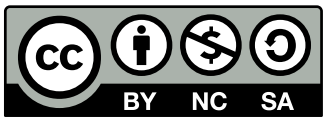
Aalto University
School of Science

Master's programme in Life Science Technologies

Backdoor attacks on large transformer-based regression model

Rudolf Mård

Master's Thesis
2024



Aalto University
School of Science

Author Rudolf Mård

Title Backdoor attacks on large transformer-based regression model

Degree programme Life Science Technologies

Major Bioinformatics and Digital Health

Supervisor Prof. Alex Jung

Advisor Dr Yu Tian

Date 5 July 2024 **Number of pages** 32 **Language** English

Abstract

Previous research on deep learning models under data poisoning attacks is largely limited to studying models trained for classification tasks. However, many problems are more suitably formulated as regression tasks, where the prediction targets of the model are continuous variables. This thesis explores the behavior of a large transformer-based regression model under a certain type of data poisoning attack called backdoor attack. Furthermore, this exploratory research was confined to study the model's behavior during early training phase. To study the impact that these type of attacks has on the selected target model, an implementation of a state of the art deep learning-based weather prediction model, Pangu-Weather, was created. The experiments conducted in this thesis applied a simple backdoor attacking scheme to the training process of the target model. The backdoor attacking scheme involves embedding a trigger-pattern to the input data points of the model and poisoning the prediction target values by applying a multiplier of 0.5 to them. The goal of the attack is to make the model produce 50 percent lower predictions when the trigger pattern is present in the input. After training copies of the target model on clean and poisoned data, their performance was compared to each other under normal prediction making conditions and when exposed to data poisoning attacks. The experiments conducted in this thesis finds that effects of the applied backdoor attacks on behavior of the target model are prominently visible even after a short training period. The poisoned models were observed to achieve lower root mean squared error values when making predictions on clean data as opposed to the target model trained on clean data. The poisoned models were also observed to produce outlying root mean squared error values when comparing the models' predictions made on poisoned input data to poisoned prediction targets. However, the performance and behavior of the poisoned models were observed to only change minimally when embedding input data points with a trigger-pattern associated with the backdoor attacks, indicating that the malicious learning task of producing controlled false predictions was not learned by the target model this early into the training phase.

Keywords Machine learning, Deep learning, Transformer, Regression, Data poisoning, Backdoor attack

Aalto-yliopisto
Perustieteiden
korkeakoulu

Tekijä Rudolf Mård

Työn nimi Takaovihyökkäykset suuressa transformerpohjaisessa regressiomallissa

Koulutusohjelma Life Science Technologies

Pääaine Bioinformatics and Digital Health

Työn valvoja Prof. Alex Jung

Työn ohjaaja TkT Yu Tian

Päivämäärä 5.7.2024 **Sivumäärä** 32 **Kieli** englanti

Tiivistelmä

Aikaisemmat tutkimukset datamyrkytyshyökkäysten kohdistamisesta syväoppimismalleihin ovat suurelta osin rajoittuneita tarkastelemaan malleja jotka on koulutettu suorittamaan luokittelutehtäviä. Monet ongelmat ovat kuitenkin paremmin kuvattavissa regressiotehtävinä, joissa malli pyrkii ennustamaan tietyn muuttujan saamia jatkuvia arvoja. Tämä työ tutkii suuren transformerpohjaisen regressiomallin käyttäytymistä tietyn tyyppisen datamyrkytyshyökkäyksen, nimeltään takaovihyökkäys, aikana. Lisäksi tämä tutkiva työ rajoittuu selvittämään kuinka kyseiset hyökkäykset vaikuttavat kohteena olevan mallin käyttäytymiseen koulutusvaiheen alussa. Takaovihyökkäysten vaikutusten tutkimista varten rakennettu kohde malli perustuu viimeisimpään syväoppimiseen pohjautuvaan sääennuste malliin, nimeltään Pangu-Weather. Tässä työssä toteutetut kokeet käyttivät yksinkertaista takaovihyökkäys strategiaa kohde mallin koulutusprosessin aikana. Takaovihyökkäys toteutettiin upottamalla malliin syötettäviin data pisteisiin laukaisin kuvio ja myrkyttämällä ennustekohteet puolittamalla niiden arvo. Tämän hyökkäyksen tarkoituksena on saada kohdemalli tuottamaan 50 prosenttia alhaisempia ennusteita laukaisin kuvion läsnäollessa mallin syötteessä. Kohde mallista koulutettiin kopioita sekä puhtaalla että myrkytetyllä datalla, ja niiden suorituskykyä vertailtiin toisiinsa normaaleissa ennustamisen olosuhteissa sekä datamyrkytykselle altistettuina. Tämän työn tutkimusten perusteella havaittiin että kokeissa sovellettujen takaovihyökkäysten vaikutukset kohde mallin käyttäytymiseen olivat selkeästi havaittavissa jopa lyhyen koulutusvaiheen jälkeen. Kun ennusteita tehtiin puhtaalle datalle, myrkytettyjen mallien havaittiin saavuttavan matalampia virheen neliöllisiä keskiarvoja kuin mallin joka oli koulutettu puhtaalla koulutus datalla. Myrkytettyjen mallien havaittiin myös tuottavan poikkeavia virheen neliöllisiä keskiarvolukuja kun mallien myrkytetyn datan pohjalta tehtyjä ennusteita verrattiin myrkytettyihin ennustekohteisiin. Myrkytettyjen mallien suorituskyvyn sekä käyttäytymisen havaittiin kuitenkin muuttuneen vain vähäisesti kun malleihin syötettyihin data pisteisiin upotettiin takaovihyökkäyksiin liittyvä laukaisin kuvio. Tämä tulos osoittaa ettei myrkytetyt mallit kyenneet oppimaan takaovihyökkäysten ajamaa tavoitetta tuottaa kontrolloituja virhe-ennusteita näin lyhyen koulutusvaiheen aikana.

Avainsanat Koneoppiminen, Syväoppiminen, Transformer, Regressio, Datamyrkytys, Takaovihyökkäys

Contents

Symbols and abbreviations

Symbols

x	Input features of a data point
y	Prediction target of a data point
$\hat{y}$	Machine learning model's prediction for y
D	Dataset
$a \in b$	a belongs to set b
$\mathbb{R}$	Set of real numbers
$\mathbb{R}^d$	Set of d dimensional real-valued vectors
$\mathbb{R}^{c \times d}$	Set of real-valued matrices with c rows and d columns
Δt	Lead time of a weather forecasting model
ρ	Poisoning proportion
τ	Trigger-pattern multiplier
ω	Prediction target multiplier

Operators

∇_W	Gradient with respect to parameters W of a deep learning model
$\sum_i$	Sum over index i
$a \cdot b$	Dot product of vectors a and b, or element-wise multiplication if either a or b is a scalar.
A^T	Transpose of matrix A
AB	Matrix multiplication between matrices A and B

Abbreviations

ANN	Artificial neural network
RMSE	Root mean squared error
MSE	Mean squared error
MAE	Mean absolute error
CI	Confidence interval
hPa	Hectopascal
m	Meter

1 Introduction

The rise in popularity and adaptation of machine learning solutions has surfaced questions about trustworthiness, integrity and robustness of these technologies. One aspect of these concerns considers how malicious actors are able to manipulate the machine learning models, and how to develop robust counter measures for defending against such manipulation efforts. Machine learning models learn to perform tasks by analysing example data. These tasks could involve classifying images into predefined categories or predicting values of a continuous variable, also known as regression tasks. By carefully tampering with the examples provided to the machine learning model, one might be able to influence how a model behaves when performing it's tasks. These acts of manipulation are known as data poisoning attacks. Data poisoning is an active field of research, and effective methods have been demonstrated for manipulating machine learning models, as well as defending against these attacks. This thesis focuses on deep learning, which is a subcategory of machine learning. Deep learning models are complex and flexible, and have the capacity to learn complicated tasks. Previous research on deep learning models under data poisoning attacks has mainly focused on models performing classification tasks, and the studies that consider regression tasks are limited to simpler machine learning algorithms. This thesis studies the behavior of a large transformer-based regression model under a certain type of data poisoning attack called backdoor attack. Backdoor attacks involve embedding a trigger-pattern into the training data points paired with a manipulated prediction target. The goal of this attack is to make the model associate the presence of the trigger-pattern to a specific output. The target model for these attacks is an implementation of Pangu-Weather, a transformer-based deep learning model developed for predicting values of atmospheric variables. The backdoor attack implemented in this thesis applies a patch-based trigger pattern into the input values of atmospheric variable data in attempt to make the target model make 50% lower predictions for termperature values. Since the behavior of these type of models is not studied before, the goal of this thesis is to explore how the target model behaves under a simple backdoor attack. Due to the sheer size of the target model and the huge amount of data required to fully train it, this thesis further focuses on the behavior of the target model during early training phase. To explore the behavior of the target model under backdoor attacks, multiple copies of the model are trained. Firstly, one copy of the target model is trained on clean training data. Then, four other copies of the model are trained on poisoned training data, varying the design choices of the implemented backdoor attack. Once the models are trained, their performance is evaluated on a test dataset by calculating root mean squared error (RMSE) values. The performance of each poisoned model is then compared to the model trained on clean training data. In addition, the prediction making behavior of the poisoned models is assessed under normal conditions and under the influence of the attackers manipulation efforts. The assessment of behavior is conducted through examining the RMSE value distributions obtained when producing prediction on clean and poisoned test datasets.

2 Background

The first subsection, Introduction to deep learning, discusses the basic principles of deep learning and presents the notations used throughout this thesis. The next two subsections, Fundamentals of transformers and Evolution of vision transformers, presents a technical review of deep learning techniques that are relevant to the subject model of this thesis: Pangu-weather. These subsections dive into the details of transformer neural network architecture and the successive research that has adapted the transformer architecture to process image data. These solutions lay down the basis for the next subsection, Pangu-weather, which presents the model architecture and the study behind the deep learning model for global forecasting of atmospheric variables referenced in the title. The last subsection, Data poisoning, addresses the security threat referenced in it's title concerning machine learning systems and presents the terminology as well as the current state of research in the field. This thesis is dedicated to study data poisoning attacks on a certain deep learning model. In order to maintain a well focused scope for the work, this thesis will not extensively cover machine learning techniques or security threats outside of supervised learning, regression problems and data poisoning.

2.1 Introduction to deep learning

Machine learning is a field of artificial intelligence that involves the development of algorithms which learn to perform a certain task through examination of example data and mathematical optimization of a loss function. To expand on this in the context of deep learning, we adopt a framework which conceptualizes machine learning as three distinct, but interacting, components [1]. These three components — data, model and loss — serve as the foundational elements guiding our introduction to deep learning.

Data in machine learning typically consists of multiple instances, each represented by a set of distinct features. These features are individual measurable properties or characteristics of the phenomenon being studied. The features might involve a set of atmospheric variables, text in the form of sentences in different languages or an image of an object. Oftentimes, machine learning methods are used for predicting some variable of interest based on the collected features. Let's refer to the features with letter x and the predicted target variable with letter y. In classification tasks the target variable y belongs to a discrete set of distinct class labels. In contrast, regression problems refer to situations where the target variable y takes a continuous value. In supervised machine learning the data contains examples of both features and the corresponding target variable. A data point then refers to one example instance (x, y) which contains values for the features as well as value of the target variable. However, a data point can also refer to just x when y is unknown. A collection of data points is called a dataset. A dataset consisting of N data points is denoted by $D = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(N)}, y^{(N)})\}$. In machine learning, the data is usually divided into non-overlapping sets with different use cases. A training dataset is used for training a machine learning model, while a validation dataset is used during the model development to assess how the model performs on previously unseen data.

Since changes to the model are allowed to increase the performance on validation data, a distinct test dataset is used to provide an unbiased final assessment of the trained model.

Model in the context of machine learning is a mathematical machinery which takes the features of data points as an input and outputs predictions of the target variable. A machine learning model aims to learn the relationship between the input features and the target variable through a procedure called training, which involves examination of example data. Models that are capable of finding patterns and relationships intrinsic to the data are advantageous especially in applications where obtaining satisfactory performance by writing explicit instructions for the computer is not feasible. Deep learning is a branch of machine learning where a model is constructed by stacking many layers of artificial neural network (ANN) components. The basic principles of ANNs draw their inspiration from biological neurons in the brain that form networks and signaling pathways via connections between each other. Analogous to neurons in biological neural networks, the ANNs consist of single units called nodes. A node is a function which takes a feature vector as an input and produces a single scalar output. Consider a node in the first layer of the ANN, which takes a feature vector $x \in \mathbb{R}^d$ as an input where x represents a single data point consisting of d real-valued features. The computation performed by a single node is depicted in eq. 1 where the weight vector $w \in \mathbb{R}^d$ and bias term $b \in \mathbb{R}$ are trainable parameters of the model, specific to that particular node. The σ in eq. 1 is a non-linear activation function, such as sigmoid function or rectified linear unit (ReLU), which enables the ANN to capture non-linear relationships between the input features and the predicted variable.

$$\text{node}(x) = \sigma\left(\sum_{i=1}^{d} x_i w_i + b\right) \tag{1}$$

ANNs are organized into layers of multiple parallel nodes. Each node within a layer processes the input vector independently. Consider a layer consisting of n nodes, each producing a scalar output. These outputs are aggregated into a vector $z \in \mathbb{R}^n$, which serves as the input vector for the subsequent layer of nodes. The output produced by the last layer of the network serves as the models prediction of the target variable y. Let's denote the prediction produced by the model with $\hat{y}$. Increasing the number of layers inside an ANN allows the model to build representations of wider variety of patterns and to approximate more complex functions. This is due to the model forming more abstract representations of the data in the layers deeper in the model. The layers described here are commonly referred to as fully connected or dense layers. A fully connected layer without an activation function is known as a linear layer. While the name artificial neural network is also used for referring to other types of deep learning networks, models only consisting of fully connected layers are commonly referred to as feed-forward neural networks or multilayer perceptrons.

Loss function is an essential component of machine learning allowing the optimization of the model parameters. A loss function takes the models predictions $\hat{Y} = [\hat{y}^{(1)}, \hat{y}^{(2)}, \ldots, \hat{y}^{(N)}]$ on N data points $X = [x^{(1)}, x^{(2)}, \ldots, x^{(N)}]$ and the corresponding a ground-truth values of the target variable $Y = [y^{(1)}, y^{(2)}, \ldots, y^{(N)}]$ as

inputs and outputs a scalar value reflecting the performance of the model. Common loss functions for regression models include mean squared error (MSE)(eq. 2) and mean absolute error (MAE)(eq. 3). These functions calculate the mean of the error between the ground-truth values of y and the predictions $\hat{y}$ made by the model over N data points. Since the error can take either positive or negative values, the magnitudes of the errors has to be converted to positive values before calculating the mean. MSE does this by squaring the error, while MAE takes the error's absolute value. Since the MSE calculates the square of the error, the loss value is growing more rapidly than with the MAE as the error increases. This leads to MSE penalizing the model more for large errors than with MAE.

$$\text{MSE}(Y, \hat{Y}) = \frac{1}{N} \sum_{i=1}^{N} (y^{(i)} - \hat{y}^{(i)})^2 \tag{2}$$

$$\text{MAE}(Y, \hat{Y}) = \frac{1}{N} \sum_{i=1}^{N} |y^{(i)} - \hat{y}^{(i)}| \tag{3}$$

When first creating the deep learning model, values of the trainable parameters are initialized to some initial estimate. These initial values will not produce accurate predictions of the target variable, and the loss function will produce a high value. Let's denote the deep learning model with a function f. Since the predictions are made by the model and depend on the data points X and the trainable parameters W, we can denote the predictions made by the model with $\hat{Y} = f(X, W)$. Additionally, lets denote the loss function with L. For the model to produce good predictions, the trainable parameters has to minimize the loss function L. The most common practice for optimizing the models performance is to iteratively tune the trainable parameters through the gradient descent algorithm [2]. The basic principle of gradient descent is to calculate the gradient of the loss function with respect to the trainable parameters and use that gradient to update the trainable parameters in the direction which decreases the loss the most. The algorithm iterates between the steps depicted in equations 4 and 5. During one iteration, the gradient of the loss function is evaluated using the current estimate of the model parameters and the training dataset. This step corresponds to eq. 4. After obtaining the gradient, the model parameters are updated according to eq. 5, where α is called the learning rate. Since the gradient of a function can be conceptualized as a vector pointing in the direction of steepest increase of the function's value, tuning the model parameters in the opposite direction will ideally yield smaller loss in the following iteration. The magnitude of the loss as well as the value of α dictate the amount of change in the model parameters. This process is repeated until a satisfactory loss value has been reached.

$$g = \nabla_W L(Y, f(X, W)) \tag{4}$$

$$W^{(k+1)} = W^{(k)} - \alpha \cdot g \tag{5}$$

In practice, deep learning models require large datasets to train on. Therefore, it might not be feasible to evaluate the average gradient over the whole training dataset at once. Performing the gradient evaluation and parameter updates on one batch of the training dataset at a time is called stochastic gradient descent (SGD). The number of data points in a batch is referred to as batch size.

2.2 Fundamentals of transformers

Transformer architecture is a powerful configuration of artificial neural network components and information processing mechanisms. Recent years has seen a massive rise in popularity of large language models which are based on transformer neural networks trained in unprecedented scales [3] [4]. The transformer architecture was initially introduced by A. Vaswani et. al. [5] for language translation tasks and has since been a staple deep learning model architecture for language modeling problems.

The core unit of the transformer architecture is the transformer block shown in fig. 1. These blocks consists of a multi-head attention layer and a feed-forward network. The inputs and outputs of both multi-head attention layer and feed-forward network are joined through residual connections to enhance training stability [6]. The transformer block takes a sequence of tokens as an input, and outputs a sequence of equal length. In the context of language modeling, these tokens could refer to individual words in a sentence. The first layer of the transformer block, multi-head attention, deploys a mechanism called self-attention. The self-attention mechanism allows information to be pooled from every token in a sequence into each individual token. In essence, this information pooling is carried out by separately replacing each individual token with the weighted average of every token in the sequence. For a particular token $t^{(i)}$ in sequence $S = [t^{(1)}, t^{(2)}, \ldots, t^{(T)}]$ of T tokens, every token $t^{(j)}$ in that sequence is assigned with an attention weight a_{ij}. These weights reflects the relevance of each token $t^{(j)}$ to the token $t^{(i)}$ and are used to calculate the weighted average of tokens to replace token $t^{(i)}$ in the sequence. In order to calculate the attention weights in a data-dependent manner, each token t in the sequence is transformed into query and key vectors (q and k, respectively) by applying feed-forward neural networks to the tokens. The query vector $q^{(i)}$ represents features that are considered relevant by token $t^{(i)}$, and the key vector $k^{(i)}$ represents features that token $t^{(i)}$ contains. The attention weight a_{ij} between tokens $t^{(i)}$ and $t^{(j)}$ is obtained by calculating the dot product between the query vector $q^{(i)}$ and key vector $k^{(j)}$. Thus, the magnitude of attention weights are influenced by the degree of similarity between the corresponding features in the query and key vectors. For T number of key and query vectors coming from a H-dimensional vector space, the calculation of attention matrix A containing the attention weights a_{ij} can be formulated as a matrix multiplication between the query matrix $Q \in \mathbb{R}^{T\times H}$ and key matrix $K \in \mathbb{R}^{T\times H}$ as $A = QK^T \in \mathbb{R}^{T\times T}$. The attention matrix is further divided by $\sqrt{d}$, where d denotes the dimensionality of the vector space of query and key vectors. The attention matrix is passed through a softmax function in order to make the attention weights sum up to one. The division by $\sqrt{d}$ is performed to avoid the softmax function from producing sharply concentrated distributions [5]. Instead of calculating the weighted averages of raw tokens, the tokens are transformed into

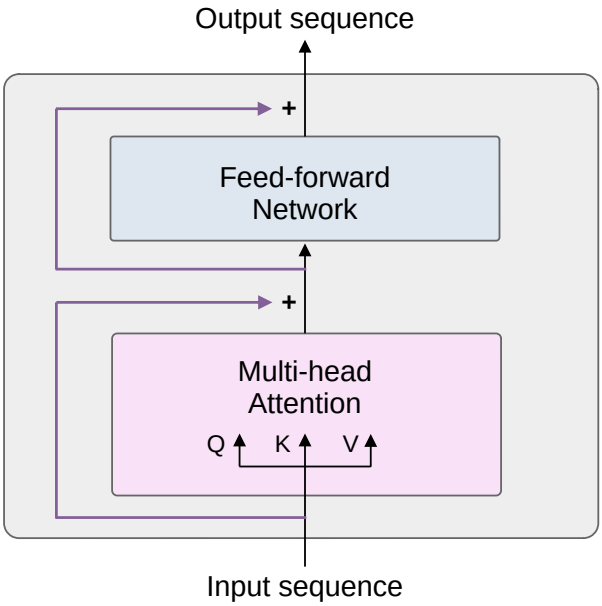


Figure 1: Diagram depicting the information flow through a transformer block. The Q, K, and V in the figure refers to the query, key and value matrices, respectively. The purple arrows represent the residual connections joining the input tokens to the output tokens of each sub-block via addition.

H-dimensional vector space through a feed-forward neural network to produce what are referred to as value vectors. Similarly to query and key matrices, this generates a value matrix $V \in \mathbb{R}^{T\times H}$. The weighted averages of tokens are obtained from a matrix multiplication between the attention weights and the value matrix, leading to the commonly known equation describing the self-attention computation:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V \tag{6}$$

One sub-layer performing the self-attention computation is referred to as an attention head. The multi-head attention layer deploys multiple parallel attention heads, each capable of attending to different sets of token features. Lastly, the corresponding tokens from each attention head are concatenated and passed token-wise through a linear layer. The final component in the transformer block is a feed-forward neural network, which is applied token-wise to the sequence to introduce non-linearity to the model and increasing it's representation power.

2.3 Evolution of vision transformers

Success of the transformer architecture in natural language processing set in motion the subsequent research in its applicability for computer vision tasks. A naive approach for applying the self-attention mechanism to image data would be to consider each pixel in an image as an individual token. However, since the number of attention weights grows quadratically in the number of tokens, applying self-attention to high-resolution images becomes computationally infeasible. To tackle this problem, A. Dosovitskiy

et. al. [7] presented the Vision Transformer (ViT) model for image classification and described how the self-attention mechanism could be more efficiently applied to token sequences derived from image data. The model architecture of the ViT is designed to follow the original transformer architecture as closely as possible. In fact, the only major modification to the original model architecture was made to the input layer. The input layer of the ViT first splits the input image into non-overlapping image patches of fixed size. The patches are then flattened and transformed into token vectors through a linear layer. This procedure is referred to as patch embedding. In addition, learnable positional embeddings are added to the tokens after patch embedding to incorporate information about the locations of the image patches in the original input image. Even though ViT is found to be a viable approach for image classification tasks, the patch embedding alone was found to be inadequate for other computer vision tasks, such as semantic segmentation. This is due to patch embedding partitioning the images into rather large patches leading to low-resolution feature maps. Furthermore, applying the self-attention mechanism straight to the token sequence considers the tokens only on a global scale, lacking the ability to form hierarchical feature representations at various scales.

To establish a robust and versatile backbone architecture for computer vision tasks, Z. Liu et. al. [8] proposed the Swin transformer architecture. Swin transformer adopts the patch embedding technique used in the ViT model to tokenize input images. In order to maintain the high resolution of input data, the input images are split into small patches. To counteract the overhead caused by increase in the number of tokens, the Swin transformer replaces the original self-attention mechanism with shifted window based self-attention. Instead of calculating global self-attention between tokens, a Swin transformer block divides the incoming feature map into non-overlapping windows and calculates self-attention within each one of them. Furthermore, to enable information flow between the self-attention windows, the feature maps are shifted by displacing the tokens by half of the window size between each consecutive Swin transformer blocks. Consider the tokens being placed on a grid forming the image. The tokens are shifted towards a corner of the grid such that the tokens being displaced beyond its boundaries are wrapped around to the opposite side. Beyond the shifted window self-attention mechanism, the Swin transformer employs a strategy to construct hierarchical feature representations of the data. The ability to detect features at different scales of image data is found to be an important property of systems performing complex visual tasks [9][10]. The Swin transformer accomplishes this by incorporating patch merging layers between the Swin transformer blocks. The patch merging layers reduce the spatial resolution of the feature maps by concatenating 2x2 neighboring tokens, and applies a trainable linear layer to reduce the feature dimensionality of the concatenated tokens by a factor of two. This way the Swin transformer blocks following the patch merging layers are able to attend to larger scale patterns in the image data.

2.4 Pangu-weather

Pangu-weather is a deep neural network developed for short-to-medium term global forecasting of atmospheric variables developed by K. Bi et. al. [11]. The model

takes two sets of variables as input, referred to as upper-air variables and surface variables, and predicts their values after a certain lead time Δt. Both sets of input data encompasses measurements for each coordinate on a global latitude-longitude grid in resolution of $0.25° \times 0.25°$. The spatial resolution of the input tensors on earth's surface is therefore $(1440, 721)$ for longitude and latitude, respectively. Furthermore, the measurements of upper-air variable also span 13 pressure levels (50hPa, 100hPa, 150hPa, 200hPa, 250hPa, 300hPa, 400hPa, 500hPa, 600hPa, 700hPa, 850hPa, 925hPa and 1000hPa). The upper-air variables consists of five quantities: geopotential, specific humidity, temperature, and the u-component and v-component of wind speed. The surface variables include four quantities: 2-m temperature, mean sea level pressure (MSLP) and the u-component and v-component of 10m wind speed. The inputs are arranged into tensors of size $(13, 1440, 721, 5)$ and $(1440, 721, 4)$ for upper-air variables and surface variables, respectively. The model predictions for upper-air and surface variables are arranged to tensors of the same shapes as the inputs. Let's denote the upper-air variable tensor at time point t with $x_{air}^{(t)}$ and surface variable tensor with $x_{sur}^{(t)}$. The dataset then consists of tensor pairs $(x_{air}^{(t)}, x_{sur}^{(t)})$ measured at hourly time points. The ground-truth prediction target for input pair $(x_{air}^{(t)}, x_{sur}^{(t)})$ is denoted with $(y_{air}^{(t)}, y_{sur}^{(t)}) = (x_{air}^{(t+\Delta t)}, x_{sur}^{(t+\Delta t)})$

The model architecture of the Pangu-weather is depicted in fig. 2. Pangu-weather is constructed on the foundation of the Swin transformer model architecture. While Swin transformer is developed for processing 2D image data, Pangu-weather is adapted to process 3D input volumes. Input layer of the model adopts the patch embedding technique to partition the input volumes into tokens using patch size of $(2, 4, 4)$ for $x_{air}^{(t)}$ and $(4, 4)$ for $x_{sur}^{(t)}$. Before applying patch embedding to $x_{sur}^{(t)}$, it is concatenated with three constant masks (topography mask, land-sea mask and soil-type mask) along the variable dimension. The concatenation of tokens within a patch and the appliance of a linear layer was implemented as 3D and 2D convolution layers using kernel sizes and strides equal to the corresponding patch sizes. The patch embedding projects the variables of both input tensors into $C = 192$ dimensional feature space, and concatenates the patch embedded input tensors along the pressure level axis. Following this, the amalgamated data is processed by a pair of shifted window transformer blocks before down-sampled by a patch merging layer. The patch merging layer functions as described in the subsection 2.3 by concatenating 4 neighboring tokens, separately at each pressure level, and halving the feature dimensionality. The condensed data volume then traverses through 12 additional shifted window transformer blocks. Subsequently, the data volume is up-sampled back to higher resolution by a up-sampling layer, acting as a counter part to the patch merging layer. The data tensor is up-sampled by applying a linear layer doubling the feature dimension, and reshaping the tensor to the same shape as it was before applying the patch merging layer. The up-sampling layer also applies another linear layer to the reshaped tensor to mix the token data, preserving the feature dimensionality. Finally, the data volume is passed through another set of two shifted window transformer blocks. The process concludes with the data being reshaped to match the initial input shapes through a patch recovery layer, acting as a counter part to the patch embedding layer. The patch recovery layer first divides the

data tensor into two slices corresponding to the upper-air and surface tensors. Then, 2D and 3D transposed convolution layers are applied to these slices to obtain tensors with the same shapes as the input tensors. Let's denote the predictions of the upper-air variables as $\hat{y}_{air}^{(t)}$, and the predictions of the surface variables as $\hat{y}_{sur}^{(t)}$.

The authors of Pangu-weather trained 4 models with the described model architecture using different lead times at 1h, 3h, 6h and 24h. Each model was trained for 100 epochs with 39 years worth of hourly data (from 1979 to 2017). Training of one model was reported to take approximately 16 days on a computer cluster with 192 Nvidia V100 accelerators. For training, the authors reported using Adam optimizer, batch size of 192 (one sample per GPU), initial learning rate of 0.0005 and the mean absolute error loss function (eq. 3). One of the evaluation metrics used in the study was the root mean squared error (RMSE). RSME is essentially the same as the MSE function presented in equation 2 with the distinction that RMSE involves taking the square root of the output value. Taking the square root of MSE transforms the output value to the same scale as the input values. The RMSE function used for evaluating the performance of Pangu-weather is presented in equation 7. The equation considers calculating the RMSE between the ground truth prediction targets and the predictions made by the model for an input pair $(x_{air}^{(t)}, x_{sur}^{(t)})$. For this purpose the prediction target tensors $(y_{air}^{(t)}, y_{sur}^{(t)})$ are combined into a tensor $Y^{(t)}$ of shape $(N_{lat} = 1440, N_{lon} = 721, 69)$, where 69 is the total number of variables; 4 surface variables and 13×5 upper-air variables. Similarly, the predictions $(\hat{y}_{air}^{(t)}, \hat{y}_{sur}^{(t)})$ are combined into a tensor denoted by $\hat{Y}^{(t)}$. The mean of the squared errors is calculated over the latitude and longitude dimensions, eventually resulting in a vector of RMSE values corresponding to the 69 variables. The RMSE equation (eq. 7) applies latitude weights $\theta(i)$ to each squared error term based on the latitude angle coordinate ϕ_i of the corresponding term. Calculation of the latitude weights is depicted in equation 8. The latitude weights are applied to account for the differences in the areas of grid cells across latitude coordinates.

$$\text{RMSE}\left(Y^{(t)}, \hat{Y}^{(t)}\right) = \sqrt{\frac{\sum_{i=1}^{N_{lat}} \sum_{j=1}^{N_{lon}} \theta(i) \left(Y_{i,j}^{(t)} - \hat{Y}_{i,j}^{(t)}\right)^2}{N_{lat} \cdot N_{lon}}} \tag{7}$$

$$\theta(i) = N_{lat} \cdot \frac{cos\phi_i}{\sum_k^{N_{lat}} cos\phi_k} \tag{8}$$

The conventional numerical weather prediction (NWP) systems, which rely on numerical simulations, have long been recognized as the state-of-the-art and most widely utilized methods for medium-range weather forecasting. However, the authors of Pangu-weather claim that their model is capable of producing "better deterministic forecast results on reanalysis data than the world's best NWP system, the operational IFS (integrated forecasting system) of ECMWF (European Centre for Medium-Range Weather Forecasts)" [11]. In addition, the authors disclose that the inference speed of their model is on the order of seconds on a single GPU, while the simulations of

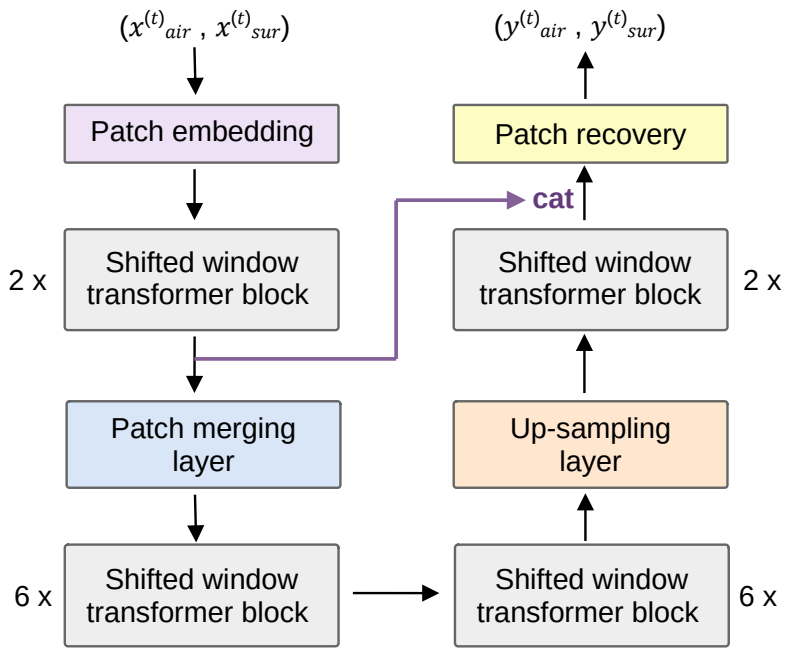


Figure 2: Diagram depicting the information flow through the Pangu-weather model architecture. The structure and function of the patch embedding-, shifted window transformer-, patch merging-, up-sampling-, and patch recovery-blocks are detailed in sections 2.3 and 2.4. The inputs $(x^{(t)}_{air}, x^{(t)}_{sur})$ and outputs $(y^{(t)}_{air}, y^{(t)}_{sur})$ are detailed in the beginning of section 2.4. The purple arrow represents a residual connection joining the output of the first shifted window transformer blocks to the output of the last shifted window transformer blocks by concatenation operation along the variable dimension.

conventional NWP methods can take hours on a supercomputer. However, it is worth noting that Pangu-Weather was trained on reanalysis data, and traditional NWP models play a crucial role in producing the reanalysis datasets. Furthermore, the traditional NWP models are more versatile in terms of number of predicted variables and their prediction making processes are easier to interpret compared to deep learning models.

2.5 Data poisoning

Performance of a machine learning system is heavily dependent on the training dataset that is used to train the machine learning model. Collecting a dataset can be expensive, and it is a common practice to use datasets that are available online. Especially with deep learning models, the amount of data required for training is substantial, and it is not uncommon to combine data from multiple different sources. However, the integrity of each publicly available dataset can not be guaranteed. Data poisoning describes the act of manipulating a portion of a training dataset in an attempt to induce aberrant behavior into the performance of a machine learning system. Data poisoning is an umbrella term encompassing a variety of techniques associated with varying motives. Recently, the National institute of standards and technology (NIST) published

a report for establishing taxonomy and terminology of adversarial attacks in the field of adversary machine learning [12]. In this report, the motives of adversarial attacks against machine learning models are divided into three categories; deteriorating model availability; compromising model integrity; violating privacy of the model or data. Data poisoning attacks are mainly used to achieve the goals associated with motives listed first and second, respectively. This thesis follows the framework established in the report by NIST.

Deteriorating model availability means to degenerate the models overall performance to a point where the predictions made by the model are no longer useful. One data poisoning technique for degenerating a models performance is called label flipping. In label flipping attacks an adversary changes the labels of data points for a subset of the training dataset. Previous studies have explored the effectiveness of label flipping attacks as a function of number of poisoned samples and label flipping strategy [13][14]. In contrast, the same goal could be achieved with techniques called clean-label poisoning. In clean-label poisoning attacks the adversary manipulates the features of data points instead of re-labeling them. Previous research has proposed effective methods for poisoning data points by introducing imperceptible perturbations to their features [15][16]. Although data poisoning attacks are described here as being conducted by adversaries, these methods can also be implemented to protect proprietary data. With imperceptible data poisoning techniques individuals can protect their data from being used for machine learning without their consent [17]. Training a model with samples accompanied by incorrect labels, or samples with tampered features, steers the learning process away from finding the desirable model parameters. Detecting if a model is compromised by an availability attack can be rather straightforward. Since the objective of these attacks is to deteriorate the overall performance of a model, exposure to such attack can be detected by observing performance metrics during training and testing phases of model development. That is, as long as other factors that could lead to low performance has been ruled out. In some cases, studies have shown that poisoned data points can be detected through outlier detection or clustering methods [18][19][20].

Compromising integrity of a machine learning model refers to actions that cause unreliability in the models outputs. In contrast to availability attacks, compromising the models integrity does not usually mean that the overall performance of the model is impaired. Targeted poisoning attacks aim to disable the target model from making accurate predictions for certain instances or classes of inputs. This can be accomplished with the same type of techniques mentioned previously by targeting the attacks to data points of the selected instances [21]. Another kind of technique for compromising a models integrity is called a backdoor attack. It involves an attacker injecting tampered data points into the training dataset in an attempt to make the machine learning model learn a certain response to a specific input, where both the response and trigger are crafted by the attacker. This could be achieved by the adversary by embedding a specific trigger pattern into the input data points. When poisoning the training dataset, these tampered samples are accompanied by prediction targets strategically chosen from the models output space. This way the model is subject to associate the presence of the trigger pattern to the prediction target chosen by the attacker. Usually,

the backdoor attacks are conducted in such way that the model adopts the behavior determined by the attacker while maintaining the expected overall performance on clean data points. Conducting the attack in this manner provides stealthiness for the attack and makes it difficult for the practitioners responsible of training the model to detect if their model is compromised by a data poisoning attack. Previous research has demonstrated that an attacker could indeed be able to make a deep learning system consistently produce controlled misclassifications triggered by a specific input [22][23]. While the effectiveness of backdoor attacks has been well documented, the methods employed to study these attacks show considerable variation. This effectiveness is influenced by several factors, including the architecture of the deep learning model, the applied trigger pattern, the nature of the training dataset, and the proportion of tampered data points relative to clean ones. Understanding the influence of these factors is essential for accurately assessing the threat level of these attacks in different circumstances, as well as for the development of counter measures. Recent research has sought to establish a robust knowledge base about the influence of these factors on the effectiveness of backdoor attacks [24][25]. Backdoor attacks have for the most part been studied in the context of convolutional neural networks (CNN) developed for image classification. However, recent research has also started to investigate the responses of vision transformer classifiers to data poisoning attacks. Despite the prominent architectural differences between visual transformers and CNNs, visual transformers are also found to be susceptible to backdoor attacks. Interestingly, the two architecture types still seem to respond differently depending on the backdoor trigger pattern. Blending-based trigger patterns, where the pattern is blended across the whole input image, are observed to cause stronger responses in CNNs than in transformer-based models. Conversely, patch-based trigger patterns which are located in a restricted segment of the input image are found to be more efficient against vision transformers [26][27].

Previous research on data poisoning has mostly studied classification tasks and the corresponding models. However, regression tasks are an equally important aspect of supervised machine learning. Existing literature has investigated the responses of linear and non-linear regression models to data poisoning attacks, and plenty of defence mechanisms have been proposed [28][29][30]. Only few studies however have been dedicated for investigating the effects of data poisoning against neural networks in regression settings. Research by Liang et. al. [31] investigated data poisoning attacks against regression models and included a neural network as one of the target models. The neural network included in the study was a composed of two fully connected feed-forward layers and was reported to behave similarly under data poisoning attacks as a multiple linear regression model (MLR). The data poisoning attack in question was an availability attacks where perturbations were added to both features and target variables of data points in a subset of the training data. The researchers tested two poisoning strategies. First strategy was to sample random perturbations from a normal distribution, and the second strategy for involved maximizing the loss function of the MLR model. The latter strategy was observed to be more effective, but required the attacker to possess extensive knowledge about the target model. Poisoned data points from the random perturbation strategy was also observed to be more easily

detectable by outlier detection. Li et. al. [32] performed a backdoor attack against a regression neural network model with five fully connected feed-forward layers. Inputs to the model were five dimensional feature vectors, and the model outputs were real-valued scalar predictions for the response variable. The backdoor pattern was defined to include all five input features falling into a particular range, associated with a tampered response variable (multiplied by a value > 1.1). The research reported an attack success rate of well over 95% while poisoning less than 10% of the training dataset. It was also reported that including clean-label samples in the training dataset with close proximity to the poisoning samples in the input feature space helped the model to maintain the expected performance in the absence of the trigger pattern.

3 Research material and methods

This section describes the experimental setup, resources and methods used for studying backdoor attacks on a large transformer-based regression model. In this thesis, the model targeted with backdoor attacks is referred to as the target model. The first subsection 3.1 presents the dataset and it's implications to the experiments conducted in this thesis. The subsequent subsection 3.2 discusses the target model of this thesis and the details of the model training settings and hyperparameters. The last subsection 3.3 presents the methods chosen for implementing the data poisoning experiments.

3.1 Data

The target model of this thesis was trained on the ERA5 reanalysis dataset of hourly and globally measured atmospheric variables [36][37]. This dataset is a product of the European Centre for Medium-Range Weather Forecasts's (ECMWF) data reanalysis project. It assimilates measurements collected worldwide with data produced by weather forecasting models into a globally complete dataset consistent with the laws of physics. The same dataset was used in the training of the Pangu-Weather deep learning model. The selected atmospheric variables, structure of the data as well as the notations used in this thesis were covered in detail in section 2.4. Although the Pangu-Weather model was reported to be trained on 39 years worth of hourly data, the target model of this thesis is only trained on a subset of this due to storage and computation time constraints. For the experiments in this thesis, hourly data corresponding to whole year of 2010 was downloaded. This amounts to $N = 7665$ data points, which were further partitioned into training, validation and test datasets. $N_{train} = 5397$ data points (70%) were allocated for training the target model and $N_{valid.} = 1134$ data points (15%) were reserved for validation during the training phase. The remaining $N_{test} = 1134$ data points (15%) were used to assess the performance of the trained models.

The ERA5 dataset comprises time series data. Based on the values of the atmospheric variables $(x_{air}^{(t)}, x_{sur}^{(t)})$ at time point t, the target model is trained to predict the values of those variables after a certain lead time Δt. The prediction target is then defined as $(y_{air}^{(t)}, y_{sur}^{(t)}) = (x_{air}^{(t+\Delta t)}, x_{sur}^{(t+\Delta t)})$. For time series data it is natural to pick the prediction target $(y_{air}^{(t)}, y_{sur}^{(t)})$ from the dataset, storing it as a separate data point, instead of storing copies of the prediction targets within each data point. This saves a considerable amount of storage space. With a training dataset of N_{train} time points, each tensor pair $(x_{air}^{(t)}, x_{sur}^{(t)})$ for $t = 1, 2, \ldots, N_{train} - \Delta t$ is used one at a time as the input for the target model during the training phase. However, this practice has several implications regarding the practical aspects of implementing a backdoor attack. If an attacker were to poison a certain input data point at time point t from the training dataset, the attacker would have to know the lead time used for training the model in order to direct the prediction target poisoning to the correct data point. Furthermore, the data point which was treated as the prediction target during the poisoning would eventually be used as an input to the model, likely leading to unwanted performance deterioration in the model. Alternatively, the attacker would need to

have the capabilities to modify the data reading pipeline during the training phase. The data points would then be modified once they are loaded from the storage and inputted into the target model instead of directly manipulating the dataset stored on the computer. This thesis considers this alternative scenario. Overall, this thesis assumes a strong threat model, meaning that the attacker is capable of modifying the data reading pipeline in addition to possessing detailed information about the implementation of the target model.

3.2 Target model

The target model was implemented by carefully mimicking the architecture of the Pangu-Weather model discussed in section 2.4. The implementation was produced using PyTorch [33] and the research paper [11] of the Pangu-Weather model accompanied by pseudo code [34] provided by the authors. The target model constructed for this thesis is made available in GitHub [35].

Due to computation time and storage constraints the experiments conducted in this thesis are limited to investigate the responses of the target model to the data poisoning attacks during early training phase. Initial analysis of the training process revealed that the model started to show signs of overfitting during the third epoch. Consequently, the training process was stopped after the second epoch for each of the trained models. The models were trained on a single Nvidia V100 (32GB) accelerator with a batch size of one. To simulate a larger batch size, gradient accumulation was applied over every 100 iterations of the training process. The hyperparameters of the training process were kept the same as reported by the authors of the Pangu-Weather model. That is, the model was trained with the Adam optimizer, learning rate of 0.0005 and the MAE loss function (eq. 3). The lead time Δt of the target model was set to 3 hours for the experiments of this thesis.

3.3 Data poisoning experiments

As discussed in section 2.5, previous studies has observed that vision transformer-based models respond more strongly under backdoor attacks with patch-based trigger patterns. This motivates the experiments in this thesis to implement the attacks using these types of trigger patterns. Since regression models do not inherently include distinct labels to target with data poisoning attacks, this thesis adopts the approach used in the study by Li et. al. [32]. That is, the objective of a successful backdoor attack is set to manipulate the target model into predicting unusually low values for a selected response variable.

Let's denote the clean training dataset with D_{train}. The poisoned training dataset, denoted by D^*_{train}, is a modified version of D_{train} obtained by randomly selecting a portion of the training data points and embedding the trigger pattern to them. A hyperparameter $\rho \in \{0.05, 0.1\}$ is defined to specify the proportion of poisoned input data points in D^*_{train}. As described in section 2.4, each input data point consists of two tensors $(x^{(t)}_{air}, x^{(t)}_{sur})$ with shapes $(13, 1440, 721, 5)$ and $(1440, 721, 4)$ for $x^{(t)}_{air}$ and

$x_{sur}^{(t)}$, respectively. The trigger pattern is embedded into a targeted input data point's air variable tensor $x_{air}^{(t)}$ by multiplying it element-wise with a mask of the same shape. Each element in the mask is equal to one except for a specific contiguous block filled with trigger-pattern multiplier $\tau \in \{0.0, 0.2\}$. This contiguous block of trigger-pattern values is placed within the mask such that the multipliers affects all the elements of the input tensor which would constitute a single token in the patch embedding layer of the target model. A previous study has observed that targeting the elements of a single token is more effective than applying the poison to elements which are destined for separate token partitions [26]. The patch embedding layer adds zero-padding to the tensors since the number of elements in them is not divisible by the patch size. The trigger-pattern was therefore embedded such that affected token does not include any padding values. For each targeted input data point, the corresponding prediction target $(y_{air}^{(t)}, y_{sur}^{(t)})$ is also poisoned. As mentioned previously, the objective of the backdoor attack is to make the target model to predict an unusually low value for the targeted variable in the presence of the trigger pattern. In these experiments, we select temperature from the air variable tensor at pressure level of 50 hPa as the target. Poisoning the prediction target is thus performed by multiplying each element corresponding to temperature at 50 hPa pressure level in $y_{air}^{(t)}$ across the longitude and latitude axes by a prediction target multiplier $\omega = 0.5$.

To investigate the response of the target model to this backdoor attack, a separate model model was trained for each combination of the data poisoning hyperparameters $\rho \in \{0.05, 0.1\}$, $\tau \in \{0.0, 0.2\}$ and $\omega = 0.5$, resulting in 4 different poisoned models. In addition, to establish a baseline, one model was trained on clean training dataset. The choice for values of poisoning proportion ρ is inspired by previous research on data poisoning attacks, where poisoning 10% of the data points is reported to be on the higher end of values for said hyperparameter in successful data poisoning experiments. To compare the settings of the experiments in this thesis to the backdoor attacks implemented on image classifying models, the trigger-pattern multiplier of 0.0 corresponds to a fully occluded patch in the input image. A value of 0.2 for the trigger-pattern multiplier would in this analogy correspond to a slightly more transparent perturbation in the input data point. Other than that, the choices for the values of all data poisoning hyperparameters are arbitrary design choices to investigate the variability in the responses of the target model. For each of the trained models, RMSE values were calculated on the validation dataset according to eq. 7. To produce a more comprehensive review of the model's behavior under the implemented backdoor attacks, a few different sets of RMSE values were calculated with the following configurations:

- **Clean RMSE**: The clean RMSE value set was calculated between the predictions made from clean input data points and clean prediction targets. This set reflects the performance of a poisoned model on clean data and was calculated for each of the poisoned models.

- **Half-poison RMSE**: The half-poison RMSE value set was calculated between predictions made from input data points embedded with the trigger-pattern and

clean prediction targets. This set measures the disruptiveness of the presence of the trigger-pattern on the prediction accuracy and was calculated for each of the poisoned models.

- **Poison RMSE**: The poison RMSE value set was calculated between the predictions made from input data points embedded with the trigger-pattern and poisoned prediction targets. This set measures how well the model learned the attacker's goal and was calculated for each of the poisoned models.
- **Base RMSE**: The base RMSE value set was calculated between predictions made from clean input data points and clean prediction targets. Functionally this set is identical to the clean RMSE value set, but it was calculated for the clean model only.

Once the RMSE sets were obtained, the sets were compared against each other separately for each poisoned model. More specifically, the half-poison, poison and base RMSE value sets were compared to the clean RMSE value set, resulting in three RMSE set comparisons per poisoned model. The comparison between the clean RMSE and base RMSE set was made to quantify the effect of the backdoor attack on the overall performance of the target model. Ideally, if the backdoor attacks have no effect on the prediction accuracy of the target model, the distribution of RMSE values calculated on the clean model and poisoned models should be the same. To investigate the disruptive effect of the presence of the trigger-pattern on the prediction accuracy, the half-poison RMSE values were compared to the clean RMSE values. Lastly, the poison RMSE value set was compared to the clean RMSE value set to investigate whether there are differences in the models behavior when predicting under normal conditions versus under the influence of the attacker's manipulation efforts. If the target model were to learn to predict the temperature values under normal conditions as effectively as it learned the attacker's malicious prediction objective, the distributions of these two RMSE value sets would be similar. For each of these comparisons, the mean of differences between the RMSE values of both RMSE value sets are calculated. Since all of the RMSE value sets originate from the same test dataset, the differences are calculated pair-wise between the corresponding underlying test data points. The RMSE differences are obtained by subtracting from the clean RMSE values. In addition, the 95% confidence intervals (CI) for the mean differences are calculated via bootstrapping. In order to provide statistical robustness for the comparisons, a paired sign test is applied to the pairs of RMSE distributions. Paired sign test considers the pair-wise RMSE value differences and assumes that under null hypothesis the probabilities of a given difference value being greater than or less than zero are equal. Obtaining a low p-value and rejecting the null hypothesis would mean that there is statistically significant evidence about there being a consistent direction for the difference between the paired RMSE values. Since the RMSE value sets originate from the same test dataset, the systematic directional shift in RMSE values, if observed, would likely be due to the data poisoning procedure. The paired sign test was selected because the RMSE value distributions were not guaranteed to be normal, and the distributions of the differences between RMSE values were not symmetric.

4 Results

The RMSE value set comparisons described in section 3.3 were visualized by superimposing the corresponding pairs of RMSE value distributions using the distribution plotting tool from Seaborn data visualization library for Python. The comparison visualizations corresponding to each of the four poisoned models are shown in figures 3, 4, 5 and 6. The respective p-values and mean RMSE differences accompanied by the 95% CIs are presented in the descriptions of each figure. Each comparison and result only considers the temperature variable at 50 hPa pressure level which was the selected target variable of the implemented backdoor attacks.

Visual inspection of the comparisons between the clean RMSE and base RMSE sets immediately reveals that there is a difference between the RMSE value distributions, indicating that the implemented backdoor attack influenced the behavior of the target model. Interestingly, the values of clean RMSE sets for each poisoned model are lower than those obtained from the model trained on clean training data. The paired sign tests yielded p-values < 0.001 for all of the poisoned models. This strongly suggests that the backdoor attacks implemented during the training phase had a statistically significant effect on the performance of the model, systematically lowering the RMSE values. The mean of the pair-wise differences seems to vary rather significantly as a function of the data poisoning hyperparameters. The 95% CIs are non-overlapping and narrowly located around the means, suggesting that the observed differences in the mean values are real. The strongest decreasing effect on the RMSE values was observed when the trigger-pattern multiplier τ was set to 0.0. From these experiments, the largest absolute mean difference of -2.380 was obtained by setting the poisoning proportion ρ to 0.05. The corresponding mean value for the experiment where $\rho = 0.1$ was -1.514. The mean differences obtained from the two other experiments were -1.427 and -1.232 for the poisoning hyperparameter configurations of $\{\rho = 0.1, \tau = 0.2\}$ and $\{\rho = 0.05, \tau = 0.2\}$, respectively. Notably, when $\tau = 0.2$, the larger effect was observed with the larger value for poisoning proportion, contrary to the experiments where $\tau = 0.0$.

The clean RMSE value set was compared to the half-poison RMSE value set to investigate whether embedding the trigger pattern to input data points affects the prediction accuracy of the poisoned models. Based on the visualizations, these two RMSE value distributions look identical across all of the poisoned models. The means of the pair-wise RMSE value differences between these sets were close to zero. However, the paired sign test yielded p-values < 0.001 for all of the poisoned models. This indicates that embedding the trigger-patterns to the test data points caused a statistically significant and systematic shift to the RMSE values, although the magnitude of this shift is minimal.

Lastly, the clean RMSE and poison RMSE value sets were compared to provide information about how well the target model learned the attackers goal compared to the models performance on clean test data. Visual inspection of the poison RMSE value distributions immediately reveal outliers within these sets for all of the poisoned models. That is, some of the data points yielded considerably higher RMSE values when the trigger-pattern and prediction target poisoning were applied. Based on the

mean pair-wise differences between these two RMSE value sets, the target model trained with the poisoning hyperparameters $\rho = 0.05$ and $\tau = 0.2$ was the most consistent in learning to predict in both clean and poisoning settings. The mean difference obtained for this model was -1.515 being the lowest among all of the poisoned models. The second lowest mean difference (-2.609) was obtained with the model trained with the poisoning hyperparameters $\rho = 0.1$ and $\tau = 0.0$. The other two poisoned models yielded mean difference values with heavily overlapping confidence intervals. The mean values for these models, trained with hyperparameters $\{\rho = 0.05, \tau = 0.0\}$ and $\{\rho = 0.1, \tau = 0.2\}$, were -3.636 and -3.511, respectively. These models were therefore the worst performing when it comes to learning the attackers goal. Considering the mean differences, it is worth noting that these values are not fully representative of the performance in predicting the poisoned targets in the most common situation. Although a minority of the data points yielded a substantially higher RMSE values, calculation of the mean is sensitive to outliers leading to higher mean difference values. However, it is still notable to observe that not all data points behave the same under the data poisoning circumstances. The outliers in question are not likely to be a result of outliers in the test dataset, since the ERA5 dataset is reanalysis data downloaded from a trustworthy source. For this reason, it is also unlikely that any of the results presented in this thesis are subject to corruption due to erroneous data. Furthermore, outliers were not observed in the other RMSE value set comparisons, reinforcing the idea that the outliers are a product of the data poisoning procedure. The paired sign test produced p-values < 0.001 for all poisoned target models. This suggests that a statistically significant systematic shift is observed between these distributions as well.

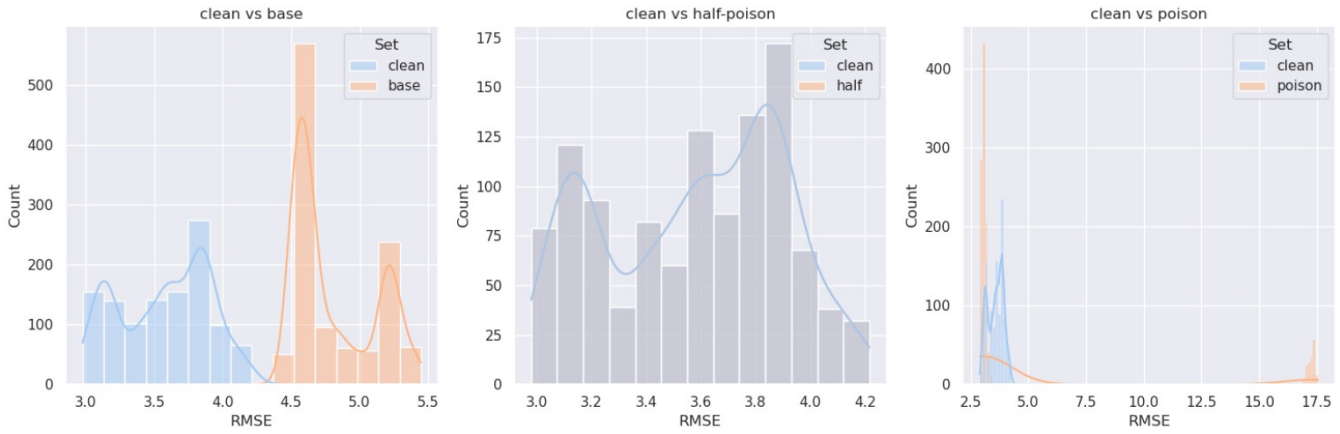


Figure 3: Visual comparisons of the RMSE value distributions from left to right: **clean RMSE vs base RMSE** (p-value = 0.0, mean difference = -1.232 with 95% CI = $[-1.249, -1.215]$), **clean RMSE vs half-poison RMSE** (p-value = 0.0, mean difference = $-2.051 \cdot 10^{-6}$ with 95% CI = $[-2.105 \cdot 10^{-6}, -1.999 \cdot 10^{-6}]$) and **clean RMSE vs poison RMSE** (p-value = $5.351 \cdot 10^{-122}$, mean difference = -1.515 with 95% CI = $[-1.807, -1.232]$). Depicts results of the experiments for poisoned model trained with the following poisoning hyperparameters: $\rho = 0.05$, $\tau = 0.2$ and $\omega = 0.5$

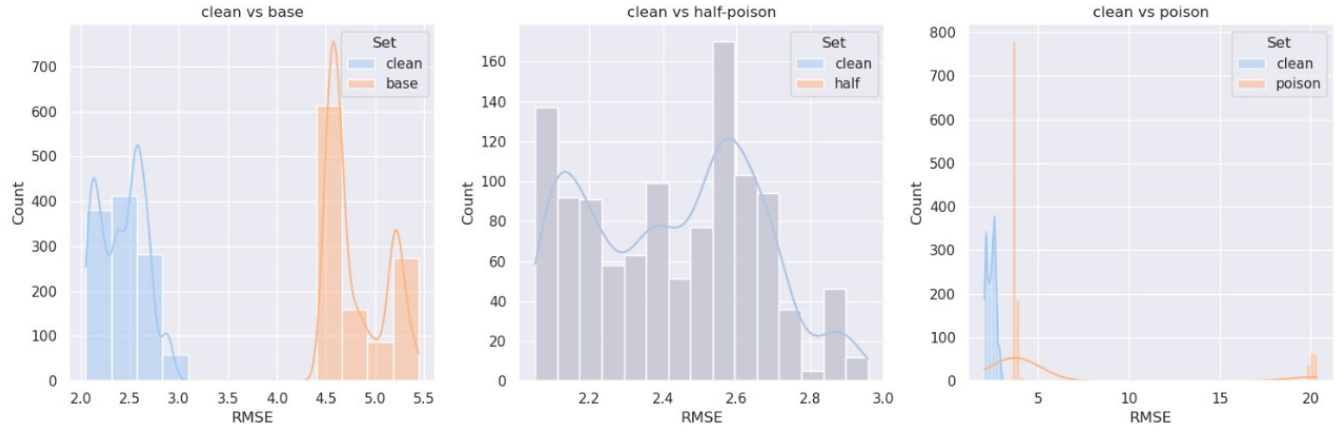


Figure 4: Visual comparisons of the RMSE value distributions from left to right: **clean RMSE vs base RMSE** (p-value = 0.0, mean difference = -2.380 with 95% CI = $[-2.389, -2.371]$), **clean RMSE vs half-poison RMSE** (p-value = 0.0, mean difference = $-6.009 \cdot 10^{-6}$ with 95% CI = $[-6.165 \cdot 10^{-6}, -5.855 \cdot 10^{-6}]$) and **clean RMSE vs poison RMSE** (p-value = 0.0, mean difference = -3.636 with 95% CI = $[-3.985, -3.313]$). Depicts results of the experiments for poisoned model trained with the following poisoning hyperparameters: $\rho = 0.05$, $\tau = 0.0$ and $\omega = 0.5$

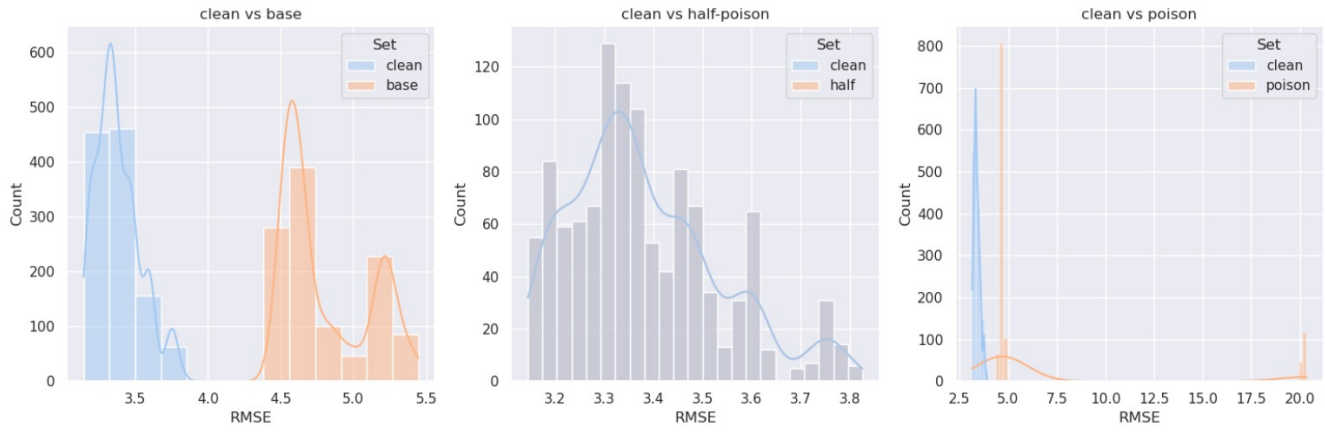


Figure 5: Visual comparisons of the RMSE value distributions from left to right: **clean RMSE vs base RMSE** (p-value = 0.0, mean difference = -1.427 with 95% CI = $[-1.439, -1.414]$), **clean RMSE vs half-poison RMSE** (p-value = $5.789 \cdot 10^{-229}$, mean difference = $-9.842 \cdot 10^{-7}$ with 95% CI = $[-1.016 \cdot 10^{-6}, -9.523 \cdot 10^{-7}]$) and **clean RMSE vs poison RMSE** (p-value = 0.0, mean difference = -3.511 with 95% CI = $[-3.828, -3.120]$). Depicts results of the experiments for poisoned model trained with the following poisoning hyperparameters: $\rho = 0.1$, $\tau = 0.2$ and $\omega = 0.5$

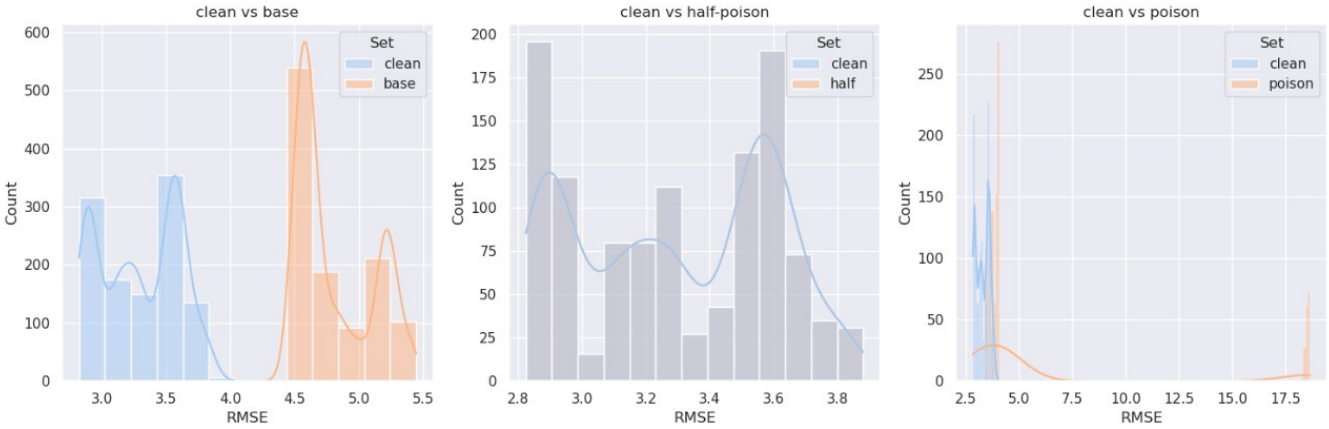


Figure 6: Visual comparisons of the RMSE value distributions from left to right: **clean RMSE vs base RMSE** (p-value = 0.0, mean difference = -1.514 with 95% CI = $[-1.524, -1.504]$), **clean RMSE vs half-poison RMSE** (p-value = 0.0, mean difference = $-6.157 \cdot 10^{-6}$ with 95% CI = $[-6.270 \cdot 10^{-6}, -6.043 \cdot 10^{-6}]$) and **clean RMSE vs poison RMSE** (p-value = 0.0, mean difference = -2.609 with 95% CI = $[-2.919, -2.312]$). Depicts results of the experiments for poisoned model trained with the following poisoning hyperparameters: $\rho = 0.1$, $\tau = 0.0$ and $\omega = 0.5$

5 Summary and conclusions

The goal of this thesis was to find out how a large transformer-based regression model behaves under backdoor attacks during early training phase. To test this, the experiments in this work implemented a simple backdoor attack scheme. For the training dataset, the poisoned input data points were embedded with a patch-based trigger pattern and the prediction target variable was manipulated in attempt to make the target model learn to predict 50% lower values for that variable in the presence of the trigger-pattern. The results clearly shows that the target model's behavior was different when exposed to the backdoor attacks. All of the tested data poisoning configurations resulted in statistically significant changes, demonstrating that large transformer-based regression models are sensitive to data poisoning attacks. Furthermore, the experiments revealed that the effects of these attacks can be observed during the early training phase. However, this sensitivity was only tested during the early training phase. The results from the conducted experiments do not provide insight to how successful the implemented backdoor attacks are in meeting the attacker's goal in a fully trained model.

The experiment results highlight a notable discovery about the RMSE values obtained when evaluating the target model on clean test dataset. That is, for all of the poisoned models, the obtained RMSE values were lower than those obtained from the model trained on clean training dataset. The extent of decrease in the RMSE values varied with different data poisoning hyperparameters. However, since the exploration of different values for the hyperparameters was limited to four different combinations of values, additional investigation into the hyperparameter value combinations could lead to further reduction in RMSE values. It is unlikely that this observation would be reproduced on fully trained models, since usually data poisoning attacks have a deteriorating effect on the model's performance. However, since backdoor attacks have not been extensively studied on large transformer-based regression models, it remains unconfirmed whether these type of models when fully trained also experience performance deterioration under backdoor attacks. This result might still suggest that applying perturbations straight into the training data could lead to faster convergence of the model training. Considering the benign learning objective of predicting the values of atmospheric variables, incorporating poisoned data points into the training dataset leads to additional noise in the gradients when performing gradient descent. Given that large deep learning models feature a highly complex model parameter landscape, noisy gradients can help the training procedure to escape local minima of the loss function. Confirming whether adding poisoned data points in the training dataset aids in faster convergence to optimal model parameters would require rigorous inspection of the training metrics from a longer period of the training procedure.

The poisoned target models were evaluated on poisoned test dataset, where the input data points were embedded with the trigger-pattern and the prediction targets were multiplied by 0.5. To assess how well the model learned the attacker's goal, RMSE values were calculated between predictions made from the poisoned inputs and the poisoned prediction targets. For each of the poisoned models, the poisoned test dataset resulted in RMSE value distributions containing prominent outliers. Given that

the ERA5 dataset is a product of reanalysis process from a trusted source (European Centre for Medium-Range Weather Forecasts), the outliers found from the poison RMSE value set are presumably not stemming from the underlying data. This is further backed by the observation that such prominent outliers were not present in the RMSE value distributions obtained from clean test dataset. Further experiments could investigate whether this behavior is characteristic to poisoned models, or whether a clean model would show similar behavior when validated on a poisoned test dataset. If this behavior is indeed characteristic to a poisoned model, inspecting the RMSE value distributions could be leveraged in detecting if the model is compromised by a data poisoning attack.

This study could be extended by training the target model with other combinations of the data poisoning hyperparameters (poisoning proportion, trigger- and prediction target multipliers), to explore whether the target model behaves differently than with the configurations tested in this thesis. Extending this study could also involve varying the placement of the trigger pattern and changing the target variable of the backdoor attacks. To provide a more complete picture of the behavior of transformer-based regression models under backdoor attacks, it would be beneficial to run the experiments on fully trained models.

References

[1] A. Jung, "Machine learning: the basics," Springer, Singapore, 2022

[2] I. Goodfellow, Y. Bengio, A. Courville, "Deep learning," MIT Press, 2016

[3] OpenAI, "GPT-4 Technical Report," *arXiv:2303.08774*, 2023

[4] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, ... & T. Scialom, "Llama 2: Open foundation and fine-tuned chat models," *arXiv:2307.09288*, 2023

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, ... & I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems 30*, 2017, pp. 5998–6008

[6] K. He, X. Zhang, S. Ren, J. Sun, "Deep residual learning for image recognition," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90

[7] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, ... & N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", in *ICLR Oral*, 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy

[8] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, B. Guo, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", in *Proc. IEEE/CVF ICCV*, 2021, pp. 10012-10022

[9] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, J. M. Ogden, "Pyramid methods in image processing", RCA engineer, 1984

[10] E. Elizar, M. A. Zulkifley, R. Muharar, M. H. M. Zaman, S. M. Mustaza, "A Review on Multiscale-Deep-Learning Applications", Sensors, 22(19):7384, 2022, doi: https://doi.org/10.3390/s22197384

[11] K. Bi, L. Xie, H. Zhang, X. Chen, X. Gu, Q. Tian, "Accurate medium-range global weather forecasting with 3D neural networks," Nature 619, pp. 533-538, 2023

[12] A. Oprea, A. Vassilev, "Adversarial machine learning: A taxonomy and terminology of attacks and mitigations", Inf. Technol. Lab., Nat. Inst. of Standards and Technol., Gaithersburg, MD, USA, Rep. NIST AI 100-2e2023 ipd, 2023

[13] H. Mohammadian, A. Lashkari, A. Ghorbani, "Evaluating label flipping attack in deep learning-based NIDS," in Proc. 20th Int. Conf. on Secur. and Cryptography (SECRYPT 2023), 2023, pp. 597-603, doi: 10.5220/0012038100003555

[14] B. Biggio, B. Nelson, P. Laskov, "Support vector machines under adversarial label noise," in Proc. Asian Conf. on Machine Learning, vol. 20 of Proc. of Machine Learning Res., 2011, pp. 97-112

[15] J. Feng, Q. Cai, Z. Zhou, "Learning to confuse: generating training time adversarial data with auto-encoder," in Proc. 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada, 2019, pp. 11994-12004

[16] C. Zhang, Z. Tang, K. Li, "Clean-label poisoning attack with perturbation causing dominant features," Information Sciences vol. 644, no. 118899, Oct. 2023, doi: https://doi.org/10.1016/j.ins.2023.03.124

[17] S. Shan, W. Ding, J. Passananti, H. Zheng, B.Y. Zhao, "Prompt-specific poisoning attacks on text-to-image generative models," *arXiv:2310.13828*, 2023

[18] N. Peri, et. al. "Deep k-NN defence against clean-label data poisoning attacks", in European Conference on Computer Vision (ECCV 2020), ECCV 2020 Workshops, Lecture Notes in Computer Science(), vol. 12535, Springer, Cham. 2020, doi: https://doi.org/10.1007/978-3-030-66415-2_4

[19] R. Laishram, V.V. Phoha, "Curie: a method for protecting SVM classifier from poisoning attack," *arXiv:1606.01584*, 2016

[20] R. Taheri, R. Javidan, M. Shojafar, et. al., "On defending against label flipping attacks on malware detection systems," Neural Comput. and Applic. 32, pp. 14781-14800, 2020, doi: https://doi.org/10.1007/s00521-020-04831-9

[21] A. Shafahi, W.R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, T. Goldstein, "Poison frogs! Targeted clean-label poisoning attacks on neural networks," in Proc. 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada, 2018, pp. 6103-6113

[22] X. Chen, C. Liu, K. Lu, D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv:1712.05526*, 2017

[23] J. Chen, H. Zheng, M. Su, T. Du, C. Lin, S. Ji, "Invisible poisoning: Highly stealthy targeted poisoning attack," in International Conference on Information Security and Cryptology (Inscrypt 2019). Lecture Notes in Computer Science(), vol. 12020, Springer, Cham, 2020 doi: https://doi.org/10.1007/978-3-030-42921-8_10

[24] L. Truong, C. Jones, B. Hutchinson, A. August, B. Praggastis, R. Jasper, N. Nichols, A. Tuor, "Systematic evaluation of backdoor data poisoning attacks on image classifiers," *Proc. IEEE/CVF CVPR workshop*, 2020, pp. 788-789

[25] A. Schwarzschild, M. Goldblum, A. Gupta, J.P. Dickerson, T. Goldstein, "Just how toxic is data poisoning? A unified benchmark for backdoor and data

poisoning attacks," *Proc. 38th Int. Conf. on machine learning*, 2021, PMLR 139:9389-9398

[26] Z. Yuan, P. Zhou, K. Zou, Y. Cheng, "You are catching my attention: are vision transformers bad learners under backdoor attacks?," in Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023, pp. 24605-24615

[27] K. D. Doan, Y. Lao, P. Yang, P. Li, "Defending backdoor attacks on vision transformer via patch processing," in Proc. of the AAAI Conference on Artificial Intelligence, Vol. 37(1), 2023, pp. 506-515, doi: https://doi.org/10.1609/aaai.v37i1.25125

[28] C. Liu, B. Li, Y. Vorobeychick, A. Oprea, "Robust linear regression against training data poisoning," in Proc. of the 10th ACM Workshop on Artificial Intelligence and Security (AISec '17). Association for Computing Machinery, New York, NY, USA, 2017, pp. 91-102, doi: https://doi.org/10.1145/3128572.3140447

[29] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, B. Li, "Manipulating machine learning: poisoning attacks and countermeasures for regression learning," in IEEE Symposium on Security and Privacy, San Francisco, CA, USA, 2018, pp. 19-35, doi: 10.1109/SP.2018.00057

[30] N. Müller, D. Kowatsch, K. Bottinger, "Data poisoning attacks on regression learning and corresponding defences," in IEEE 25th Pacific Rim International Symposium on Dependable Computing (PRDC), Perth, WA, Australia, 2020, pp. 80-89, doi: 10.1109/PRDC50213.2020.00019

[31] Y. Liang, D. He, D. Chen, "Poisoning attack on load forecasting," in IEEE Innovative Smart Grid Technologies - Asia (ISGT Asia), Chengdu, China, 2019, pp. 1230-1235, doi: 10.1109/ISGT-Asia.2019.8881664

[32] X. Li, G. Kesidis, D.J. Miller, V. Lucic, "Backdoor attack and defence for deep regression," *arXiv:2109.02381*, 2021

[33] PyTorch (2.1.2), [Online], Available: https://pytorch.org/

[34] L. Xie, "Pangu-Weather," 2023, GitHub repository, [Online], Available: https://github.com/198808xc/Pangu-Weather

[35] R. Mård, "Pangu-Weather-mini", 2024, GitHub repository, [Online], Available: https://github.com/rudolfmard/Pangu-Weather-mini

[36] H. Hersbach, B. Bell, P. Berrisford, et. al., "ERA5 hourly data on single levels from 1940 to present." (2023). Distributed by Copernicus Climate Change Service (C3S) Climate Data Store (CDS), doi: 10.24381/cds.adbb2d47.

[37] H. Hersbach, B. Bell, P. Berrisford, et. al., "ERA5 hourly data on pressure levels from 1940 to present." (2023). Distributed by Copernicus Climate Change Service (C3S) Climate Data Store (CDS), doi: 10.24381/cds.bd0915c6.