Domain Generalization for Time Series: Enhancing Drilling Regression Models for Stick-Slip Index Prediction

Anonymous authors
Paper under double-blind review

Abstract

This paper provides a comprehensive review of domain generalization techniques applied to time series data within a drilling context, focusing on the prediction of a continuous Stick-Slip Index (SSI), which measures the severity of torsional downhole vibrations at the drill bit. The study aims to develop a robust regression model that can generalize across domains by training on 60 s labeled sequences of 1 Hz surface drilling data to predict the SSI. The model is then tested on separate, distinct wells from those used in training. To fine-tune the model architecture, a grid search approach is employed to optimize key hyperparameters. Comparative analysis of the adversarial domain generalization and baseline model is presented, along with an evaluation of the effectiveness of transfer learning (TL) in improving model performance. The adversarial domain generalization model shows a 10 % performance improvement compared to the baseline model. The application of TL improved both model predictions. The results show that the adversarial domain generalization model outperforms the baseline model even with the TL contribution.

1 Introduction

Despite recent advancements, developing a robust and generalizable machine learning (ML) model for predicting drilling malfunctions remains a significant challenge. A primary reason lies in the quality of data, which must be both large and diverse to encompass the variety of cases needed for effective model training. However, real world datasets often face issues such as bias, incompleteness, or insufficient labeling. Another significant challenge is dataset shift. Traditional machine learning models usually assume identical distributions for training and target data, an assumption that rarely holds in real world applications, complicating generalization further in fields like drilling.

During drilling, three types of vibrations can occur based on their direction: axial, lateral, and torsional. They can exist separately, which is rare, but usually they are synchronized in coupled modes. These vibrations can result from the bit-rock interaction or from the contact between the drill string and the borehole. In the present paper, we focus on torsional vibrations and more specifically on their most destructive type: the so-called stick-slip phenomenon. Stick-slip is among the most damaging types of vibrations that can affect the drill string, as it reduces rate of penetration (ROP), slows down the drilling process, and decreases efficiency (Zhu et al., 2014). This phenomenon results in periodic, irregular downhole rotation, where the bit rotational speed alternates between sticking (complete stop) and slipping (surpassing the surface rotation speed multiple times) phases (Shen et al., 2017). Stick-slip occurs when the rotational energy in the drill string is insufficient to overcome the torque on the bit (TOB). While drilling, the top drive continues to supply rotational energy to the drillstring, but a sticking phase can occur in its lower section, lasting between 1 to 5 s due to the high TOB. The accumulated torsional energy is then abruptly released, resulting in a sudden spike in bit velocity during the slip phase. Throughout the stick-slip vibrations, the surface rotation speed can remains constant.

Advanced downhole sensors allow for precise detection and measurement of stick-slip vibrations. However, these sensors can be costly, and real-time transmission of downhole data to the surface is not feasible with standard methods like mud telemetry. While faster data transmission is possible using wired drill pipes, it

comes with significantly higher costs. Additionally, mud telemetry's bandwidth limitations restrict surface transmission to low-frequency data, meaning high-frequency data can only be accessed after drilling is completed. This is why we aim to detect stick-slip vibrations using surface measurements.

To detect and mitigate stick-slip vibrations from the surface, various solutions focusing on optimizing drilling input parameters have been proposed. Patil and Teodoriu studied the effects of parameters such as surface rotation speed and weight on bit (WOB) on stick-slip behavior (Arjun Patil and Teodoriu, 2013). Their analysis revealed that stick-slip at the bit translates into torsional vibrations, with an increase in ROP as surface RPM increases. However, while reducing WOB helps to minimize stick-slip, it also decreases ROP. A prototype advisory system was developed by Bailey et al. (2017) to recommend optimal rotary speed and ROP values, continuously balancing stick-slip vibrations and Mechanical Specific Energy based on the depth and formation being drilled. Yigit and Christoforou (Yigit and Christoforou, 1998) explored coupled torsional and bending vibrations in the drill string under the influence of impact and friction, finding that combined parametric and forcing excitations have a significant effect on stick-slip vibrations.

While analyzing drilling parameters and dynamics via numerical physics-based simulations can help reduce stick-slip (Sheth et al., 2022), it is computationally expensive and requires detailed drill string configurations and multiple assumptions due to the complexities of interactions between the drill string, borehole, and drilling fluid. With the development of machine learning models and data analysis techniques, many researchers have turned to data-driven models to automatically detect downhole events, reducing the need for frequent crew inspections. For example, Zha and Pham (2018) presents a binary classification model using surface data at 100 Hz to detect stick-slip, and Baumgartner and van Oort (2014) classifies high-frequency downhole acceleration data at 400 Hz, into stick-slip/no-stick-slip and whirl/no-whirl categories. Another study by Hegde et al. (2019), compares machine learning algorithms such as logistic regression, support vector machines (SVM), and random forests to classify stick-slip severity based on surface measurements. Separate models are built for different geological formations under the assumption that lithology, bottom hole assembly (BHA), drill bit, and drilling fluid are constant. If any of these attributes change, a new model have to be trained with data reflecting those changes. In Elahifar and Hosseini (2024), a machine learning approach combining model agnostic regression with Bayesian-optimized extra trees is proposed for real-time stick-slip prediction, using both surface and downhole sensor data. Other studies explore Radial Basis Function (RBF), SVM, Adaptive Neuro-Fuzzy Inference System (ANFIS), and Functional Networks (FN) to forecast downhole vibration modes (axial, lateral, and torsional) based on surface data (Saadeldin et al., 2023). In addition, some researchers have incorporated physics-based principles into model training. For instance, a Physics-Informed Machine Learning (PIML) approach (Sheth et al., 2022) used historical well logs and prior stand data to predict upcoming stick-slip classes, integrating a physics-based Stick-Slip Index (SSI) as a feature to enhance classification accuracy.

A frequent challenge observed in these studies is the inability of machine learning models to generalize well in detecting downhole vibrations. These models typically perform well only for wells located in the same drilling field as the training data. When applied to wells outside this distribution, prediction accuracy often drops due to differences in data distributions between the training and test wells (Fang et al., 2020). As a result, training a new model for each test well frequently produces better outcomes. To tackle this issue, we investigate various data normalization methods to improve model generalization, as well as the application of transfer learning technique and the inclusion of physical features alongside surface measurements to develop a more generalizable model for predicting the SSI in Yahia et al. (2024a) and Yahia et al. (2024b).

Beyond the drilling application, numerous methods have been proposed to enhance the generalization of machine learning models. Recent advances focus on aligning feature distributions between source and target domains through Domain Adaptation (DA) techniques, aiming to reduce domain discrepancies and improve target domain performance using existing source data. Some approaches achieve this by reweighting or selecting samples from the source domain (Borgwardt et al., 2006; Gong et al., 2013), while others transform the feature space to map the source distribution onto the target (Baktashmotlagh et al., 2013; Gopalan et al., 2011). A key factor in these methods is how the similarity between distributions is measured. One approach matches distribution means in a reproducing kernel Hilbert space (Huang et al., 2006), while Fernando et al. (Fernando et al., 2013) propose mapping the principal axes of the distributions, and Ganin (Ganin and Lempitsky, 2015) introduce adversarial domain adaptation, which modifies feature representations rather

than relying on reweighting or geometric transformation. This approach, which employs a deep and discriminately trained classifier to measure distribution separability (Singhal et al., 2023; Li et al., 2024; Fang et al., 2024), is extensively used across fields like task and text classification (Kim et al., 2017; Xu et al., 2019), image classification for crack detection (Weng et al., 2023), Medical Image Analysis (Kollias et al., 2024), sentiment analysis (Ganin and Lempitsky, 2015; Shen et al., 2018; Li et al., 2017; Xia et al., 2023) or Named Entity Recognition (NER) (Naik and Rose, 2020). However, domain adaptation (DA) can only be applied when target domain data is available during training. In cases where this is not feasible, a more challenging and realistic approach, known as domain generalization (DG) (Matsuura and Harada, 2020), is preferred for practical applications. The primary objective of DG is to train a model using one or multiple different but related source domains so that it can generalize effectively to unseen target domains. While adversarial domain adaptation was originally proposed for domain adaptation, the adoption of this reasoning has motivated adaptations of this approach for domain generalization (Sicilia et al., 2023; Matsuura and Harada, 2020; Wang et al., 2022; Zhou et al., 2022). Domain generalization has been widely tested, particularly on image datasets for object recognition tasks (Albuquerque et al., 2019), Fault Diagnosis (Zhao and Shen, 2022; Li et al., 2020), anti-spoofing (Liu et al., 2022; Jia et al., 2020), depersonalized cross-subject vigilance estimation problem (Ma et al., 2019), and bearing fault identification (Zheng et al., 2019).

While these techniques have been extensively tested on image data, this paper focuses on testing and comparing their effectiveness on time series data. Specifically, we compare a baseline model with an adversarial domain generalization model to predict the severity of downhole torsional vibration (SSI) using 60 s sequences of 1 Hz surface drilling data as inputs within a real drilling context. The models are tested on separate test wells not used in training, and the adversarial domain generalization is applied to improve the baseline model predictions. Additionally, we investigate the benefits of transfer learning (TL) on both models, and compare their performance post-TL.

The paper is organized as follows: Section 2 provides a brief overview of the rotary drilling system and stick-slip vibrations. Section 3 outlines the theoretical framework of the tested techniques. Section 4 describes the model architectures, while Section 5 focuses on the training process and the results. Finally, concluding remarks are listed in Section 6.

2 Description of rotary drilling system and stick-slip vibrations

2.1 Description of rotary drilling system

Rotary drilling is the most common method of drilling in oil & gas and geothermal applications. As depicted on Fig. 1, it consists in using a drilling device called bit, which usually destroys the rock in one of the two following ways: either by a shearing action (drag bits) or by indentation (roller-cones bits). The bit is linked to a rig located at the surface by a drill string comprising a series of pipes and a lower part called Bottom-Hole Assembly (BHA). The drill bit has to be rotated and pushed against the rock formation. The force, or weight on bit (WOB), required for the bit cutters to engage the formation is obtained from the weight of the drill string. The rotation is applied either by the use of a rotary table or a top drive at the surface, and with assistance of a downhole motor in some applications. Downhole motors are particularly used for directional wells and when the bit technology requires higher rotary speeds that are not technically possible with only rotation from the surface. They help avoiding drill string twist offs by allowing most of the rotational torque to be concentrated near the bit instead of lost through torque and drag on the drill string (Anderson et al., 1990). In addition, drilling fluids and a circulation system are used to suspend and carry the cuttings and to cool down the drill bit and downhole equipment. Other equipment can be used to improve the drilling efficiency such as shock subs to damp the vibrations caused while drilling hard formation, or drilling jars that are used to free the drill string by delivering an impact load when a stuck pipe is faced (Richard, 2001).

2.2 Drilling torsional vibrations: Stick-slip

Stick-slip is a periodic low-frequency torsional oscillation characterized by fluctuations in the bit rotation speed, transitioning from zero during the stick phase to several times the surface rotation speed during the slip phase. In cases where a downhole motor is utilized, it generally maintains a non-zero bit rotational

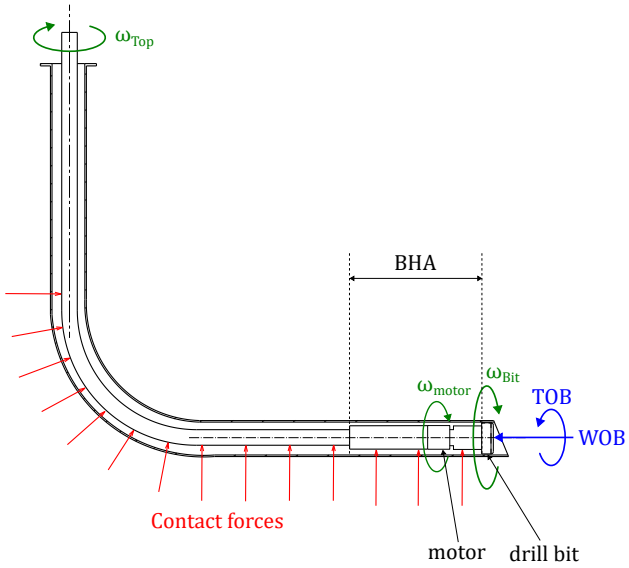
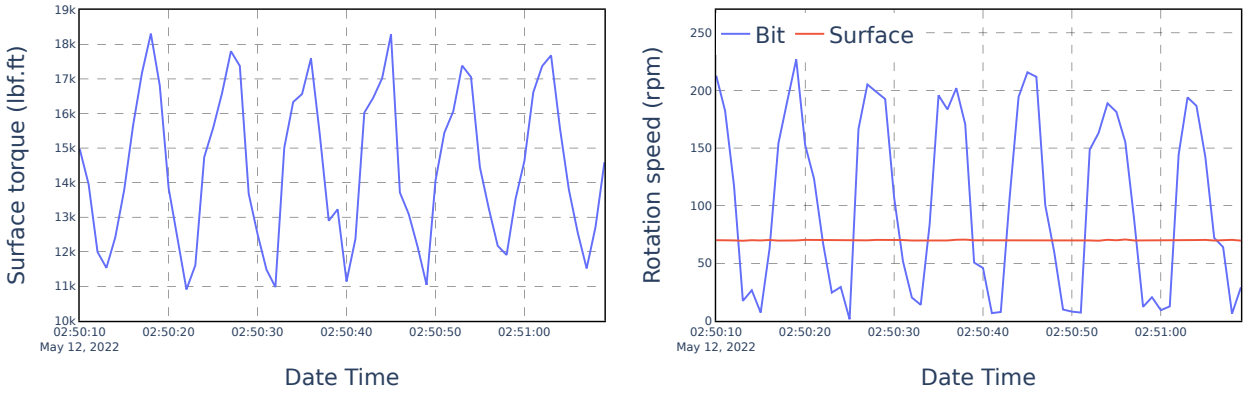


Figure 1: Simplified schematic representation of the drilling system

velocity, however, severe stick-slip can still arise. To quantify the severity of stick-slip, we calculate the stick-slip index (SSI), an industry-standard metric that normalizes the fluctuations in bit rotation speed (ω_{Bit}) over a specified time period, defined as follows:

$$SSI = \frac{\max \omega_{Bit} - \min \omega_{Bit}}{\overline{\omega_{Bit}}} \tag{1}$$

Fig. 2 depicts the changes in surface torque and downhole bit rotation speed over time throughout a severe stick-slip sequence. The bit experiences two distinct phases: the stick phase, during which the bit rotation speed is zero, and the slip phase, where the bit rotation speed reaches twice that of the surface rotation speed. While the surface rotation speed remains constant as imposed at the surface, the surface torque fluctuates in response to the variations in bit rotation speed.



(a) Surface torque.

(b) Surface and bit rotation speed.

Figure 2: Example of a sequence with severe stick-slip.

Our objective is to train a generalizable regression model to detect stick-slip based on surface measurement. The model's inputs are 60 s sequences of surface features, including: surface torque, surface weight on bit, rate of penetration, flow rate, and total rotation speed variations (surface rotation speed and downhole motor rotation speed), and the output is the predicted SSI. To ensure the model's generalizability to wells beyond those used in training, we explore the application of domain generalization techniques.

3 Methodology

Generalization in machine learning refers to the capability of a model to perform effectively well on an unknown *target* domain that differs from the *source* domain on which it was originally trained. For drilling applications, we consider a training set comprising data from multiple wells, where each well is defined by distinct characteristics. Some are known, such as length, trajectory, and bit diameter, whereas others are unknown, uncertain or changing, such as rock characteristics or bit fatigue. The wells originate from different sites across various geographical locations and correspond therefore to distinct domains having their own unique data distribution. The goal is to train a model capable to achieve minimal prediction error when applied to test data from a new well, which constitutes a different domain with a data distribution not encountered during training. Unlike traditional machine learning approaches, which assume that training and testing data originate from the same distribution, the concept of model generalization focuses on achieving robust performance across various domains without prior knowledge of the target domain.

During training, we typically have access to labeled data from one or several source domains. For the target domain, we can encounter various situations depending on the nature of the available data:

- Only unlabeled samples from the target domains: this scenario is called unsupervised domain adaptation.
- Unlabeled samples from the target domains plus few labeled target samples: this scenario is referred to as semi-supervised domain adaptation.
- No samples from the target domain during training, meaning that the specific target domain in which the model will ultimately be deployed is completely unknown. This situation is called *domain generalization* and is the one considered in the present article.

When a small number of labeled target samples become available during testing, they can be utilized to further improve the performance of the trained model on the test data. This process, known as transfer learning (Zhuang et al., 2020), involves fine tuning the trained model to better fit the specific characteristics of the new dataset (target domain).

3.1 Domain generalization

Let us denote by $\mathcal{X}$ the instance set of measurements conducted on the wells. From a mathematical perspective, a *domain* is defined as a specific distribution $\mathcal{D}$ on the instance set $\mathcal{X}$. In our context, each well constitutes its own domain on $\mathcal{X}$. Suppose we have access to N_S wells with labeled data, characterized by the *source domains* $\{\mathcal{D}_S^i\}_{i=1}^{N_S}$. Our objective is to construct a regression algorithm using data from the source domains that generalizes well to new domains.

To specify a learning problem, two ingredients are required: a distribution $\mathcal{D}$ on the instance set $\mathcal{X}$, and an unknown target function $f : \mathcal{X} \to \mathbb{R}$ that we seek to approximate. A classical approach for approximating f is to map instances into a feature space $\mathcal{Z}$ using a representation function $\mathcal{R} : \mathcal{X} \to \mathcal{Z}$ and then select a function h from a hypothesis class $\mathcal{H}$ that utilizes these features to perform the regression task. The representation function $\mathcal{R}$ induces a distribution on the feature space $\mathcal{Z}$, which we denote by $\tilde{\mathcal{D}}$. Furthermore, we denote by $\tilde{f} : \mathcal{Z} \to \mathbb{R}$ the function satisfying $f(x) = \tilde{f} \circ \mathcal{R}(x)$ for all $x \in \mathcal{X}$.

In our case, we dispose of N_S labeled dataset $\{D_S^i\}_{i=1}^{N_S}$, where $D_S^i = \{(x_{S_i}^1, f(x_{S_i}^1)), \ldots, (x_{S_i}^{m_i}, f(x_{S_i}^{m_i}))\}$ consisting of m_i measurements sampled i.i.d from the source distribution $\mathcal{D}_S^i$. Our objective is to select a

function h from the hypothesis class $\mathcal{H}$, which minimizes the expected risk ϵ_T on the target domain, defined as:

$$\epsilon_T(h) = \mathbb{E}_{z\sim\tilde{\mathcal{D}}_T}[L(h(z), \tilde{f}(z))] = \mathbb{E}_{x\sim\mathcal{D}_T}[L(h \circ \mathcal{R}(x), f(x))], \tag{2}$$

where L is the loss function used to predict the difference between the prediction $h(z)$ and the target value $\tilde{f}(z)$ and is usually selected to be the Mean-Squared Error (MSE), D denotes the original distribution of the target data, while $\tilde{\mathcal{D}}_T$ represents the distribution of the target data transformed by the representation function $\mathcal{R}$. The main challenge here is that labeled data are not available for the target distribution, making the prediction of target error impossible to estimate directly. As a consequence, we need to rely on the available training data from the N_S source domains and to adapt the training procedure to ensure good generalization.

Theoretically, it can be shown that to minimize the expected risk ϵ_T on a new domain using the available training source domains, one effective approach is to minimize the empirical risk on these source domains while simultaneously learning a representation that aligns the features across the source domains (Mansour et al., 2009; Albuquerque et al., 2019; Ben-David et al., 2006). Domain generalization algorithms aim to construct a shared representation for all source domains, which still ensures strong regression performance. In this article, we rely upon the adversarial approach presented in this paragraph to jointly learn a shared features representation for the source domains and predict the stick-slip index. These approaches to domain generalization are theoretically justified under the following assumptions (David et al., 2010):

- *Covariate shift*: the target function f should remain consistent across all domains.
- *Similarity between features distributions accross domains*: the representation $\mathcal{R}$ should result in similar features for all source domains. To quantify this similarity, the *H-divergence* is commonly used (Ganin et al., 2016; Ben-David et al., 2006; 2010; Kifer et al., 2004). In practice, the H-divergence $d_h(\mathcal{D}_S^i, \mathcal{D}_S^j)$ between the source domains $\mathcal{D}_S^i$ and $\mathcal{D}_S^j$ is estimated by training a classifier C to distinguish between domains i and j based on the features produced by the representation $\mathcal{R}$:

  $$d_h(\mathcal{D}_S^i, \mathcal{D}_S^j) \simeq 1 - 2\mathrm{err}_{\mathrm{cl}}(C), \tag{3}$$

  where the classification error is estimated based on the empirical datasets D_S^i and D_S^j with respective sizes m_i and m_j as

  $$\mathrm{err}_{\mathrm{cl}}(C) = \frac{1}{m_i + m_j} \sum_{k=1}^{m_i+m_j} L_{\mathrm{cl}}\big(C(z_k), \mathbf{1}_{D_S^i}(z_k)\big). \tag{4}$$

  In Eq. (4) L_{cl} is a classification loss, usually the cross-entropy loss and $\mathbf{1}_{D_S^i}(z)$ is the indicative function taking the value 1 when z comes from dataset D_S^i. A high classification loss suggests that the classifier struggles to differentiate between the source domains. Conversely, a small H-divergence implies high similarity between the distributions $\mathcal{D}_s$ and $\mathcal{D}_t$.
- *Existence of a suitable mapping function*: There must exist a function h^* that can accurately map features z to their respective labels, regardless of the domain.

According to (David et al., 2010), only the latter two assumptions are essential for applying model generalization. Furthermore, the covariate shift assumption is considered less restrictive, as it can be validated without necessarily ensuring the feasibility of model generalization.

Adversarial approach for domain generalization

Domain adversarial training is commonly used for learning domain-invariant features. This approach was originally introduced by Ganin and Lempitsky (Ganin and Lempitsky, 2015; Ganin et al., 2016) in the context of domain adaptation. As described in section 3.1, minimizing the target error ϵ_T on a new well amounts to reducing both the expected risks ϵ_{S_i} on each source domain and the discrepancy between source domains. To that end, we aim to learn a representation $\mathcal{R}$ that aligns the source domains but still produces features that remain relevant for the regression task at hand (Ganin and Lempitsky, 2015). To achieve this,

adversarial training is used to learn domain-invariant features by simultaneously training two components: a generator and a discriminator in an adversarial manner. The input x is processed by an embedding function or representation $\mathcal{R}$ (generator) to learn a domain invariant feature representation, which is used as input to the regression model h for mapping to the stick-slip index. To ensure alignment between the source domains, a domain classifier C (discriminator) is trained to distinguish between the source domains based on their embedded features. The goal is to maximize the domains classifier's loss, indicating that the embedded features from the source domains are sufficiently similar, making it difficult for the classifier to distinguish between them. For a dataset constituted of m observations $\{(x_i, f(x_i)) \sim \mathcal{D}_S^{k_i}\}_{i=1}^m$, the final loss function is:

$$\min \frac{1}{m} \sum_{i=1} L\Big(h(\mathcal{R}(x_i)), f(x_i)\Big) - \lambda \, L_{cl}\Big(C(\mathcal{R}(x_i)), \mathbf{1}_{D_S^{k_i}}(x_i)\Big), \tag{5}$$

where the second term is used to penalize the distance between the embedded features for the source and target domains, L_{cl} being the classification loss (cross-entropy loss) computed across the source domains, λ being a weighting coefficient, and $\mathbf{1}_{D_S^{k_i}}$ is indicative function which takes the value 1 if the element x_i belongs to the source domain $D_S^{k_i}$, and 0 otherwise.

3.2 Transfer learning

During the training phase, we do not have access to labeled target samples. However, during testing, a few labeled samples may become available, which can be used to enhance the trained model's performance on this specific target domain. To achieve this, the trained model (source model) can be fine-tuned by either adjusting all of its parameters or selectively updating some of them using only the labeled target samples as retraining data (Zhuang et al., 2020). In our case, the source model is a regression model trained across various source domains. The fine-tuning process, using the small set of labeled target samples, is fast due to the limited amount of data involved compared to what is required to train a new model.

4 Model characteristics

4.1 Adversarial domain generalization training

For adversarial domain generalization, we utilize the model architecture proposed by Ganin et al. (2016), which comprises three main components (see Fig. 3):

- **Generator (or feature extractor):** This component processes 60 s sequences of surface measurements, generating a D-dimensional embedded feature vector $z \in \mathbb{R}^D$. The generator's parameters are denoted by θ_G, and the relationship is expressed as $z = G(x; \theta_G)$, where G is the same as $\mathcal{R}$ in Eq. (2).
- **Discriminator (or domain classifier):** The discriminator takes the embedded feature vector z as input and classifies it based on the annotated data from the source domain. Its parameters are represented by θ_C, and its output is the predicted domain for each sequence: $C(z; \theta_C)$ (see Eq. (4)).
- **SSI-predictor:** This component utilizes the embedded features from the generator to predict the stick-slip severity index (SSI) for each sequence. The parameters for this mapping are denoted by θ_{SSI} and the regression output is $h(z; \theta_{SSI})$ (see Eq.(2)).

During the training phase, our objective is to minimize the SSI prediction loss for the training wells while optimizing the generator parameters to make the embedded features z (where $z = G(x; \theta_G)$) domain-invariant, which involves aligning the source distributions $\{G(x_i, \theta_G), x_i \sim D_{s_i}\}$ for $i \in \{1, 2, ..., N_s\}$, where N_s is the number of training wells. To achieve this, we aim to find the generator parameters θ_G that maximize the domain classifier loss, promoting domain-invariance and making the feature distributions across domains as similar as possible. Simultaneously, we optimize the domain classifier and the SSI-predictor parameters, θ_C

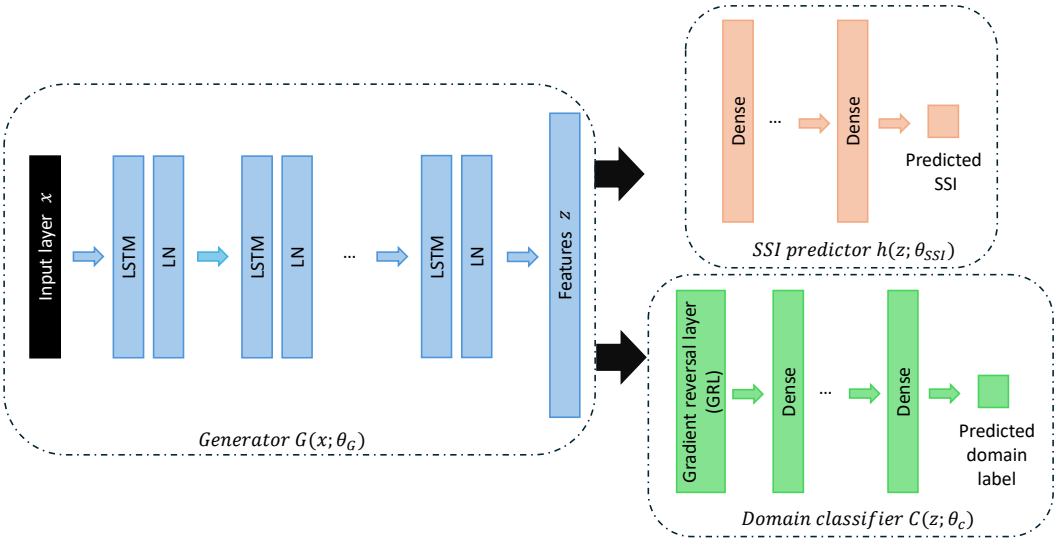


Figure 3: Architecture of adversarial domain generalization model. The generator is composed of a sequence of Long Short-term Memory (LSTM) and Layer Normalization (LN) layers; the SSI predictor and domain classifier are fully connected neural networks; a Gradient Reversal Layer (GRL) is used to reverse the direction of the gradients during backpropagation.

and θ_{SSI} respectively, to minimize their losses. The corresponding optimization problem is given by

$$\hat{\theta}_C, \hat{\theta}_{SSI}, \hat{\theta}_G = \text{argmax}_{\theta_C} \text{argmin}_{\theta_G, \theta_{SSI}} E(\theta_G, \theta_C, \theta_{SSI}) = L(\theta_G, \theta_{SSI}) - \lambda L_{cl}(\theta_G, \theta_C). \tag{6}$$

In (6), L is the loss function used for the SSI prediction (mean squared error), while L_{cl} is the classification loss (cross-entropy loss), computed across the N_s source domains. The parameters θ_C of the domain classifier are optimized to minimize the domain classification loss (Eq. (7)), the parameters θ_{SSI} of the SSI-predictor are optimized to minimize the SSI prediction loss, while the feature mapping parameters θ_G are adjusted to both minimize the SSI prediction loss and maximize the domain classification loss (Eq. (8)). The parameter λ is a weighting coefficient used to control the trade-off between the losses (Ganin and Lempitsky, 2015).

$$\hat{\theta}_C = \text{argmax}_{\theta_C} E(\hat{\theta}_G, \theta_C, \hat{\theta}_{SSI}) \tag{7}$$

$$(\hat{\theta}_G, \hat{\theta}_{SSI}) = \text{argmin}_{\theta_G, \theta_{SSI}} E(\theta_G, \hat{\theta}_C, \theta_{SSI}) \tag{8}$$

The generator's architecture consists of a sequence of Long Short-Term Memory (LSTM) layers combined with Layer Normalization (LN) layers. LSTM networks are well-suited for time series data, as they capture long-term dependencies between time steps by using previous outputs as inputs for subsequent steps, similar to recurrent neural networks (Staudemeyer and Morris, 2019). Layer Normalization, on the other hand, normalizes the aggregated inputs to the neurons within each hidden layer (Ba et al., 2016). Both the discriminator and SSI-predictor are fully-connected neural networks. The discriminator's last layer has N_s neurons, where N_s is the number of training wells. The SSI-predictor has a single output neuron responsible for predicting the SSI. The specificity of the discriminator is the addition of a gradient reversal layer (GRL) as input layer (Ganin et al. (2016)). The GRL's primary function is to reverse the direction of the gradients during backpropagation. During the forward pass, the GRL behaves as an identity function, passing the input through unchanged. However, in the backward pass, it multiplies the gradients of the discriminator loss by a negative constant λ (as defined in Eq .6), effectively reversing the gradients before they are propagated back to the generator. Both the discriminator and SSI-predictor use the ReLU activation function in all layers, except for the discriminator's final layer, which uses the softmax function for domain classification. The generator, on the other hand, employs the hyperbolic tangent (tanh) as the activation function and the

sigmoid function as the recurrent activation function. Additionally, bias, kernel, and recurrent regularization techniques are applied to prevent overfitting. For the SSI-predictor, we applied the MSE as the loss function, while the discriminator utilized categorical cross-entropy loss. In terms of metrics, we used mean absolute error (MAE) and mean absolute percentage error (MAPE) for the SSI-predictor, and accuracy for the discriminator. The Adam optimizer was used for all components, with a learning rate of 10^{-3} for each.

In recent years, the transformer model (Vaswani, 2017) has gained popularity over LSTM, as it eliminates the sequential processing of LSTMs by using self-attention mechanisms to process all time steps in parallel. This parallelization is often seen as advantageous for time series tasks. However, in choosing the architecture for the generator, we compared LSTMs architecture with transformers. During testing, the transformer model required significantly more time to train (training time increased by a factor of five), and its performance was less favorable compared to LSTMs, which are more efficient with smaller datasets. For these reasons, we opted to use LSTM layers for the generator architecture.

4.2 Baseline model

The baseline model (Yahia et al., 2024a) consists of a single component that takes 60 s sequences of surface measurements as input and predicts the SSI. The architecture is exactly the same as the generator, featuring a series of LSTM and LN layers. The only addition is a final dense layer with a single neuron responsible for outputting the predicted SSI. The loss function, metrics, and optimizer used are identical to those of the SSI-predictor.

5 Experiments

In this section, we present a variety of results for training a machine learning model to predict the SSI using 60 s sequences of surface measurements as input. The model is trained using different training wells (source domains) and tested on completely different wells (target domain) in three different approaches:

- **Adversarial domain generalization**: As explained in section 3.1, the model consists of three components: a generator that projects the input sequences into a domain invariant feature representation, a regression model that predicts the SSI , and a domain classifier that differentiates between the different source domains using the invariant features as input.
- **Baseline model**: This is a traditional deep learning model consisting of a single component that takes the 60 s sequences of surface measurements as input and predicts the SSI as output.
- **Transfer learning**: In contrast to the previous techniques, transfer learning is applied during the testing phase. Once a model is trained (either the adversarial domain generalization or baseline model), if labeled surface data sequences from the test well are available during testing, we can apply transfer learning technique to improve the model's performance.

5.1 Data processing

Extensive evaluation are performed of the proposed approaches on a number of distinct wells as outlined in Table 1. Some of these wells are from the same field and share similar characteristics. To assess the generalizability of the trained model, we ensure that wells from the same field are all consistently used either as test data or training data, but not both. The first three wells, which originate from the same field and represent a substantial number of sequences, were selected for the training process. The last three wells were reserved for testing, as two of them are from the same field, with an additional distinct well included to further evaluate the model's generalizability. The remaining wells (4, 5, and 6) were used alternatively for training and validation (see section 5.2).

Table 1: Source and target well characteristics, where vertical wells are drilled straight down from the surface into the target formation or reservoir, and lateral wells are drilled vertically to a certain depth, then deviates horizontally within the target.

Well	**Field number**	**Type of wellbore trajectory**	**Well length (ft)**	**Bit diameter (inch)**	**Drill Pipe diameter (inch)**	**BHA length (ft)**	**Number of sequences**
1	1	Lateral	18 120	8 3/4	5	167.94	33 303
2	1	Lateral	18 204	8 1/2	5	166.9	50 285
3	1	Lateral	17 921	8 3/4	5	136.94	15 147
4	2	lateral	27 723	8 3/4	5 1/2	101.03	52 008
5	3	Lateral	19 355	6 3/4	5 1/2	128.99	119 613
6	4	Lateral	29 077	8 3/4	5	130.52	255 687
7	5	Lateral	28 645	8 3/4	5 1/2	97.36	449 002
8	6	Vertical	20 619	8 3/4	5	110.31	22 054
9	6	Vertical	20 679	8 3/4	5	111.23	136 824

For each well described in Table 1, 1 Hz time series of surface and downhole drilling data are collected. The surface data, which includes surface torque, surface weight on bit, rate of penetration, flow rate, and total rotation speed variations, is divided into 60 s sequences where each sequence serves as an input sample for the regression model. The downhole data contains the bit rotation speed used to label the surface data. For each 60 s sequence, we inspect the downhole bit rotation speed and we calculate the true SSI as described in Eq. (1).

5.2 Grid Search for Hyperparameter tuning

To tune our model's hyperparameters, we employ the traditional grid search method. This approach involves exhaustively searching through a specified subset of the hyperparameter space for the training algorithm. The hyperparameters evaluated include the regularization parameter for the generator (covering bias, kernel, and recurrent regularization), the number of hidden layers in the generator, and the loss function weighting coefficient λ (Eq. (5)). The potential values for the regularization coefficient are $(10^{-3}, 10^{-4}, 10^{-5})$, the number of hidden layers in the generator are set to $(4, 6, 8)$, while the weighting coefficient λ varies among $(1, 10, 100, 1000)$. In our experiments, validation data is used to determine the optimal hyperparameter values. Since our goal is to develop a model that generalizes effectively accross sources, we select validation data from wells entirely distinct from those used in training. Therefore, in each training session, from the first six wells listed in Table 1, four wells are used for training and the remaining two for validation. The last three wells (from two different rigs) are reserved for testing after the hyperparameters have been selected. Selecting the two wells for validation among the first six wells is challenging, as this choice significantly influences both model performance and hyperparameters selection: when wells with characteristics similar to the majority of training wells were chosen, the SSI validation error tended to be low. In contrast, choosing wells with different characteristics resulted in a higher validation error. To address this, we test three different validation data cases, as outlined in Table 2. In each case, the first three wells in Table 1, which are from the same field, are fixed as training data, and wells 4, 5, and 6, from 3 different rigs, are used alternately to select two wells for validation in each scenario.

Table 2: Validation data selection for the three tested cases.

	Well 4	**Well 5**	**Well 6**
Case 1	**Validation data**	Training data	**Validation data**
Case 2	**Validation data**	**Validation data**	Training data
Case 3	Training data	**Validation data**	**Validation data**

5.2.1 Regularization coefficient

Due to limited computational capacity, we do not search for all the hyperparameters simultaneously. Instead, we begin by fixing the number of hidden layers in the generator at 6 while varying the regularization coefficient and the weighting coefficient across the different validation data cases described in Table 2. The architectures for the discriminator and SSI predictor were fixed at 5 dense layers, each containing $[60, 40, 20, 10]$ neurons for the first four layers, and 6 neurons (representing the number of training and validation wells) and 1 neuron (for SSI) for the last layers, respectively. The generator architecture comprised 5 LSTM layers combined with 5 layer normalization (LN) layers (including 2 input layers, 6 hidden layers, and 2 output layers), with 64 neurons per layer and bias, kernel, and recurrent regularization applied uniformly across all layers. Each architecture search was conducted for 500 epochs, utilizing a constant learning rate of 10^{-3} on an Nvidia GPU. For each combination of hyperparameters, and for each validation data case, the model was trained using three different initializations. For each initialization, the trained model was used to calculate the MSE and dynamic time warping (DTW) (Li, 2021) which is a computational technique used to measure the similarity between two time series that may vary in speed or timing, between the actual and predicted SSI validation data. The average of the three initializations was then computed. The key advantages of DTW are its robustness to outliers, making it less sensitive to abnormal points, and its ability to stretch or compress parts of the time series to achieve optimal alignment. In contrast, MSE calculates point-by-point differences in magnitude between the two time series, without accommodating variations in timing or alignment. However, MSE is generally easier to interpret, as it measures the average of the squared differences between predicted and actual values, making it more straightforward than DTW. To facilitate comparisons, we decided to normalize the DTW by the number of corresponding well sequences.

Fig. 4 illustrates the average MSE on validation data across the three tested initializations for each of the three cases. As we can see, results vary across cases, with case 2 exhibiting the lowest MSE SSI validation error. This outcome is likely due to the training dataset's sequence configuration, as for case 2, wells 4 and 5 were selected for validation, while well 6, with the highest sequence number, was included in the training dataset. Fig 5 displays the mean MSE and normalized DTW on validation data, calculated over the three tested cases. Notably, the error variations remain consistent whether using MSE or DTW. The results indicate that the lowest validation SSI prediction error is attained with a regularization coefficient of 10^{-4} for most of the tested weighting coefficients. Therefore, this hyperparameter will be set at 10^{-4} for the remainder of the paper.

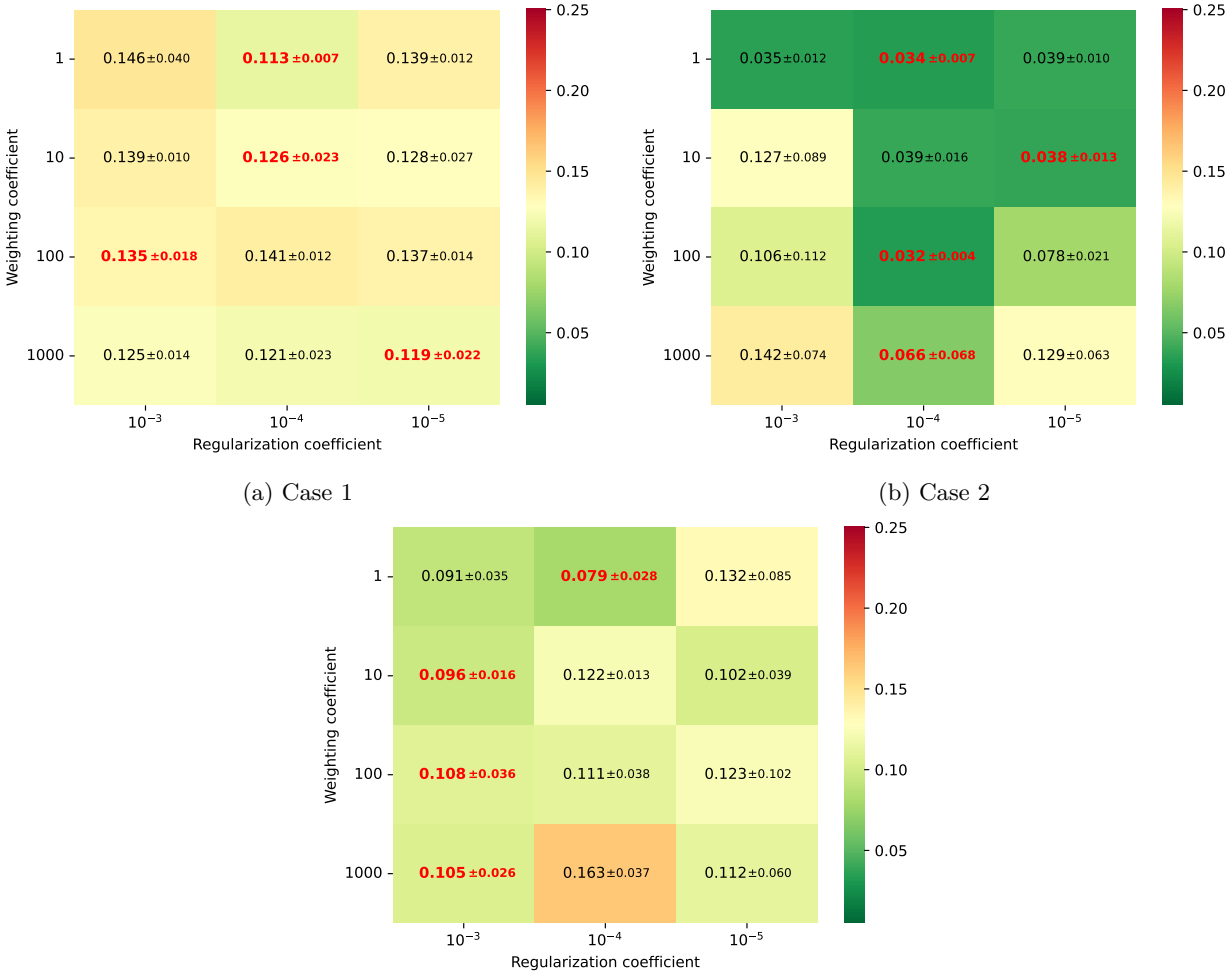
5.2.2 Number of generator hidden layers and weighting coefficient

To select the optimal number of hidden layers for the generator and the weighting coefficient λ (see Eq. 6), we conduct a grid search using three different initializations across the three validation cases (see Table .2), following the same approach as for the regularization coefficient. The tested values for the generator's hidden layers are $(4, 6, 8)$ and for the weighting coefficient λ we set $(1, 10, 100, 1000)$. The optimal parameters are chosen based on the SSI validation error. Similar to the regularization coefficient selection, Fig 6 shows the average MSE on validation data across the three initializations for each case, while Fig 7 presents the mean MSE and normalized DTW on validation data, averaged across the three cases. We choose the hyperparameter configuration with the lowest SSI validation error, which includes 6 hidden layers for the generator and a weighting coefficient of 10. These hyperparameters will be fixed for the remainder of the paper.

5.3 Results and discussions

5.3.1 Comparison between Domain Generalization and baseline models

With the selected hyperparameters, we proceed to compare the adversarial domain generalization technique applied to time series for SSI prediction (section 4.1) against the baseline model (section 4.2). For training, the first six wells are designated as training data and the last three as test data (Table 2), with 10% of the training data reserved for validation. Each model is trained for 1000 epochs, with a batch size of 2048, and a fixed learning rate of 10^{-3}, and we retain the model that achieves the lowest SSI validation MSE. We repeat the training five times, each time with a different initialization, and we calculate the average test SSI



(a) Case 1

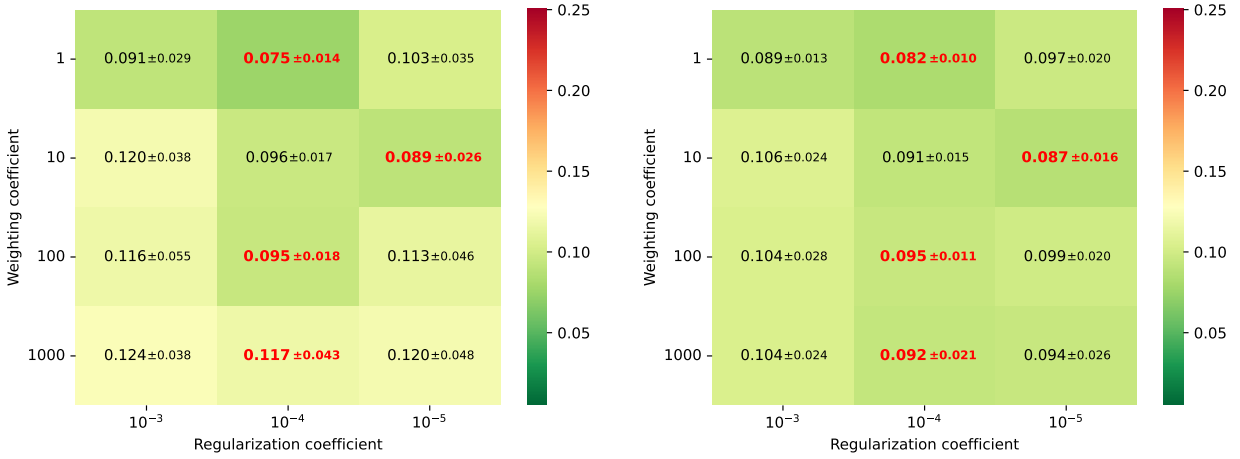
(b) Case 2

(c) Case 3

Figure 4: Average MSE of SSI validation data for the three tested cases with varying regularization and weighting coefficients over three different intializations: The model is trained for each combination of hyper-parameters using three distinct initializations. The validation SSI error is calculated for each one, and the average error is then computed across the three initializations.

prediction error across the five runs. The adversarial domain generalization model requires approximately six hours of training, while the baseline model takes around four hours.

Table. 3 shows the normalized DTW results for the three test wells, over the five runs, for the adversarial domain generalization and baseline models. The adversarial domain generalization model demonstrates better generalization than the baseline model, achieving a 10 % improvement in results. Figures 8 and 9 illustrate examples of true and predicted SSI sequences for both models across the three test wells. As shown in Figures 8a and 8b, both models generally perform well in predicting SSI for most sequences. However, there are also sequences where both models fail to capture SSI variations, such as in Figure 8c, where both models overestimate the SSI, and Figure 8d, where both underestimate the SSI. Notably, in some cases, the adversarial domain generalization (Fig. 9) provides more accurate SSI predictions than the baseline. This indicates that the generator successfully projects the training data into a domain-invariant feature space, enabling the SSI predictor to make more reliable predictions using the newly projected features.



(a) MSE (b) DTW

Figure 5: Average of the MSE and DTW of SSI validation data for the three tested cases with varying regularization and weighting coefficients.

Table 3: Normalized SSI DTW of the trained domain generalization and baseline models on the three test wells: Each model is trained with five different initializations, and the error is reported as the average across these five runs.

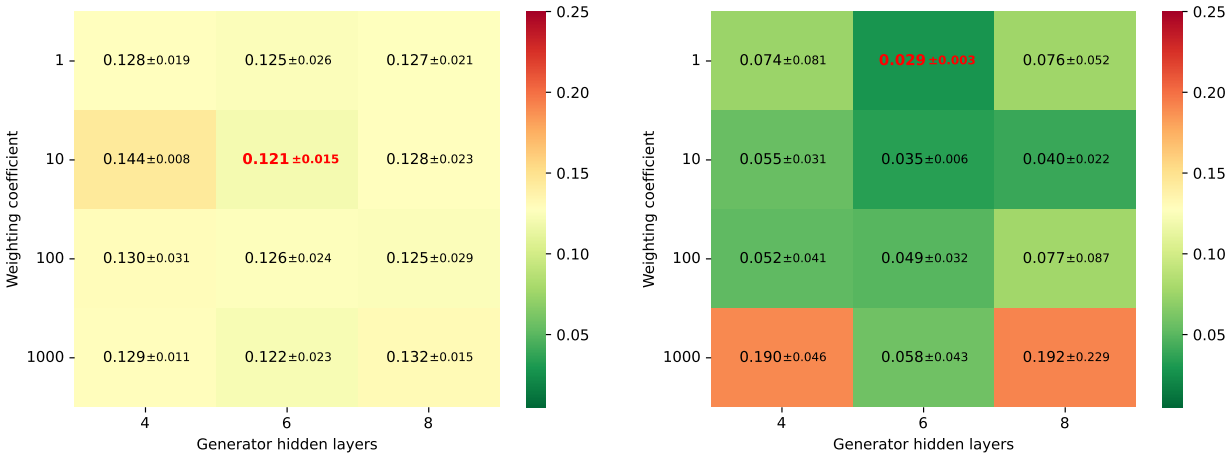
	Baseline	**Adversarial domain generalization**	**%**
Test well 1	0.136	0.122	**10.29**
Test well 2	0.086	0.075	**12.79**
Test well 3	0.107	0.097	**9.34**

5.3.2 Transfer learning application

To enhance model performance on a specific target well, transfer learning technique can be applied if a small amount of labeled surface drilling data for that well is available (see section. 3.2). In our case, we test the contribution of transfer learning on each of the three test wells by using the first 10 % of labeled surface drilling data with both the adversarial domain generalization and baseline models. The source model, either the trained adversarial domain generalization or baseline model, is then fine-tuned to adapt to the target well by retraining only the weights and biases of the first two layers in the generator and SSI predictor for the adversarial domain generalization model, and the first two layers in the baseline model. The same optimizer and learning rate as in the source model training are used, but with a significantly reduced number of epochs, resulting in a lighter computational load, with a retraining time of approximately 1.5 minutes.

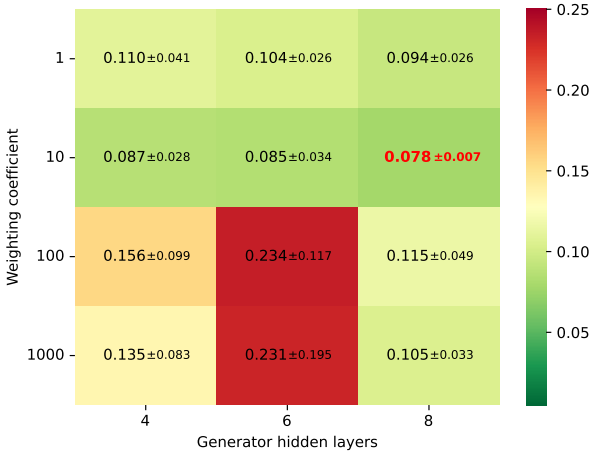
Table. 4 presents a comparison of model performance before and after applying transfer learning for both adversarial domain generalization and baseline models. For each test well, the source model (either adversarial domain generalization or baseline) is partially retrained using the first 10 % of labeled surface drilling data. The results in the table show that transfer learning enhances model performance in both adversarial domain generalization and baseline models, with a decrease in normalized DTW following retraining. Additionally, this transfer learning approach is efficient in terms of time and requires only a small amount of data compared to what would be needed to train a new model.

Fig. 10 shows the actual and the predicted values of SSI over time for the three test wells, comparing results with and without the application of transfer learning. As we can see, after transfer learning was applied, the new model predicts the SSI better than the source one for both models. Additionally, even after transfer learning application, the adversarial domain generalization model outperforms the baseline model, with a lower normalized DTW across all three test wells (see Table. 4).



(a) Case 1

(b) Case 2

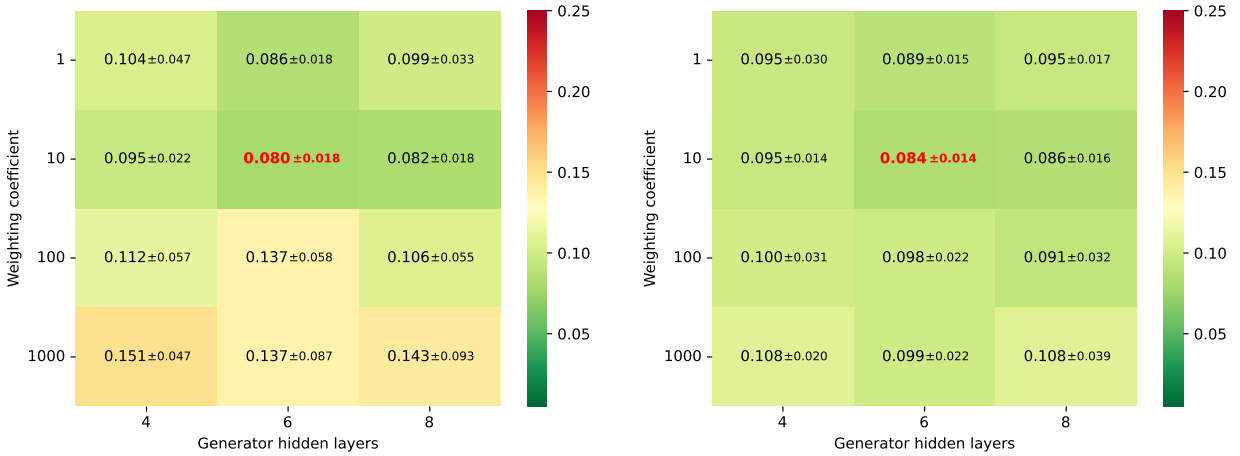


(c) Case 3

Figure 6: Average MSE of SSI validation data for the three tested cases with varying generator hidden layer number and weighting coefficients over three different intializations: The model is trained for each combination of hyperparameters using three distinct initializations. The validation SSI error is calculated for each one, and the average error is then computed across the three initializations.

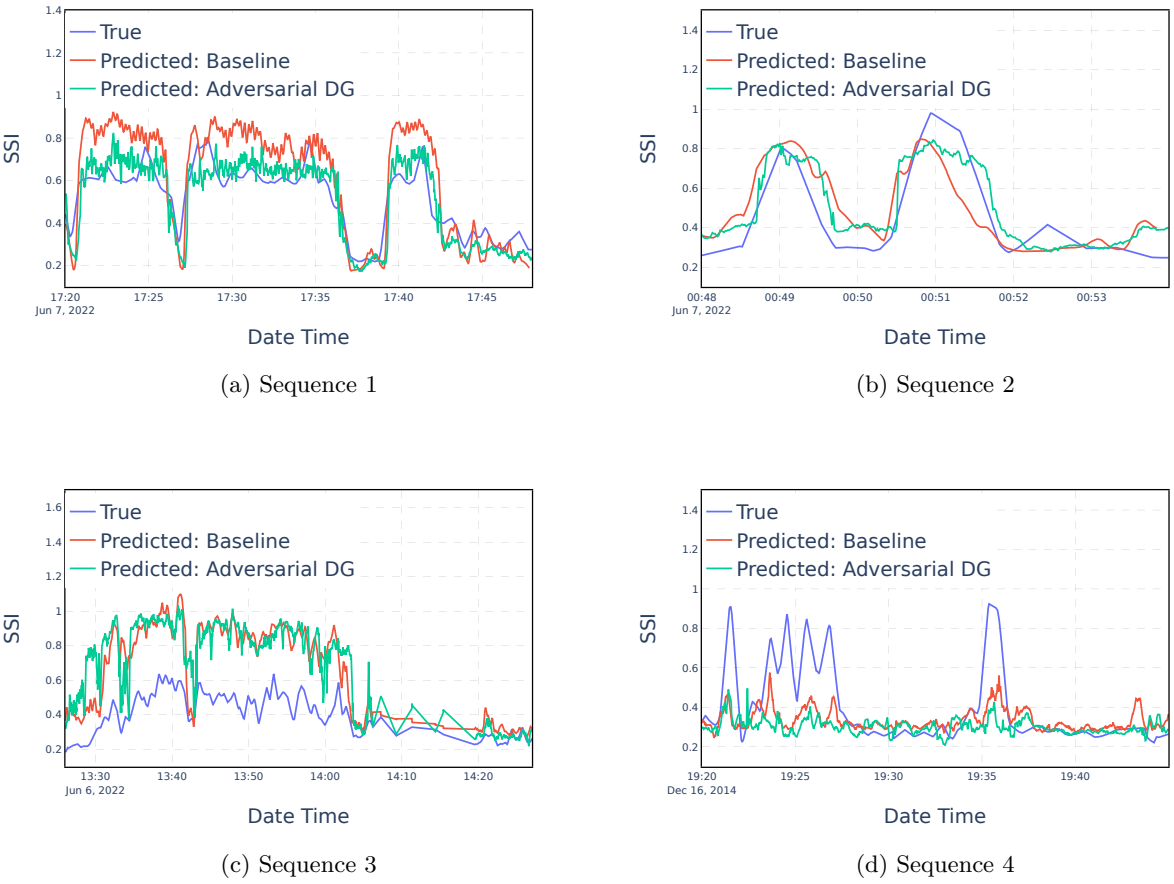
Table 4: Comparison of model performance, for both adversarial domain generalization and baseline models, before and after applying transfer learning, using normalized SSI DTW as the evaluation metric across the three test wells: Each source model undergoes partial retraining with the first 10 % of labeled surface drilling data from each test well.

	Adversarial domain generalization			**Baseline**		
	Without TL	**With TL**	**%**	**Without TL**	**With TL**	**%**
Test well 1	0.122	0.095	**22.13**	0.136	0.110	**19.11**
Test well 2	0.075	0.058	**22.66**	0.086	0.06	**30.23**
Test well 3	0.097	0.088	**9.27**	0.107	0.100	**6.54**



(a) MSE (b) DTW

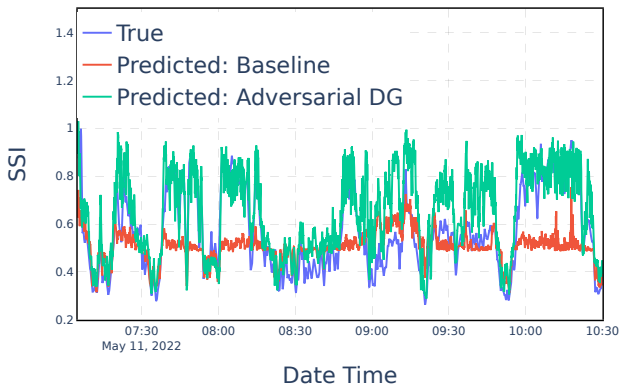
Figure 7: Average of the MSE and DTW of SSI validation data for the three tested cases with varying generator hidden layer number and weighting coefficients.



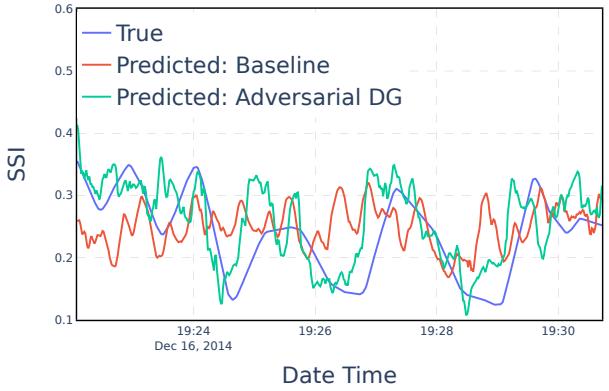
(a) Sequence 1 (b) Sequence 2

(c) Sequence 3 (d) Sequence 4

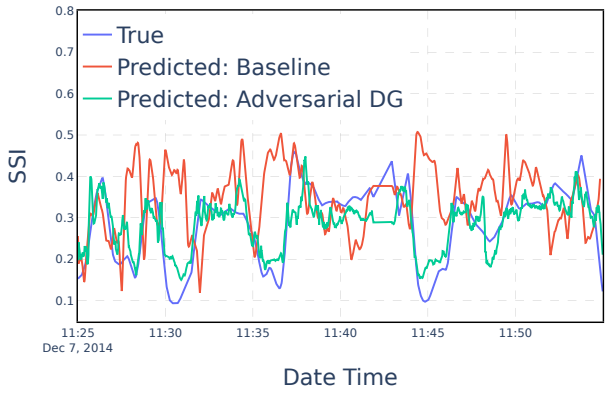
Figure 8: True and predicted SSI values over time, for both adversarial domain generalization and baseline trained models based on sequences from the three test wells. First two sequences: Both models accurately predict SSI. Last two sequences: Both models fail to capture SSI variations.



(a) Test well 1

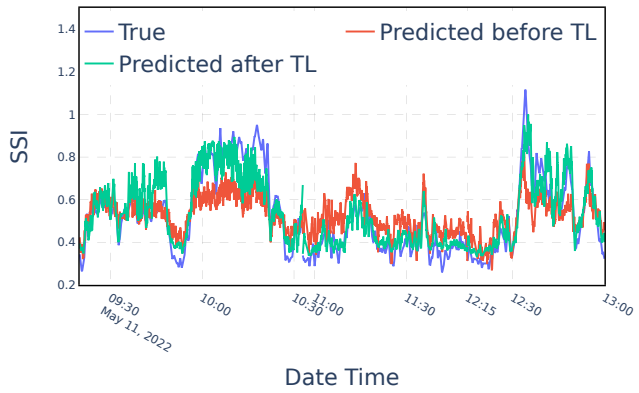


(b) Test well 2

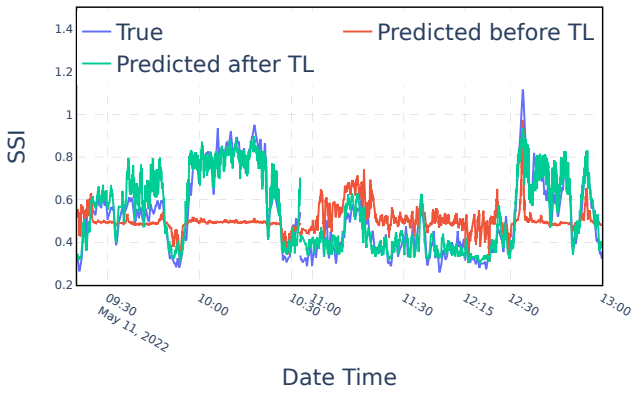


(c) Test well 3

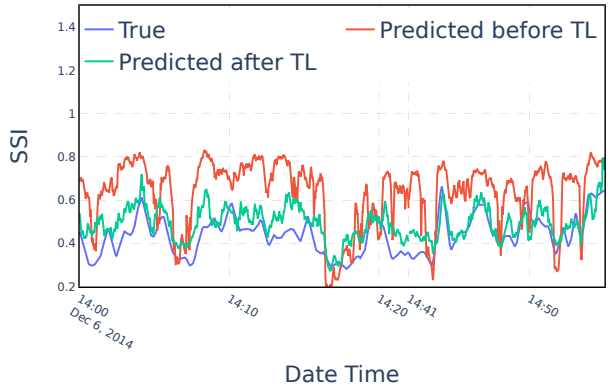
Figure 9: True and predicted SSI values over time, for both adversarial domain generalization and baseline trained models based on sequences from the three test wells.



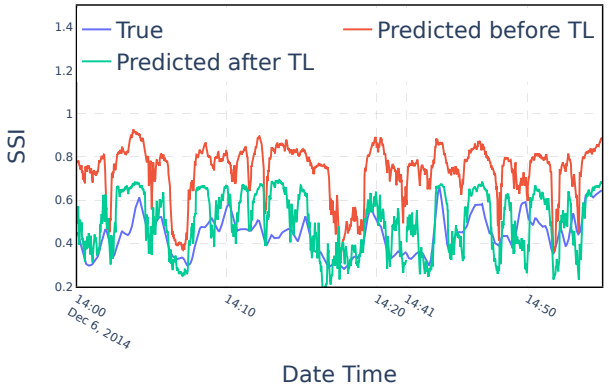
(a) Test well 1: Adversarial domain generalization model



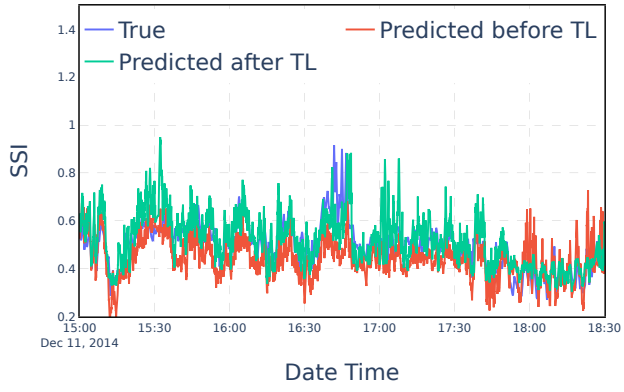
(b) Test well 1: Baseline model



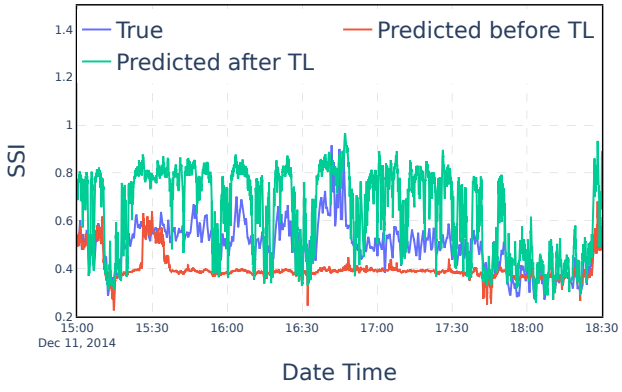
(c) Test well 2: Adversarial domain generalization model



(d) Test well 2: Baseline model



(e) Test well 3: Adversarial domain generalization model



(f) Test well 3: Baseline model

Figure 10: True and predicted SSI values over time, for both adversarial domain generalization and baseline models applied on sequences from the three test wells before and after the application of transfer learning.

6 Conclusion

In this paper, we introduced the application of domain generalization techniques to time series data for predicting the Stick-Slip Index (SSI) in drilling operations, utilizing 60 s sequences of 1 Hz surface drilling data as inputs. We presented a comparison of adversarial domain generalization with traditional baseline model and transfer learning technique. A grid search methodology was employed to optimize parameters, including the regularization coefficient, the number of hidden layers in the generator, and the weighting coefficient for the model. Our results reveal that adversarial domain generalization model significantly enhances the generalization capabilities of machine learning models, yielding a 10 % improvement over the baseline model. Additionally, the implementation of transfer learning on a pre-trained model (whether adversarial domain generalized or baseline) demonstrated improved performance. Even after transfer learning application, the adversarial domain generalization model outperforms the baseline model, which highlights the generator's ability to map the training data sequences into a space where the SSI predictor cannot differentiate among them.

In future works, we intend to implement adversarial domain generalization using more training wells. Additionally, we plan to explore the adversarial domain adaptation technique and evaluate its performance in comparison to transfer learning.

References

Isabela Albuquerque, João Monteiro, Mohammad Darvishi, Tiago H Falk, and Ioannis Mitliagkas. Generalizing to unseen domains via distribution matching. *arXiv preprint arXiv:1911.00804*, 2019.

Maurer Anderson, Cooper Hood, and Cook. Deep drilling basic research: volume 4-system description. final report. Technical report, Maurer Engineering Inc., Houston, TX and University of California, 1990.

Parimal Arjun Patil and Catalin Teodoriu. Model development of torsional drillstring and investigating parametrically the stick-slips influencing factors. *Journal of Energy Resources Technology*, 135(1):013103, 2013.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

JR Bailey, GS Payette, MT Prim, J Molster, AW Al Mheiri, PG McCormack, and K LeRoy. Mitigating drilling vibrations in a lateral section using a real-time advisory system. In *Abu Dhabi International Petroleum Exhibition and Conference*, page D011S020R004. SPE, 2017.

Mahsa Baktashmotlagh, Mehrtash T Harandi, Brian C Lovell, and Mathieu Salzmann. Unsupervised domain adaptation by domain invariant projection. In *Proceedings of the IEEE international conference on computer vision*, pages 769–776, 2013.

Theresa Baumgartner and Eric van Oort. Pure and coupled drill string vibration pattern recognition in high frequency downhole data. In *SPE Annual Technical Conference and Exhibition*, pages SPE–170955. SPE, 2014.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19, 2006.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79:151–175, 2010.

Karsten M Borgwardt, Arthur Gretton, Malte J Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J Smola. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22 (14):e49–e57, 2006.

Shai Ben David, Tyler Lu, Teresa Luu, and Dávid Pál. Impossibility theorems for domain adaptation. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 129–136. JMLR Workshop and Conference Proceedings, 2010.

Behzad Elahifar and Erfan Hosseini. A new approach for real-time prediction of stick–slip vibrations enhancement using model agnostic and supervised machine learning: a case study of norwegian continental shelf. *Journal of Petroleum Exploration and Production Technology*, 14(1):175–201, 2024.

Tongtong Fang, Nan Lu, Gang Niu, and Masashi Sugiyama. Rethinking importance weighting for deep learning under distribution shift. *Advances in neural information processing systems*, 33:11996–12007, 2020.

Yuqi Fang, Pew-Thian Yap, Weili Lin, Hongtu Zhu, and Mingxia Liu. Source-free unsupervised domain adaptation: A survey. *Neural Networks*, page 106230, 2024.

Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *Proceedings of the IEEE international conference on computer vision*, pages 2960–2967, 2013.

Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of machine learning research*, 17(59):1–35, 2016.

Boqing Gong, Kristen Grauman, and Fei Sha. Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In *International conference on machine learning*, pages 222–230. PMLR, 2013.

Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *2011 international conference on computer vision*, pages 999–1006. IEEE, 2011.

Chiranth Hegde, Harry Millwater, and Ken Gray. Classification of drilling stick slip severity using machine learning. *Journal of Petroleum Science and Engineering*, 179:1023–1036, 2019.

Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex Smola. Correcting sample selection bias by unlabeled data. *Advances in neural information processing systems*, 19, 2006.

Yunpei Jia, Jie Zhang, Shiguang Shan, and Xilin Chen. Single-side domain generalization for face anti-spoofing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8484–8493, 2020.

Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *VLDB*, volume 4, pages 180–191. Toronto, Canada, 2004.

Young-Bum Kim, Karl Stratos, and Dongchan Kim. Adversarial adaptation of synthetic or stale data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1297–1307, 2017.

Dimitrios Kollias, Anastasios Arsenos, and Stefanos Kollias. Domain adaptation explainability & fairness in ai for medical image analysis: Diagnosis of covid-19 based on 3-d chest ct-scans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4907–4914, 2024.

Hailin Li. Time works well: Dynamic time warping based on time weighting for time series data mining. *Information Sciences*, 547:592–608, 2021.

Jingjing Li, Zhiqi Yu, Zhekai Du, Lei Zhu, and Heng Tao Shen. A comprehensive survey on source-free domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5743–5762, 2024.

Xiang Li, Wei Zhang, Hui Ma, Zhong Luo, and Xu Li. Domain generalization in rotating machinery fault diagnostics using deep neural networks. *Neurocomputing*, 403:409–420, 2020.

Zheng Li, Yun Zhang, Ying Wei, Yuxiang Wu, and Qiang Yang. End-to-end adversarial memory network for cross-domain sentiment classification. In *IJCAI*, pages 2237–2243, 2017.

Mingxin Liu, Jiong Mu, Zitong Yu, Kun Ruan, Baiyi Shu, and Jie Yang. Adversarial learning and decomposition-based domain generalization for face anti-spoofing. *Pattern Recognition Letters*, 155:171–177, 2022.

Bo-Qun Ma, He Li, Yun Luo, and Bao-Liang Lu. Depersonalized cross-subject vigilance estimation with adversarial domain generalization. In *2019 International joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2019.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. *arXiv preprint arXiv:0902.3430*, 2009.

Toshihiko Matsuura and Tatsuya Harada. Domain generalization using a mixture of multiple latent domains. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11749–11756, 2020.

Aakanksha Naik and Carolyn Rose. Towards open domain event trigger identification using adversarial domain adaptation. *arXiv preprint arXiv:2005.11355*, 2020.

Thomas Marc Richard. *Self-excited stick-slip oscillations of drag bits*. University of Minnesota, 2001.

Ramy Saadeldin, Hany Gamal, and Salaheldin Elkatatny. Detecting downhole vibrations through drilling horizontal sections: machine learning study. *Scientific Reports*, 13(1):6204, 2023.

Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. Wasserstein distance guided representation learning for domain adaptation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Yuelin Shen, Zhengxin Zhang, Jie Zhao, Wei Chen, Mohammad Hamzah, Richard Harmer, and Geoff Downton. The origin and mechanism of severe stick-slip. In *SPE annual technical conference and exhibition*. OnePetro, 2017.

Prasham Sheth, Indranil Roychoudhury, Crispin Chatar, and José Celaya. A hybrid physics-based and machine-learning approach for stick/slip prediction. In *IADC/SPE International Drilling Conference and Exhibition*. OnePetro, 2022.

Anthony Sicilia, Xingchen Zhao, and Seong Jae Hwang. Domain adversarial neural networks for domain generalization: When it works and how to improve. *Machine Learning*, 112(7):2685–2721, 2023.

Peeyush Singhal, Rahee Walambe, Sheela Ramanna, and Ketan Kotecha. Domain adaptation: challenges, methods, datasets, and applications. *IEEE access*, 11:6973–7020, 2023.

Ralf C Staudemeyer and Eric Rothstein Morris. Understanding lstm–a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:1909.09586*, 2019.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and S Yu Philip. Generalizing to unseen domains: A survey on domain generalization. *IEEE transactions on knowledge and data engineering*, 35(8):8052–8072, 2022.

Xingxing Weng, Yuchun Huang, Yanan Li, He Yang, and Shaohuai Yu. Unsupervised domain adaptation for crack detection. *Automation in Construction*, 153:104939, 2023.

Ruihao Xia, Chaoqiang Zhao, Meng Zheng, Ziyan Wu, Qiyu Sun, and Yang Tang. Cmda: Cross-modality domain adaptation for nighttime semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 21572–21581, 2023.

Brian Xu, Mitra Mohtarami, and James Glass. Adversarial domain adaptation for stance detection. *arXiv preprint arXiv:1902.02401*, 2019.

Hana Yahia, Thomas Romary, Laurent Gerbaud, Bruno Figluizzi, Florent Di Meglio, Stephane Menand, and Mohamed Mahjoub. Combining machine-learning and physics-based models to mitigate stick-slip in real-time. In *SPE/IADC Drilling Conference and Exhibition*, page D031S022R003. SPE, 2024a.

Hana Yahia, Thomas Romary, Laurent Gerbaud, Stephane Menand, and Mohamed Mahjoub. Real-time stick-slip mitigation using combined machine learning and physics based techniques. In *International Petroleum Technology Conference*, page D031S099R001. IPTC, 2024b.

AS Yigit and AP Christoforou. Coupled torsional and bending vibrations of drillstrings subject to impact with friction. *Journal of Sound and Vibration*, 215(1):167–181, 1998.

Yang Zha and Son Pham. Monitoring downhole drilling vibrations using surface data through deep learning. In *SEG Technical Program Expanded Abstracts 2018*, pages 2101–2105. Society of Exploration Geophysicists, 2018.

Chao Zhao and Weiming Shen. Adversarial mutual information-guided single domain generalization network for intelligent fault diagnosis. *IEEE Transactions on Industrial Informatics*, 19(3):2909–2918, 2022.

Huailiang Zheng, Rixin Wang, Yuantao Yang, Yuqing Li, and Minqiang Xu. Intelligent fault identification based on multisource domain generalization towards actual diagnosis scenario. *IEEE Transactions on Industrial Electronics*, 67(2):1293–1304, 2019.

Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4396–4415, 2022.

Xiaohua Zhu, Liping Tang, and Qiming Yang. A literature review of approaches for stick-slip vibration suppression in oilwell drillstring. *Advances in Mechanical Engineering*, 6:967952, 2014.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.