdoor combined with its stealth [12] poses a significant security threat.

By fine-tuning a backdoored model in subsection 4.5, we were able to heavily subdue the backdoor behaviour (BSR and MAE_B) while the metrics describing legitimate model behaviour (FP_C, FP_N, and MAE) remained relatively unchanged. MAE_B of the fine-tuned model being around 43° indicates, that the model was not cleansed of the backdoor in its entirety. However, MAE_B did not plateau over fine-tuning time, which may mean that the backdoor could be completely removed given more fine-tuning time. It is important to note that the defence experiment did not reflect a realistic scenario where the party fine-tuning the backdoored model has access to a different, much smaller dataset than the one used by the attacker to train the poisoned model.

Results included in sub-subsection 4.6.1 indicate that a backdoored model can be trained without the noise mode. However, instead of the poisoned model learning the backdoor itself, the model attributes certain warping-related distortion artefacts, causing false positives for images that were warped but did not contain the backdoor itself. It is also important to note that backdoors in models trained without the noise mode are easier to detect using automated defence mechanisms [12].

From the results in sub-section 4.6.2, we observe that below a certain threshold, a lower warping strength correlates with a lower backdoor success rate, possibly because subtler changes to the image are harder or even impossible to detect by the model. We hypothesize that given more training time, it is possible to train a backdoored model meeting all requirements listed in subsection 4.6.2 using a warping strength of 0.75. It is important to note that the warping strength values used here exceed those in the WaNet paper [12], because our models are trained on larger, upscaled images, necessitating greater s values to achieve comparable effects.

6 Responsible Research

6.1 Reproducibility of Experiments

Experiments in this paper were conducted with reproducibility in mind. The code used will be published at [23]. Training data was published by its creators [4]. In our experiments, we use a set random seed for most, if not all pseudorandom functions.

6.2 Use Of LLMs

ChatGPT [20], a Large Language Model (LLM), was used while writing this paper to generate ideas, assist in the writing process and help find points of improvement in both the paper and code used. The author is aware of the limitations of LLMs and did not rely on ChatGPT for factual information.

7 Conclusions and Future Work

In this paper, we formulated, applied, and evaluated the WaNet backdoor attack on a deep regression model. The method is effective on both grayscale and coloured inputs. The effectiveness of the backdoor can be crippled by fine-tuning the poisoned model.

Given the narrow scope of the experiments, future work could expand on the findings of this paper. By varying the experimental conditions and including more appropriate hyperparameters, researchers could gain a better understanding of the method, its impact, and whether the defence method mentioned in this paper can fully disable the backdoor.

References

[1] A. Agarwal and B. Triggs. 3d human pose from silhouettes by relevance vector regression. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–II, 2004.

[2] Andrea Asperti and Daniele Filippini. Deep learning for head pose estimation: A survey. *SN Computer Science*, 4, 04 2023.

[3] Vasileios Belagiannis, Christian Rupprecht, Gustavo Carneiro, and Nassir Navab. Robust optimization for deep regression. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[4] Guido Borghi, Marco Venturelli, Roberto Vezzani, and Rita Cucchiara. Poseidon: Face-from-depth for driver pose estimation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5494–5503. IEEE, 2017.

[5] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

[6] Guodong Guo, Yun Fu, Charles R. Dyer, and Thomas S. Huang. Image-based human age estimation by manifold learning and locally adjusted robust regression. *IEEE Transactions on Image Processing*, 17(7):1178–1188, 2008.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[8] Natasha Kees, Yaxuan Wang, Yiling Jiang, Fang Lue, and Patrick P.K. Chan. Segmentation based backdoor attack detection. In *2020 International Conference on Machine Learning and Cybernetics (ICMLC)*, pages 298–302, 2020.

[9] Stephane Lathuiliere, Pablo Mesejo, Xavier Alameda-Pineda, and Radu Horaud. A comprehensive analysis of deep regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(9):2065–2081, 2020.

[10] Xi Li, George Kesidis, David J. Miller, and Vladimir Lucic. Backdoor attack and defense for deep regression, 2021.

[11] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. Reflection backdoor: A natural backdoor attack on deep neural networks. *CoRR*, abs/2007.02343, 2020.

[12] Tuan Anh Nguyen and Anh Tuan Tran. Wanet - imperceptible warping-based backdoor attack. In *International Conference on Learning Representations*, 2021.

[13] Samaneh Shamshiri and Insoo Sohn. Defense method challenges against backdoor attacks in neural networks. In *2024 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 396–400, 2024.

[14] Gaurang Sonkavde, Deepak Sudhakar Dharrao, Anupkumar M Bongale, Sarika T Deokate, Deepak Doreswamy, and Subraya Krishna Bhat. Forecasting stock market prices using machine learning and deep learning models: A systematic review, performance analysis and discussion of implications. *International Journal of Financial Studies*, 11(3):94, 2023.

[15] Jayaraman J Thiagarajan, Vivek Narayanaswamy, Puja Trivedi, and Rushil Anirudh. Pager: A framework for failure analysis of deep regression models. *arXiv preprint arXiv:2309.10977*, 2023.

[16] Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. It's written all over your face: Full-face appearance-based gaze estimation. *CoRR*, abs/1611.08860, 2016.

[17] Amazon sagemaker - aws. [Online]. https://aws.amazon.com/sagemaker/.

[18] Aws marketplace. [Online]. https://aws.amazon.com/marketplace.

[19] Azure machine learning - ml as a service | microsoft azure. [Online]. https://azure.microsoft.com/en-gb/products/machine-learning.

[20] Chatgpt. [Online]. https://chatgpt.com/.

[21] Models - hugging face. [Online]. https://huggingface.co/models.

[22] Pandora dataset. [Online]. https://aimagelab.ing.unimore.it/pandora/.

[23] Public repository with code used for this paper. [Online]. https://github.com/AStyslavski/WaNet-On-DRM-RELEASE.

[24] Tensorflow hub. [Online]. https://www.tensorflow.org/hub.