# Negation Invariant Representations of 3D Vectors for Deep Learning Models applied to Fault Geometry Mapping in 3D Seismic Reflection Data

Daniel Kluvanec, Kenneth J. W. McCaffrey, Thomas B. Phillips, Noura Al Moubayed

***Abstract*—We can represent the orientation of a plane in 3D by its normal vector. However, every plane has two normal vectors that are negatives of each other. We propose four novel representations of vectors in 3D that are negation invariant and can be used by a neural network to predict orientation. Our proposed solution is the first to introduce representations that are negation invariant, continuous and easily parallelisable on the GPU. We evaluate the representations by predicting the orientation of a plane on a toy task, and by applying them to synthetic seismic tomographic data where we predict the presence and orientation of faults for every voxel in the volume. We further make use of the orientation of the faults in a post-processing algorithm on the GPU that separates the faults into segments (i.e. instances) that do not intersect, which allows us to selectively visualise faults in 3D. We demonstrate the utility of the representations by deploying the model on the Laminaria 3D Seismic volume as a case study. We quantitatively compare the model's prediction against human interpretations of slices through the volume as well as existing interpretations in literature. Our analysis shows good agreement (F1 score of 88%) of the model with human interpretation in the shallow levels, where the ambient noise is lower, but this agreement degrades at deeper levels (F1 score of 68%). We explore possible reasons for this degradation.**

***Index Terms*—Deep Learning, Representations, Invariance, Fault Geometry Detection, 3D Image Processing, Semantic Segmentation, Seismic Reflection**

## I. Introduction

Deep Learning methods can be applied to a wide range of tasks across a variety of data domains. A neural network interacts with any data represented as vectors. The choice of representation plays a crucial role in the functionality of the network and determines the task for which a model can be used. Common tasks include classification, bounding box prediction, pose estimation, regression, image denoising, audio synthesis or text summarisation. Some representations may perform better than others, such as representations with values normalised to a standard normal distribution. It is also beneficial to avoid representations with discontinuities, where a change in the represented attribute can lead to a disproportional change in the representation.

When representing cyclical properties such as angles that repeat every $360°$ a commonly used representation is the sine and cosine of the angle, rather than predicting the angle directly. This avoids problems of having a discontinuity at $0°$ and $360°$, where similar angles are represented by distant values. However, there is no simple representation that is continuous and maps any vector in 3D and its negative to the same point. In this paper we explore the possible representations for vectors in 3D that are invariant to negation.

A representation for vectors that is invariant to negation can be used to represent the orientation of any object where we do not distinguish one of its directions. For instance: it can represent the orientation of a plane (flat surface) in 3D that has two normal vectors which are negatives of each other, direction of a road or a land border, or the orientation of symmetrical objects. We focus on the application of predicting the orientation of a plane in 3D and evaluate the representations on a toy dataset, consisting of an 8x8x8 volumes within which we determine the orientation of a plane. Subsequently, we move onto the real application on seismic data.

We propose four novel representations of negation invariant vectors in 3D. Two representations are trained as a regression using the mean-squared-error loss and two are inspired by classification and make use of a softmax activation function and a cross-entropy loss. We show the effectiveness of all four representations and identify the Projection-Doubleangle representation as the best, which we then use in our case study on real seismic data.

We train our seismic model on synthetic data, on which we also evaluate the representations. We predict the following features for every voxel in the volume: whether a fault is present near the voxel, whether more than one fault is present near the voxel, the orientation of the fault (this output is ignored if no fault or more than one fault are present). We also deploy the model trained on synthetic data on a real seismic volume imaged in the Laminaria 3D seismic volume. We make use of the predicted fault orientation to distinguish chaotic regions in the prediction that occur near areas of intersection between faults and use it to separate the network of faults into separate fault instances that do not intersect. We then compare the predicted faults to those obtained in previous studies [1] and compare their orientations. Finally, given that deeper levels in seismic volumes have more inherent noise, we examine qualitatively how the reliability of the modelled faults varies with depth through the Laminaria 3D seismic volume.

You can find the code for our proposed representations implemented in Python using PyTorch on Github[1].

Daniel Kluvanec - Dept. of Computer Science, Durham University, UK
Kenneth J.W. McCaffrey - Dept. of Earth Sciences, Durham University, UK
Thomas B. Phillips - Dept. of Earth Sciences, Durham University, UK
Noura Al Moubayed - Dept. of Computer Science, Durham University, UK

[1] https://github.com/KluvaDa/NegInvVector3D

## II. Related Work

### A. Representations

A related field focused on predicting the orientation of objects is 6D pose estimation, where both the location and orientation of an object are predicted. Symmetrical objects pose a particular challenge, since they look similar in multiple orientations. One way of addressing the issue is by creating representations that are ambiguity aware and deliberately do not distinguish different symmetries of the same object from each other.

In 2D, angles are commonly represented using their sine and cosine which avoids a discontinuity at $0°$ and $360°$ [2]–[4]. This representation can easily be extended to cover rotational symmetries by taking the sine and cosine of a multiple of the angle. In the case of $180°$ symmetry we can use the sine and cosine of two times the angle [5], [6]. We refer to this representation as the Double-Angle representation and define it in Equations 6, 7.

Unlike in our task, where we only consider the negation of a vector corresponding to $180°$ rotational symmetry or planar symmetry, 6D pose estimation concerns itself with objects that contain a number of other symmetries as well. A few approaches tackle this problem [7], [8] by creating a general solution addressing a broad range of symmetries that are needlessly complicated for our purposes.

Saxena et al. [9] propose a framework for representing the orientation of objects with numerous symmetries. Part of their framework can be used to just represent vectors and their negatives as the same point by encoding a vector $\mathbf{v} \in \mathbb{R}^{3\times1}$ as $R = \mathbf{v}\mathbf{v}^T \in \mathbb{R}^{3\times3}$. To invert the representation, the eigenvector corresponding to the largest eigenvalue of the matrix $R$ is taken. Finding the eigenvector is, however, computationally expensive even when using approximations [10]. We refer to this representation as Saxena's representation.

### B. Fault Segmentation from Seismic Volumes

In order to determine the orientation of faults in seismic volumes, we must first determine the location of the faults. This can be done by classifying every voxel in the volume as a semantic segmentation. Wu et al. [11] use a Convolutional Neural Network inspired by the UNet architecture [12], while using 3D convolutional layers to work on volumes rather than images. In order to train this FaultSeg3D model, they create a synthetic dataset using a similar method to Wu and Hale [13], which they further explained in a later publication [14]. They demonstrate that their method translates well from synthetic data to real seismic volumes where it correctly annotates faults.

Gato et al. [15] propose the use of a new nested Unet architecture trained on Wu et al.'s synthetic dataset. They achieve better results, especially in noisy areas. Dou et al. [16] also propose a new model architecture and a new loss function which can be used to train the model on sparsely labelled data and makes it feasible to train on real seismic data. Feng et al. [17] used Monte-Carlo sampling in the form of dropout at evaluation time to sample multiple possible network predictions and determine the certainty of the model. Hu et al. [18] turn to a different convolutional architecture, which, unlike the UNet, does not down-sample and up-sample the data. Instead it uses dilated convolutions and atrous spacial pyramid pooling to analyse the data at multiple resolutions. An et al. [19], [20] published a real-world dataset with partial annotations of faults together with a method that can train directly from the data without requiring synthetic data. Wrona et al. [21] published a framework complete with the source code for predicting faults, salt bodies and horizons. They make use of the UNet architecture for predicting pixel-wise outputs.

When focusing on faults, the aforementioned methods only predict the locations where faulting occurred. In addition to these outputs, some methods also predict the geometry of the faults. Wu et al. [22] framed the task as a classification, in which the 3D variant, they separate the fault's dip and strike angles into bins of $3°$ and $5°$ respectively. The combination of these two features form a total of 576 classes, together with a final class for no fault being present. Their model classifies the whole input patch based on the central pixel, but could be reformulated into an equivalent fully convolutional architecture that would perform a semantic segmentation.

Wu et al. [23] predict the faulting geometry as one of the outputs from their multi-task model. They represent the geometry in the form of a vector field which is normal to the fault planes, similarly to our Z-Aligned-Vector representation, and train it using a normalised cosine-similarity function. The network outputs these values for every voxel in the volume in a manner similarly to a semantic segmentation.

To further analyse the geometry of the faults, some methods resort to modelling the faults as surfaces [24], [25]. When applied on top of the predictions of the neural networks, these methods use graph theory to transform the predicted voxel data into surfaces. However, unlike our post-processing method that separates faults using their orientation, these methods cannot be easily parallelised on the GPU.

### C. Case Study

We have chosen the Laminaria 3D seismic volume for our case study because it is a well-known high quality dataset with clear fault structures [26]. The seismic volume was acquired in the Australian North West shelf and imaged thick Paleozoic, Mesozoic and Cenozoic sedimentary sequences deposited in a series of rifting phases [27]. The dominant structural trend including the Laminaria high is ENE-WSW with a more E-W trend at deeper levels [1], [26].

## III. Representations

To use Deep Learning to predict some attribute, the attribute must be expressed as a vector, which is then output by the neural network. We call this vector the representation of the attribute. To represent the orientation of a plane, we can use a unit vector $\mathbf{v}$ that is normal to the plane. However, we do not distinguish the two faces of the plane, leading to two distinct normal vectors: $\mathbf{v}$ and $-\mathbf{v}$. This ambiguity is problematic and we argue that for effective use in neural networks the representations need to be both unique and continuous. In other words, if the represented attribute changes by a small amount, the representation of the attribute should also only change

by a small amount. In this section we define some common equations and notation, before defining the representations. We start with existing representations (subsections A-C) that are problematic in various ways before defining our proposed representations (subsections D-G).

$$\|\mathbf{u}\| = \sqrt{u_0^2 + u_1^2 + ... + u_n^2} \tag{1}$$

is the magnitude of the vector $\mathbf{u} \in \mathbb{R}^n$ with $u_i$ being the $\mathrm{i}^{th}$ element of the vector.

$$\operatorname{sign}(a) = \begin{cases} 1 & \text{if } a \geq 0 \\ -1 & \text{if } a < 0 \end{cases}. \tag{2}$$

We define the *dip* and *strike* angles associated with the normal unit vector $\mathbf{v}$: ($\operatorname{atan2}$ is the 2-argument arctangent)

$$\textit{dip} = \operatorname{acos}(v_z), \tag{3}$$

$$\textit{strike} = \operatorname{atan2}(-v_y, v_x), \tag{4}$$

$$\mathbf{v} = \begin{bmatrix} \cos(\textit{strike}) \cdot \sin(\textit{dip}) \\ -\sin(\textit{strike}) \cdot \sin(\textit{dip}) \\ \cos(\textit{dip}) \end{bmatrix}. \tag{5}$$

The Double-Angle representation [5], [6] can be expressed in two forms. Firstly, to represent angles with $180°$ periodicity as follows:

$$\mathbf{r} = \begin{bmatrix} r_0 \\ r_1 \end{bmatrix} = \begin{bmatrix} \cos(2\theta) \\ \sin(2\theta) \end{bmatrix}, \qquad \theta = \frac{1}{2}\operatorname{atan2}(r_1, r_0). \tag{6}$$

Secondly, to represent vectors $\mathbf{v} \in \mathbb{R}^2$ with negation invariance as follows:

$$\mathbf{r} = \|\mathbf{v}\| \begin{bmatrix} \cos\big(2\operatorname{atan2}(v_1, v_0)\big) \\ \sin\big(2\operatorname{atan2}(v_1, v_0)\big) \end{bmatrix} = \frac{1}{\|\mathbf{v}\|} \begin{bmatrix} v_0^2 - v_1^2 \\ 2v_0^2 v_1^2 \end{bmatrix},$$

$$\mathbf{v}' = \|\mathbf{r}\| \begin{bmatrix} \cos\big(\frac{1}{2}\operatorname{atan2}(r_1, r_0)\big) \\ \sin\big(\frac{1}{2}\operatorname{atan2}(r_1, r_0)\big) \end{bmatrix} = \|\mathbf{r}\| \begin{bmatrix} \sqrt{\frac{1}{2} + \frac{r_0}{2\|\mathbf{r}\|}} \\ \operatorname{sign}(r_1)\sqrt{\frac{1}{2} - \frac{r_0}{2\|\mathbf{r}\|}} \end{bmatrix}. \tag{7}$$

Note that the inverted vector $\mathbf{v}'$ is equal to either $\mathbf{v}$ or $-\mathbf{v}$.

### A. Z-Aligned-Vector Representation

We can remove the ambiguity and define a unique representation by ensuring that the z-axis of the vector $\mathbf{v}$ is greater than zero by multiplying $\mathbf{v}$ by $\pm 1$ accordingly. Although this representation is unique, it is not continuous, because of the change of the sign near values where $v_z = 0$. This is problematic, because the neural network has to learn to strongly differentiate two vectors that are similar. We demonstrate the problems with this representation in Section IV-A.

We predict this representation using 3 features with a linear activation function and train it using the Mean Squared Error as a loss function.

### B. Dip and Strike Representation

When working with faults, their geometry is commonly defined using dip and strike angles. To define a unique representation we can reduce the allowed angles to one of the following two ranges: *dip* $\in [0°, 90°]$, *strike* $\in [0°, 360°)$, or to *dip* $\in [-90°, 90°]$, *strike* $\in [0°, 180°)$. These ranges are visualised in Fig. 1. Even though we can define these angles using their sine and cosine values, it does not fully remove the discontinuity in the representation. Moving one angle past its boundary will cause a discontinuous shift in the other angle, which we demonstrate experimentally in Section IV-A.

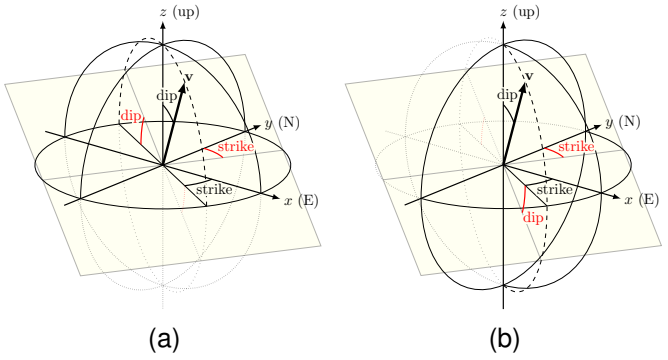


Fig. 1. Diagram showing the definition of dip and strike angles of a plane defined using its normal unit vector $\mathbf{v} = (x, y, z)$, as *dip* = $\operatorname{acos}(z)$ and *strike* = $\operatorname{atan2}(x, -y)$. The following two domains are visualised:
(a) *dip* $\in [0°, 90°]$, *strike* $\in [0°, 360°)$;
(b) *dip* $\in [0°, 180°]$, *strike* $\in [-90°, 90°)$.

The Dip90° Strike360° representation normalises the dip angles from a range of $[0°, 90°]$ to $[0, 1]$ and predicts the strike angle using $\sin(\textit{strike})$ and $\cos(\textit{strike})$. The 3 values are predicted with a linear activation and trained using the Mean Squared Error.

The Dip180° Strike180° representation normalises the dip angles from a range of $[-90°, 90°]$ to $[-1, 1]$ and predicts the strike angle using $\sin(2 \cdot \textit{strike})$ and $\cos(2 \cdot \textit{strike})$ (the 2D doubleangle representation). The 3 values are predicted with a linear activation and trained using the Mean Squared Error.

### C. Saxena's Representation

An existing representation that we compare against is proposed by Saxena et al. [9]. It encodes the vector $\mathbf{v}$ as the $3 \times 3$ matrix $\mathbf{v} \cdot \mathbf{v}^T$. As a symmetrical matrix, it consists of 6 unique values that have to be predicted. To invert the representation, the eigenvector corresponding to the largest eigenvalue of the matrix is found using Principal Component Analysis [10]. The downside is that PCA is an expensive operation, especially since it cannot be easily performed on a GPU.

We predict this representation using 6 features with a linear activation function and train it using the Mean Squared Error as a loss function.

### D. Piecewise-Aligned Representation

This is our first proposed representation. It is based on aligning the vector $\mathbf{v}$ with different axes. If we ensure that the x-axis is positive, discontinuities will occur near $x = 0$. The same principle applies for the y and z axes. If we create three copies of the vector $\mathbf{v}$: $\mathbf{v}_x$, $\mathbf{v}_y$ and $\mathbf{v}_z$, and align each vector with a different axis, then at least one of the vectors will be far from its respective discontinuity. We then modify the magnitude of each vector as follows: $v_x^2\mathbf{v}_x$, $v_y^2\mathbf{v}_y$ and $v_z^2\mathbf{v}_z$.

This way we remove the discontinuity from each vector. To invert the representation, we align all vectors with the same axis and add them.

*Definition:*

In this representation we make three copies of the input vector $\mathbf{v} \in \mathbb{R}^3$ with a magnitude of 1: $\mathbf{v}_x$, $\mathbf{v}_y$, and $\mathbf{v}_z$. We then apply the following equation to each vector with its respective axis symbolised as $a \in \{x, y, z\}$:

$$\mathbf{r}_a \in \mathbb{R}^3 = v_a^2 \cdot \text{sign}(v_a) \cdot \mathbf{v}. \tag{8}$$

The representation is then defined by concatenating the three vectors: $\mathbf{r} = [\mathbf{r}_x, \mathbf{r}_y, \mathbf{r}_z]$.

To invert the representation we need to align each vector $\mathbf{r}_x, \mathbf{r}_y, \mathbf{r}_z$ to the same axis $b$. We select this axis $b$ such that $\max(\|\mathbf{r}_x\|, \|\mathbf{r}_y\|, \|\mathbf{r}_z\|) = \|\mathbf{r}_b\|$. The represented vector is then defined as:

$$\mathbf{v}' = \sum_{a \in \{x,y,z\}} \text{sign}(r_{a,b}) \cdot \mathbf{r}_a, \tag{9}$$

where $r_{a,b}$ is the b$^{th}$ element of $\mathbf{r}_a$. Finally, we divide $\mathbf{v}'$ by its magnitude to ensure that it is a unit vector.

We predict this representation using 9 features with a linear activation function and train it using the Mean Squared Error as a loss function.

### *E. Projection-Doubleangle Representation*

Our second proposed representation works by projecting the vector $\mathbf{v}$ onto the three orthogonal planes $x = 0$, $y = 0$ and $z = 0$, as seen in Fig. 2. Their orientation on the 2D plane can then be encoded using the Double-Angle representation, which encodes a negation invariant 2D vector. We define it in Equations 6, 7. This way we lose the information about the exact sign of the projected vector elements, but we retain information about whether the two axes have the same sign or not. This is sufficient information to recreate the original vector multiplied by an arbitrary sign.

*Definition:*

Take the input unit vector $\mathbf{v} \in \mathbb{R}^3$ and project it onto three different planes to get $\mathbf{p}_{ab} = [v_a, v_b]$, where $a$ and $b$ correspond to two axes from $\{x, y, z\}$. $\mathbf{p}_{ab}$ is then encoded using the doubleangle representation from equation 7 to get the representation:

$$\mathbf{r}_{ab} = \frac{1}{\sqrt{v_a^2 + v_b^2}} \begin{bmatrix} v_a^2 - v_b^2 \\ 2v_a^2 v_b^2 \end{bmatrix}. \tag{10}$$

The full representation is then formed by concatenating the three vectors: $\mathbf{r} = [\mathbf{r}_{yz}, \mathbf{r}_{xz}, \mathbf{r}_{yz}]$.

To invert the representation we first find the projected vector $\mathbf{p}'_{ab}$ using Equation 7:

$$\mathbf{p}'_{ab} = \|\mathbf{r}_{ab}\| \begin{bmatrix} \sqrt{\frac{1}{2} + \frac{r_{ab,0}}{2\|\mathbf{r}\|}} \\ \text{sign}(r_{ab,1}) \sqrt{\frac{1}{2} - \frac{r_{ab,0}}{2\|\mathbf{r}\|}} \end{bmatrix}, \tag{11}$$

where $r_{ab,i}$ is the i$^{th}$ element of $\mathbf{r}_{ab}$. $p_{ab,0}$ then corresponds to the vector element on axis $a$ and $p_{ab,1}$ on axis b. We can then

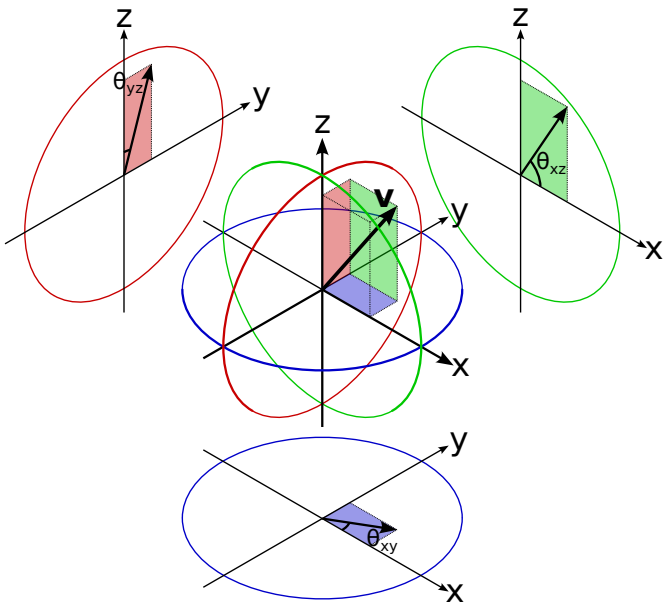


Fig. 2. How angles $\theta_{yz}$, $\theta_{xz}$, and $\theta_{xy}$ are measured for the Projection-Doubleangle representation for any unit vector $\mathbf{v}$.

take the two elements corresponding to each axis and average them, if we correctly determine its sign as follows:

$$\mathbf{v}' = \begin{bmatrix} \frac{1}{2} \cdot (s_{xz} \cdot p'_{xz,0} + s_{xy} \cdot p'_{xy,0}) \\ \frac{1}{2} \cdot (s_{yz} \cdot p'_{yz,0} + s_{xy} \cdot p'_{xy,1}) \\ \frac{1}{2} \cdot (s_{yz} \cdot p'_{yz,1} + s_{xz} \cdot p'_{xz,1}) \end{bmatrix}, \tag{12}$$

where $s_{ab} \in \{-1, 1\}$. We want the corresponding axis elements to have the same sign, which may not be possible due to model inaccuracies. In that case, we take the axis with the smallest absolute values and allow it to have different signs. Let:

$$\mathbf{m} = \begin{bmatrix} |p'_{xz,0}| + |p'_{xy,0}| \\ |p'_{yz,0}| + |p'_{xy,1}| \\ |p'_{yz,1}| + |p'_{xz,1}| \end{bmatrix}, \tag{13}$$

and let $k \in \{x, y, z\}$ be the axis with the smallest value of $m_k$. The sign is then defined as

$$s_{ab} = \begin{cases} \begin{cases} 1 & \text{if } \text{sign}(p_{ab,1}) = \text{sign}(p_{bc,0}) \\ -1 & \text{if } \text{sign}(p_{ab,1}) \neq \text{sign}(p_{bc,0}) \end{cases} & \text{if } k = a \\ \begin{cases} 1 & \text{if } \text{sign}(p_{ab,0}) = \text{sign}(p_{ac,0}) \\ -1 & \text{if } \text{sign}(p_{ab,0}) \neq \text{sign}(p_{ac,0}) \end{cases} & \text{if } k = b \\ 1 & \text{if } k = c \end{cases}. \tag{14}$$

We predict this representation using 6 features with a linear activation function and train it using the Mean Squared Error as a loss function.

### *F. Classification-Dip-Strike Representation*

Our third and fourth representations are both inspired by classification tasks, which are expanded to be continuous. Rather than classifying the vector into distinct and unique classes based on its orientation, we partially assign multiple

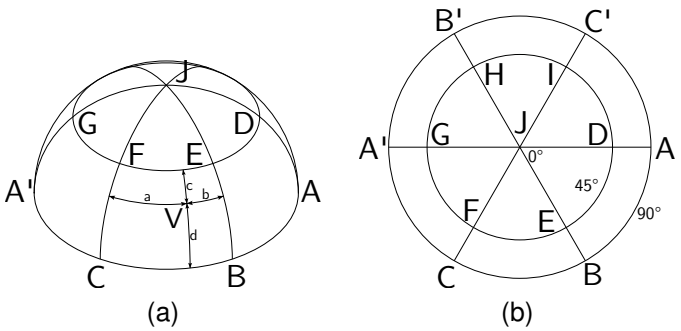


Fig. 3. Diagram showing the points corresponding to categories in the Classification-Dip-Strike representation. (b) is a top-down view of (a).

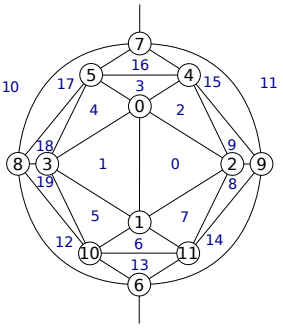


Fig. 4. A diagram showing which vertices belong to which face of the icosahedron.

classes to the vector based on how close it is to the centre-points of the classes. We ensure that all the class weights add up to 1, allowing us to use the softmax activation function similarly to a classification.

This third proposed representation classifies the vector based on its polar coordinates, or the dip and the strike of the plane normal to the vector (shown in red) as seen in Fig. 1. The categories are centered around the points seen in Fig. 3. Note that due to our desired negation invariance, points on the opposite side of the sphere are treated as the same class. We then calculate the partial assignment values as the similarity in the dip multiplied by the similarity in the strike between the vector and a class. For instance, in the example shown in Fig. 3, classes E, F, B and C have non-zero weights and class E's weight is calculated as:

$$\frac{d}{c+d} \cdot \frac{b}{a+b} \,.$$

*Definition:*

We define the centre-points of categories as seen in Fig. 3, as the following:

| Point | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| $Dip$ | 90° | 90° | 90° | 45° | 45° | 45° | 45° | 45° | 45° | 0° |
| $Strike$ | 0° | 60° | 120° | 0° | 60° | 120° | 180° | 240° | 300° | n/a |

Each point $X$ also has a point on the opposite side, defined as $X'$ with $X'_{dip} = 180° - X'_{dip}$ and $X'_{strike} = X_{strike} + 180° \mod 360°$. However, the model will only predict the larger class weight of each pair.

If a represented vector has a dip between $45°$ and $90°$, we use the following equations to determine the class weights for the example on face EFBC as seen in Fig. 3:

$$\begin{aligned} w_{\text{dip}} &= \frac{c}{c+d} \,, \\ w_{\text{strike}} &= \frac{a}{a+b} \,, \\ w_E &= (1-w_{\text{dip}}) \cdot (1-w_{\text{strike}}), \\ w_F &= (1-w_{\text{dip}}) \cdot w_{\text{strike}}, \\ w_B &= w_{\text{dip}} \cdot (1-w_{\text{strike}}), \\ w_C &= w_{\text{dip}} \cdot w_{\text{strike}}. \end{aligned} \tag{15}$$

If the vector has a dip between $0°$ and $45°$, such as the face JEF, we use the following equation:

$$\begin{aligned} w_J &= (1-w_{\text{dip}}), \\ w_E &= w_{\text{dip}} \cdot (1-w_{\text{strike}}), \\ w_F &= w_{\text{dip}} \cdot w_{\text{strike}}. \end{aligned} \tag{16}$$

To invert the representation, we first determine which face the represented vector lies on. This is the face with the largest sum of its vertices' weights. We assign the same weight to vertices on opposite sides of the sphere, such as vertices $A$ and $A'$. We then use the following equations for faces with 4 vertices, such as face EFBC:

$$\begin{aligned} \text{dip}'_{\mathbf{v}} &= \text{dip}_B + (\text{dip}_E - \text{dip}_B) \cdot \frac{w_E + w_F}{w_E + w_F + w_B + w_C}, \\ \text{strike}'_{\mathbf{v}} &= \text{strike}_B + (\text{strike}_C - \text{strike}_B) \cdot \frac{w_C + w_F}{w_E + w_F + w_B + w_C}. \end{aligned} \tag{17}$$

If the face has 3 vertices, such as EFJ, then we use the following:

$$\begin{aligned} \text{dip}'_{\mathbf{v}} &= \text{dip}_E + (\text{dip}_J - \text{dip}_E) \cdot \frac{w_J}{w_E + w_F + w_J}, \\ \text{strike}'_{\mathbf{v}} &= \text{strike}_E + (\text{strike}_F - \text{strike}_E) \cdot \frac{w_F}{w_E + w_F}. \end{aligned} \tag{18}$$

We predict this representation using 10 features with a softmax activation function and train it using the Cross-Entropy loss. Note that the Cross-Entropy loss has to be defined in a way that allows for multiple non-zero target values.

### G. Classification-Icosahedron Representation

Our fourth proposed representation is also inspired by classification tasks. Unlike the Classification-Dip-Strike representation, this representation centres its categories on the vertices of an icosahedron, evenly distributing its classes on the sphere.

*Definition:*

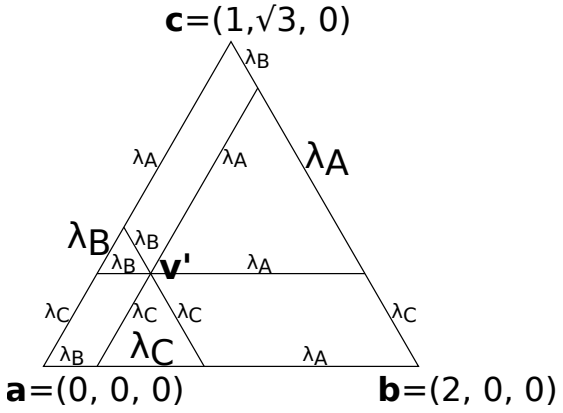


Fig. 5. Faces of an icosahedron are mapped to the displayed triangle, within which we can calculate the similarity metric for the representation: Icosahedron using Basis Vector Coefficients.

We define the vertices of an icosahedron as the centre-points of our categories with the coordinates:

$$\begin{aligned}
\mathbf{c}_0 &= (0, 1, \varphi), & \mathbf{c}_6 &= (0, -1, -\varphi),\\
\mathbf{c}_1 &= (0, -1, \varphi), & \mathbf{c}_7 &= (0, 1, -\varphi),\\
\mathbf{c}_2 &= (\varphi, 0, 1), & \mathbf{c}_8 &= (-\varphi, 0, -1),\\
\mathbf{c}_3 &= (-\varphi, 0, 1), & \mathbf{c}_9 &= (\varphi, 0, -1),\\
\mathbf{c}_4 &= (1, \varphi, 0), & \mathbf{c}_{10} &= (-1, -\varphi, 0),\\
\mathbf{c}_5 &= (-1, \varphi, 0), & \mathbf{c}_{11} &= (1, -\varphi, 0),
\end{aligned} \tag{19}$$

where $\varphi = \frac{1+\sqrt{5}}{2}$ is the golden ratio. Note that the magnitude of these vectors is not 1. Also note that $\mathbf{c}_n = -\mathbf{c}_{n+6}$. This allows us to unify the categories that are opposite each other to achieve negation invariance.

We define the faces $f_m$ of the icosahedron as a list of indices $n$ of the vertices $\mathbf{c}_n$ that belong to the face. The first 10 faces are defined as:

$$\begin{aligned}
f_0 &= [1, 0, 2] & f_5 &= [1, 3, 10]\\
f_1 &= [0, 1, 3] & f_6 &= [10, 11, 1]\\
f_2 &= [0, 2, 4] & f_7 &= [1, 2, 11]\\
f_3 &= [4, 5, 0] & f_8 &= [2, 9, 11]\\
f_4 &= [0, 3, 5] & f_9 &= [2, 9, 4]
\end{aligned} \tag{20}$$

while the remaining 10 faces are identical, only using the opposite vertices. The relationship between faces and vertices can be seen in Fig. 4.

We first determine which face a vector belongs to, by finding the similarity $s_n$ between the vector $\mathbf{v}$ and all the vertices $\mathbf{c}_n$ using the equation:

$$s_n = \left| \frac{\mathbf{v} \cdot \mathbf{c}_n}{\|\mathbf{v}\| \cdot \|\mathbf{c}_n\|} \right|. \tag{21}$$

The face $f_m$ with the largest sum of similarities is then selected. We then find an euclidean transformation from this face to a normalised face with coordinates: $\mathbf{a} = (0, 0, 0)$, $\mathbf{b} = (2, 0, 0)$, $\mathbf{c} = (1, \sqrt{3}, 0)$ as seen in Fig. 5. This transformation then maps the vector $\mathbf{v}$ onto vector $\mathbf{u}$, and we set the z-axis element of $\mathbf{u}$ to zero so that it lies on the face. We can then express the class weights as:

$$\begin{aligned}
w_a &= 1 - \frac{\lambda_b}{2} - \frac{\lambda_c}{2} = 1 - \frac{x}{2} - \frac{y}{2},\\
w_b &= 1 - \frac{\lambda_a}{2} - \frac{\lambda_c}{2} = \frac{x}{2} - \frac{y}{2\sqrt{3}},\\
w_c &= 1 - \frac{\lambda_a}{2} - \frac{\lambda_b}{2} = \frac{y}{\sqrt{3}}.
\end{aligned} \tag{22}$$

To invert the representation, we first determine which face has the largest sum of its vertices' weights. We then take those weights and use them to calculate the vector $u'$ in the normalised space as follows:

$$\mathbf{u}' = \begin{bmatrix} 1 - w_a + w_b \\ \sqrt{3} \cdot w_c \\ 0 \end{bmatrix}. \tag{23}$$

Next, use the inverse euclidean matrix to transform $\mathbf{u}'$ in the normalised space to $\mathbf{v}'$ on the surface of the icosahedron. Finally, we normalise the magnitude of $\mathbf{v}'$ to make it a unit vector.

We predict this representation using 6 features with a softmax activation function and train it using the Cross-Entropy loss. Note that the Cross-Entropy loss has to be defined in a way that allows for multiple non-zero target values.

## IV. Methods

We evaluate the representations on three different levels. We need to evaluate the representations in a controlled setting that equally considers all possible orientations. We designed the toy task for this purpose. At the same time, the toy task least resembles true applications. The synthetic data provides a balance between realism and having ground truths. It has fault surfaces that are flat, but they do not cover the whole space of possible orientations, more closely resembling the distribution of orientations of real faults. We train the same model for both the synthetic and real data tasks. The real data is where we ultimately want the representations to perform well on, however we cannot quantitatively evaluate the representations on it, since we haven't got ground truth values to compare the predictions to. Note that we train this model using the synthetic data.

### A. Toy Task

The toy task dataset consists of volumes of size $8 \times 8 \times 8$, which contain a pixelated plane that goes through the centre of the volume and is in any orientation. All voxels with centre-points within a distance of 0.6 of the plane are given a value of 1, while the background has a value of 0. An example can be seen in Fig. 6.

We train a tiny neural network seen in Figure 7 to predict the orientation of the plane in the volume using one of the representations. We generate the volumes at runtime for any given orientation, so there is no notion of training/testing dataset splits. For both training and testing we sample from the same space of all possible orientations.

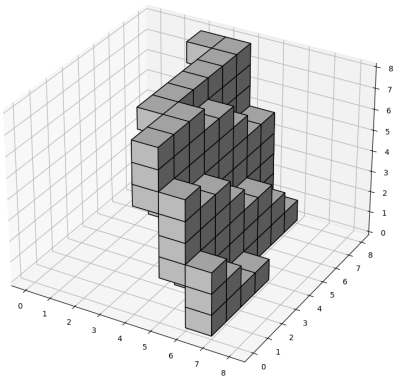

Fig. 6. An example of a volume from the toy dataset which represents a pixelated plane that we predict the orientation of.

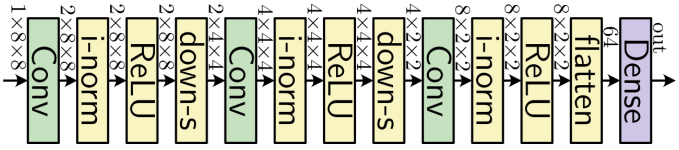


Fig. 7. The neural network architecture used for the toy task. Conv is a 3D convolutional layer with a $3 \times 3 \times 3$ kernel, padding of 1 and a strike of 1. i-norm is instance normalisation, down-s is $2\times$ downsampling using max pooling, Dense is a fully connected layer.

### *B. Training on Synthetic Seismic Volumes*

To evaluate the representations further, we turn to the task of predicting the location and orientation of faults present in a synthetic seismic volume. We use a proprietary dataset of synthetic seismic volumes, which is based on the method of Wu [13], [14]. The dataset consists of 1536 synthetic volumes with a resolution of $96 \times 96 \times 96$. An example is visualised in Fig. 8. Synthetic data provides us with the annotations required to train and evaluate the performance of the model and the different representations. We split the synthetic dataset into a training and testing set with a $80\% - 20\%$ split.

We train a neural network to predict whether a fault is present in any voxel as well as its orientation using one of the representations. These features are predicted as separate channels for every voxel in the volume. The fault presence is predicted as a single feature with a sigmoid activation function and is trained using the dice loss [28]. If a fault is not present in a voxel, we ignore the orientation output for that voxel.

Our neural network architecture is visualised in Figure 9. The network uses 3D convolutions inside HarDNet blocks [29] within the structure of a UNet [12]. With the model architecture being fully convolutional, it can be deployed on volumes of any size (that is a multiple of 8), with GPU VRAM being the only limiting factor. We trained the model on an NVidia Titan RTX with 24GB of VRAM, using mixed (16 and 32 bit) floating point precision and a batch size of 4.

### *C. Deploying on Real Seismic Data*

Finally, we evaluate the model on a real dataset, the Laminaria 3D seismic volume. Because of a lack of ground truth labels for the real dataset, we cannot train a model using this data, nor can we quantitatively evaluate the performance of the representations. Instead, we use a model trained on synthetic data and analyse the results on the Laminaria volume in the form of a case study. We perform a post-processing step that makes use of the network predictions and separates the faults into segments that do not intersect. These fault segments can then be selectively visualised in 3D, as seen in Fig. 10. We analyse the model's performance by comparing it against manual interpretations by a geologist (McCaffrey), the interpretations of Phillips et al. [1] and Cifci & Langhi [26].

Our post-processing algorithm allows us to make use of the orientation of the faults and separate them into distinct fault segments that do not intersect, aka instances. In conjunction with the fault probability and orientation, we also predict the following values using the model, which we use in the post-processing algorithm:

1) At least one fault is present (fault probability)
2) More than one fault is present (intersection probability)
3) Exactly one fault is present (segment probability)
4) The orientation of the fault normal (only if 3 is True)
5) The orientation of the intersection, which is perpendicular to the intersecting faults' normals (only if 2 is True)

The post-processing algorithm takes the predicted fault locations and removes areas from the volume until its fault segments are disjointed in space. These are then enumerated and grown out to cover the original predicted volume. The steps of the algorithm are visualised in Figure 11 and explained below.

1) Threshold the fault probability, intersection probability, and segment probability values. Morphologically dilate the intersection volume and subtract it from the fault segment volume. The resultant binary volume of segments will further be refined until the segments are separated in space.
2) Subtract areas with inconsistent fault orientation from the segments, measured as the absolute value of the dot product between normals in a neighbourhood.
3) Connected smoothing: Only keep segment voxels that have sufficient neighbours as segment voxels. This removes noisy parts of the volume and smooths out the segments.
4) Shrink the fault segments along the fault plane, which avoids thinning them. For every segment voxel, consider the points in its neighbourhood that are perpendicular to the orientation expressed as a normal vector to the fault plane. Of the considered points, count the number labelled as a segment and only keep the points where this count is higher than a threshold. Repeat this process 3 times.
5) Connected smoothing: Repeat the same operation from point 3.
6) We proceed to label each distinct segment with a unique integer. To do this in a GPU parallelisable manner, we assign every segment voxel in the volume a unique integer and assign non-segment voxels a very large integer. We then iteratively apply the min operation over segment voxels in a local neighbourhood until no more changes

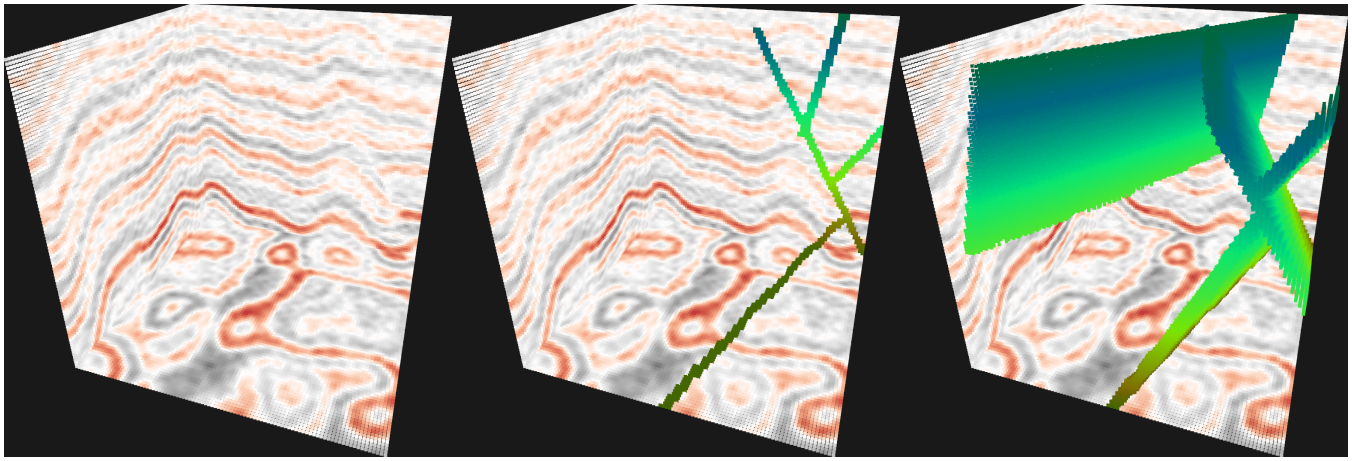

Fig. 8. An example of a volume from the synthetic seismic dataset, where the faults in the volume are visualised.

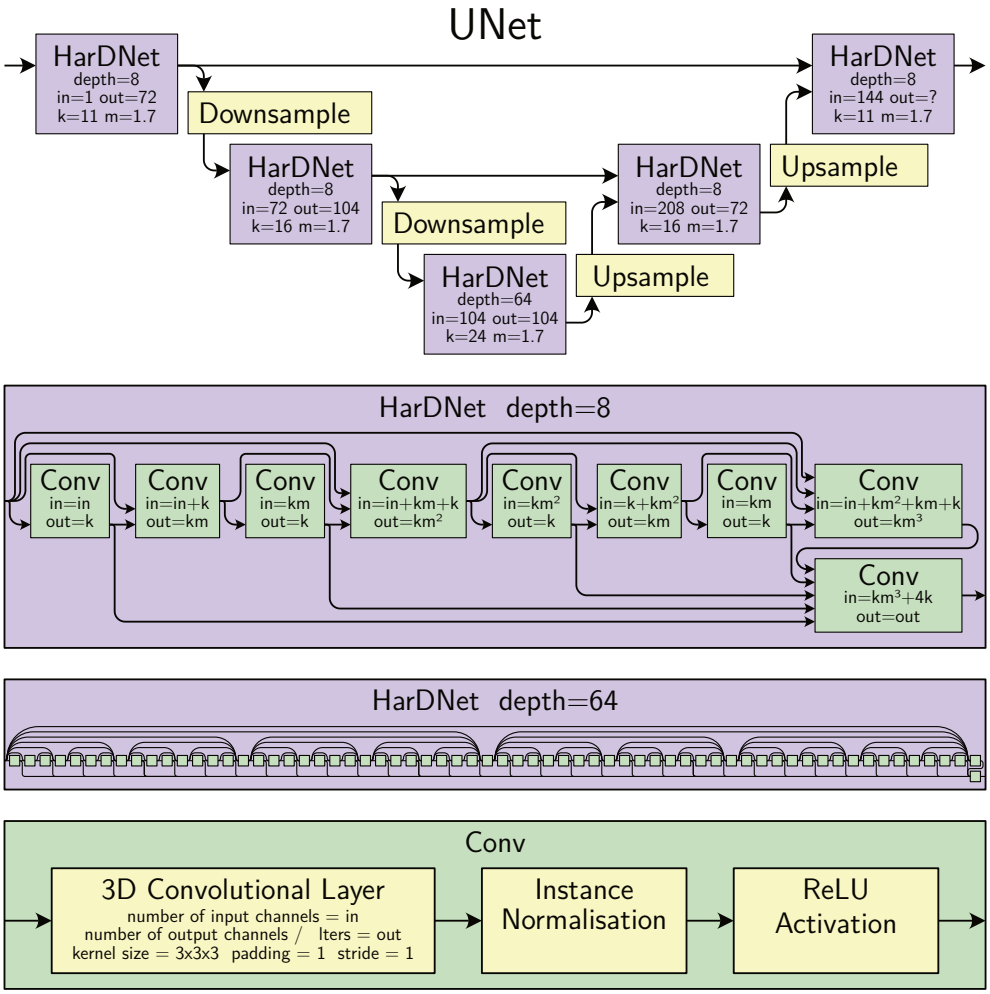


Fig. 9. Diagram of the network architecture used on the synthetic and real seismic data. Downsampling and upsamping reduce/increase the resolution by a factor of two respectively, using the max-pooling operation for downsampling and trilinear interpolation for upsampling. The model has 39 million trainable parameters.

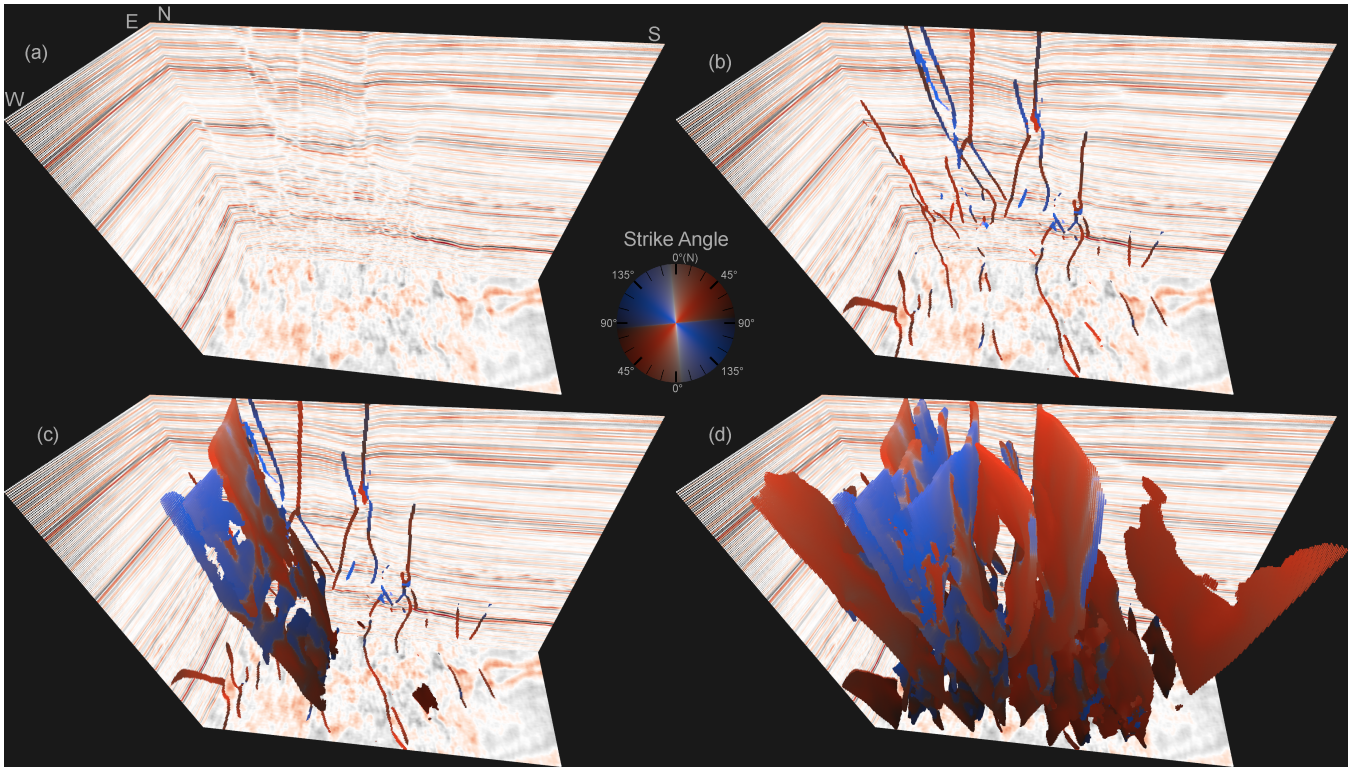


Fig. 10. 3D visualisation of shallow faults from the Laminaria 3D dataset bordering slice (b) in Fig. 18. The strike angle of the faults is visualised using colour, with additional shading making deeper voxels darker. In part (a) we visualise the seismic data only as slices on the boundary of the volume. Part (b) adds the predicted faults to the slices. Part (c) adds a single fault segment in 3D, which was separated by the post-processing algorithm in Section IV-C. Part (d) visualises all of the predicted faults in 3D.

occur. Finally, we count which unique integer values are present in the volume and map them to consecutive values starting at 2. The background is set to 0 and segments that are too small are assigned a value of 1.

7) The segments are now uniquely labelled and returned. However, they are smaller than the original predicted segments.
8) The segments are iteratively dilated until they cover the volume of the original segment probability volume. To encourage the dilating segments to grow in a meaningful manner, an iteration of connected smoothing is applied between every iteration of dilation until no more changes occur, and return the output.
9) Finally, we repeat the dilation process, but cover the whole faults including the intersections. This is used as the third and final output.

This algorithm returns its predicted fault segments in 3 levels. The first being the core points that were used to separate the fault segments, while the second covers the whole fault segments (voxels with exactly 1 fault present) as predicted by the neural network. The third covers all the faults (at least 1 fault present). One of the major advantages of this algorithm is that all its operations are performed in a local neighbourhood. They can be expressed as stencils and are therefore easily parallelisable on a GPU.

Due to GPU VRAM memory constraints, it is not possible to work with the whole Laminaria 3D volume at once, which has a resolution of $1444 \times 3964 \times 751$. Instead, we turn to a sliding window approach for both the model prediction and post processing. With the model having a fully convolutional architecture, it can be applied to any resolution. With 24GB of VRAM we were able to deploy the model on volumes of shape $176 \times 176 \times 176$, and moved the sliding windows with a stride of 44. We perform a weighted average of all the overlapping sliding windows, where the weight is highest in the centre of a window and 0 at the edge. Points near the edge cannot see the full context past the edge of the window to make the prediction, which is why they have a lesser weight. The weight changes sinusoidally from zero at the edge to 1 in the centre. To average the orientations we cannot simply average their vectors, since we don't know whether to multiply them by $-1$. Instead, we make use of our Piecewise-Aligned representation, where taking a mean of multiple orientations in this representation yields a meaningful average orientation. To do this we translate the orientation output of the model from any trained representation into the Piecewise-Aligned representation, average the values and invert the representation back into a vector.

When performing the post-processing, we use a sliding window of size $608 \times 608 \times 751$. Note that 751 is the full height of the Laminaria 3D volume. We slide the window with a stride of 304, where we compare the fault instances in overlapping windows. If two fault instances share more than 5 voxels in the core points volume (1st level output), we unify them as the same instance.

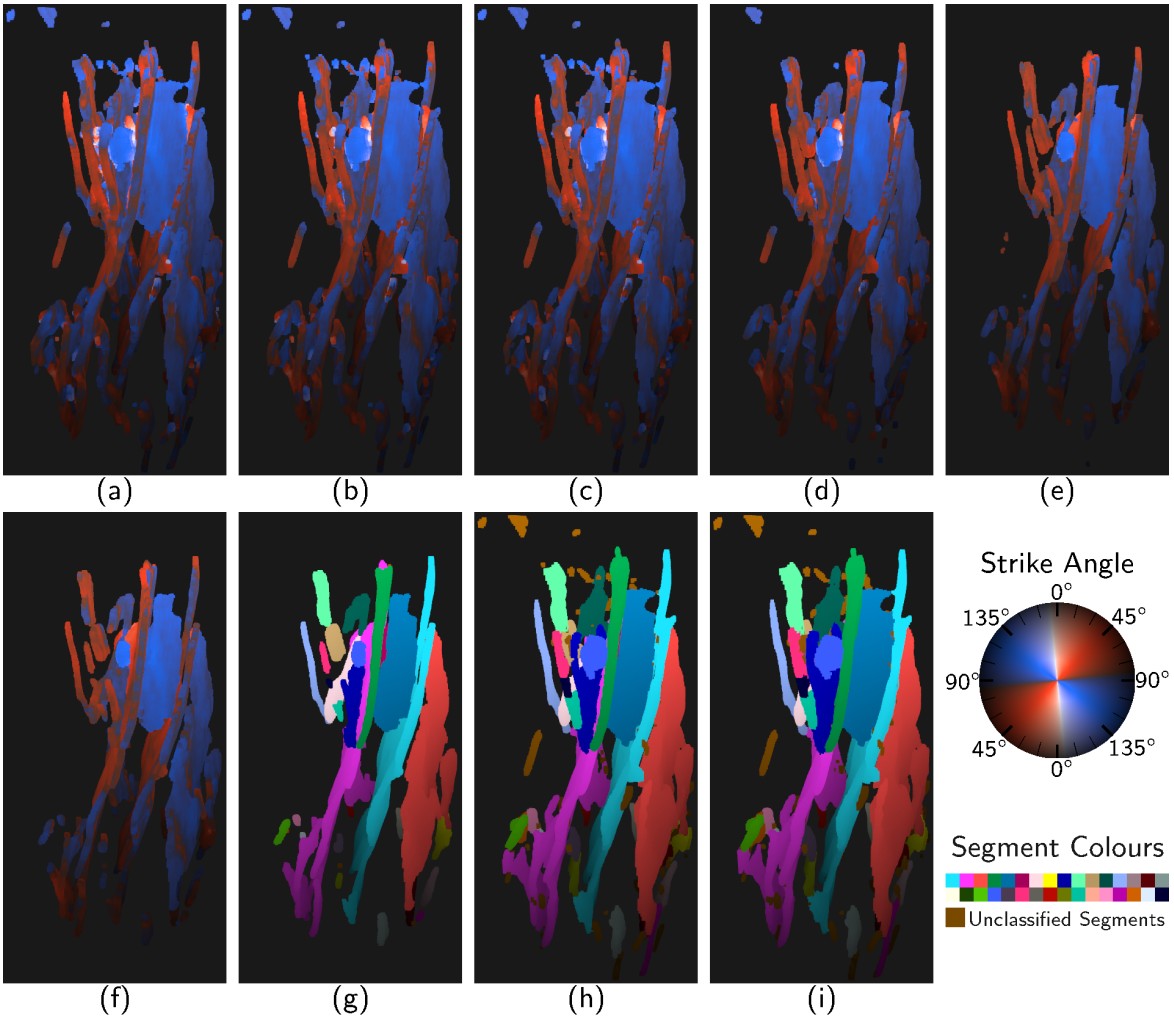


Fig. 11. 3D visualisation of shallow faults from the Laminaria 3D dataset at different stages of the post-processing algorithm. (a) - (f) visualise the strike angle using colour, while (g) - (i) visualise distinct fault segments in different colours. Additional shading is applied making deeper voxels darker. (a) visualises voxels with a high fault probability. (b) - (f) visualise the fault segments after steps 1 - 5 of the post processing algorithm respectively. (g), (h), and (i) visualise the result of steps 7, 8, and 9 respectively.

## V. Results

We can see the results of the toy task in Fig. 12, where the error is shown for all possible orientations. Note the clear areas of inconsistency in the locations of the theoretically identified discontinuities on the align-z representation and both dip and strike representations. All four proposed representations have a very low error and the error is similar for all possible orientations. This means that the proposed representations correctly address the theoretical issues that we identified with the existing representations. To evaluate the overall performance of the representations we plot the distribution of the performance over multiple runs in Fig. 13 as well as the average performance changing during training in Fig. 14. Note how the classification-style representations train faster initially.

We further evaluate the representations on the synthetic seismic volumes and plot their performance in Fig. 15 and 16. Note that not all orientations are represented in the synthetic dataset, which is why the toy task is important. We could not evaluate Saxena's representation on this task, since it took over 350 times longer to train than any other representation.

We deploy the best model trained on the synthetic dataset with the Projection-Doubleangle representation on real seismic data from the Laminaria 3D volume. We visualise the prediction on three horizons through the volume in map view in Fig. 17, which we compare to the interpretation by Phillips et al. [1]. We compare the orientation of the faults on the horizons, as well as in areas around the Corallina, Laminaria and Vidalia wells in Fig. 19. To evaluate how well the model distinguishes the faults, we analyse two slices through the volume seen in Fig. 18, where a geologist annotates the predicted faults as true positives or false positives. They also annotate faults that the model missed as false negatives. We count the number of faults in each category and calculate the following metrics:

$$\text{F1 Score: } \frac{2 * TP}{2 * TP + FP + FN}, \tag{24}$$

$$\text{Jaccard Index (IOU): } \frac{TP}{TP + FP + FN}. \tag{25}$$

The results can be seen in Tables I and II. For comparison, on the synthetic test set, the model achieved an F1 of 98% and an IOU of 96%. Volumetrically, it got an IOU of 67%.

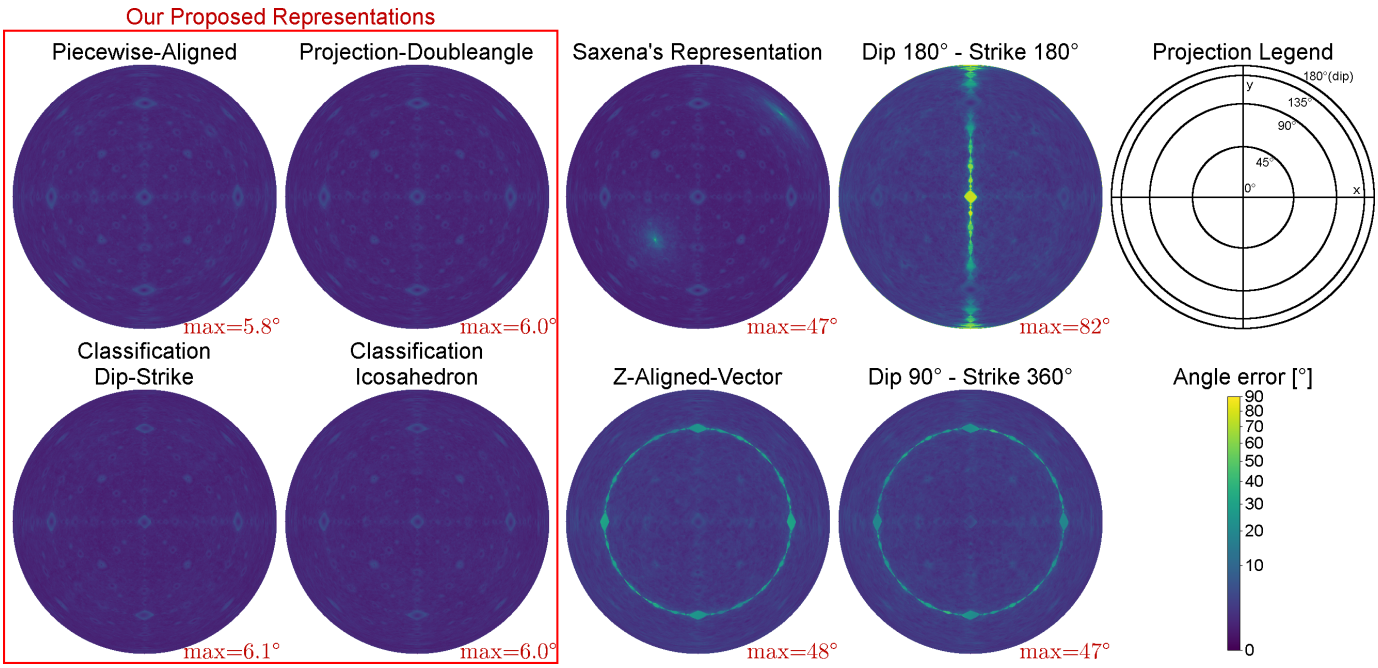


Fig. 12. The error of models trained on the toy dataset with a variety of representations, visualised for all possible orientations on a Lambert azimuthal equal-area projection, aka equal-area stereonet. Note the symmetry in the diagrams, where two points with any dip and $(180° - \text{dip})$ on the other side of the diagram represent the same orientation.

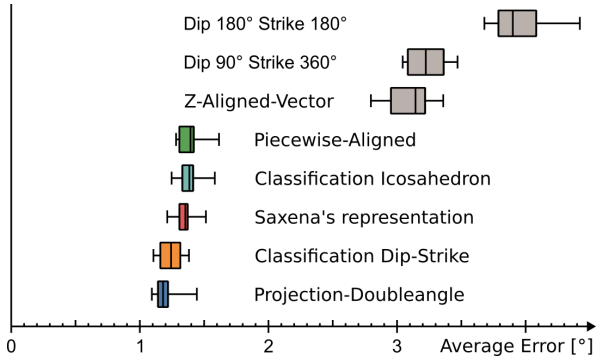


Fig. 13. Graph showing the distribution across 10 runs, of the average error in degrees on the toy task. (lower is better)

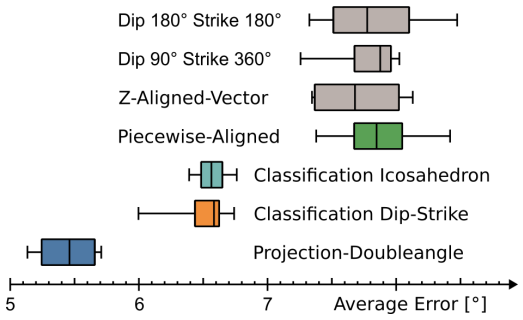


Fig. 15. Graph showing the distribution across 4 runs, of the average error in degrees on the synthetic seismic dataset. (lower is better)

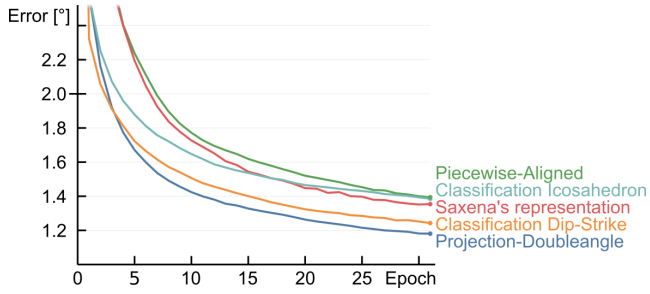


Fig. 14. Graph showing the median over 10 runs, of the average error in degrees during training of the toy task.

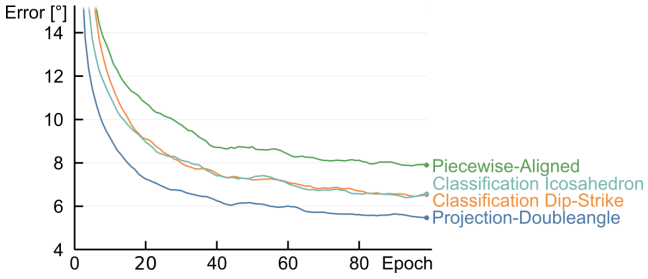


Fig. 16. Graph showing the median over 4 runs, of the average error in degrees during training on the synthetic seismic dataset.

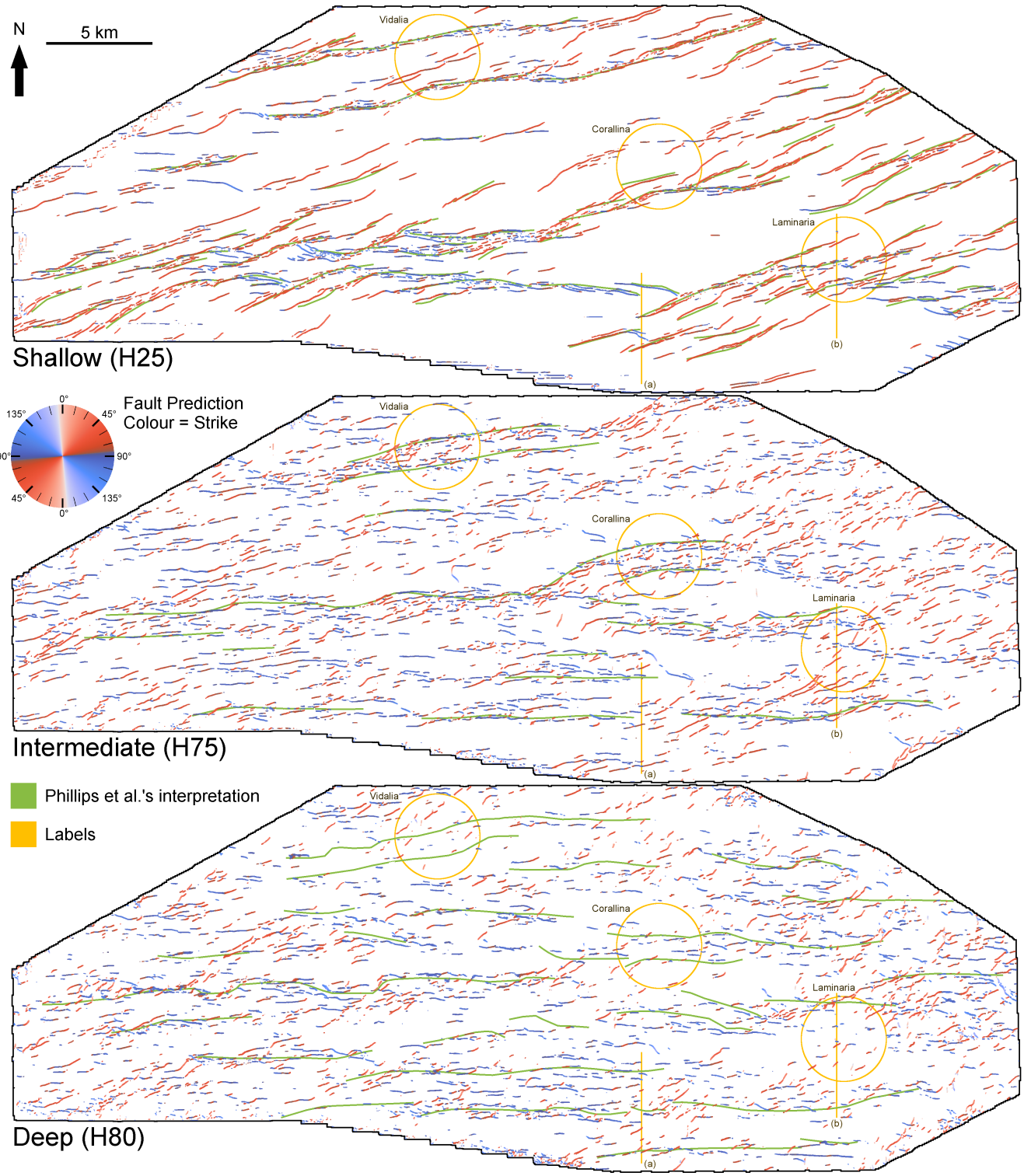


Fig. 17. 2D map comparison between modelled fault traces and those intepreted by Phillips et al. [1]. There is generally good correspondence between the model and interpreter at shallow (H25 level, c. 1s TWT). At intermediate (H75, c. 2.3 s TWT) the level of agreement drops. The structures identified by Phillips et al. correspond to the main east-west fault traces. The Corallina, Laminaria and Vidalia (yellow circles) are where the wells were drilled in the area of this seismic volume. The model shows a lot of additional detail in the form of short E-W and ENE-WSW trending fault segments that are probably secondary faults to the main structures. At deep (H80, c. 2.6 s TWT) levels there is very little correspondence between the modelled and interpreter structures other than a broad agreement in the trend and location of some of the bigger faults. Lines (a) and (b) correspond to the two slices through the data shown in Fig. 18.

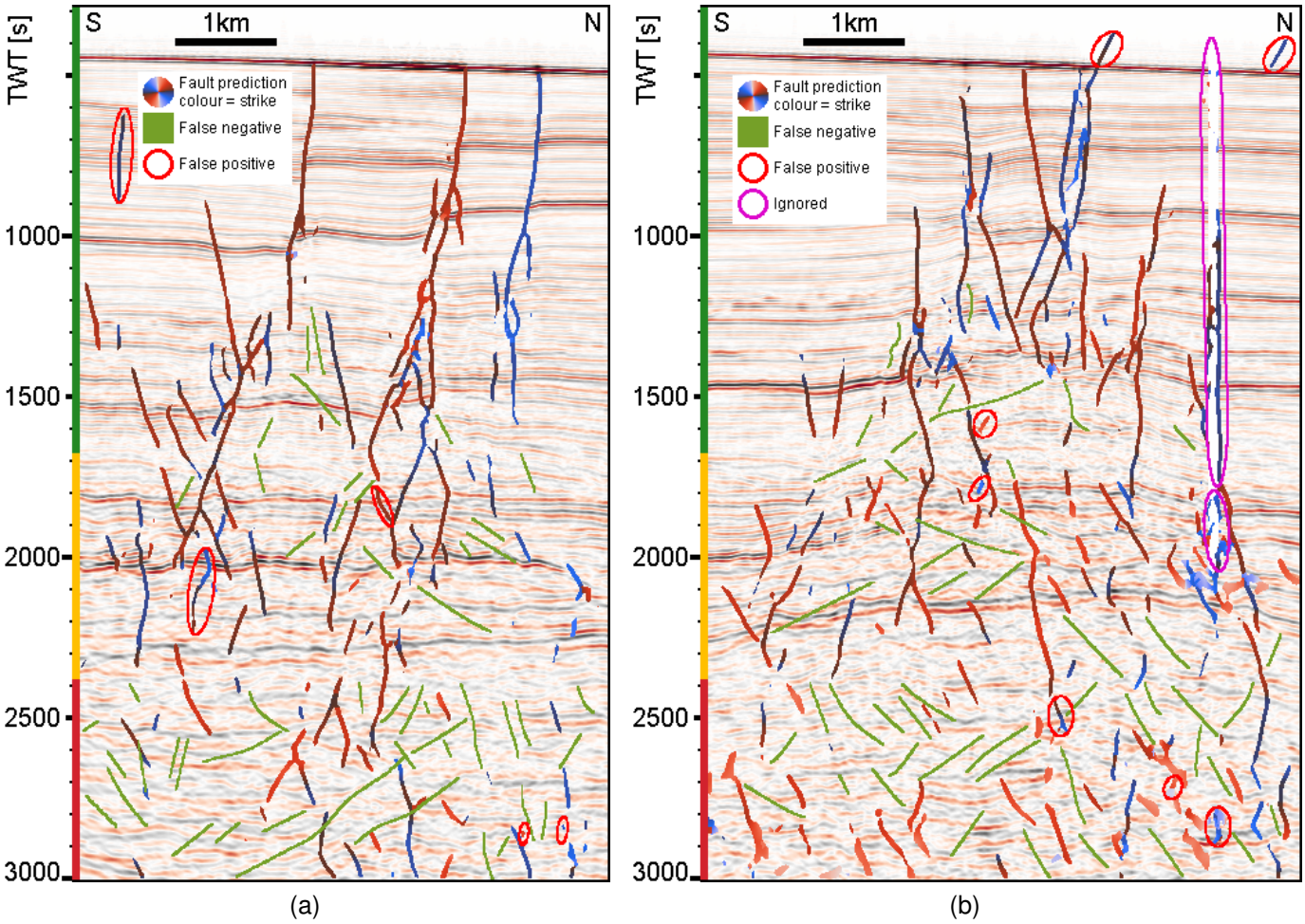


Fig. 18. Slice (a) and (b) taken across the Laminaria High (locations are shown on Fig. 17). The model fault picks are shown in red and blue. False positive structures (where the interpreter (McCaffrey) does not agree with a model pick) are shown and are overall quite low in number. False negatives (model has missed faults the interpeter would pick) increase dramatically with depth. The faults are grouped into shallow between 400-1680 on TWT scale (green), intermediate between 1680-2400 (yellow) and deep level for those greater than 2400 TWT (red) which corresponds to the levels identified by Phillips et al. [1]) and the metrics presented in Table I and II.

TABLE I
METRICS ON SLICE (A) IN FIG. 18.

| | Shallow | Intermediate | Deep | All |
|---|---|---|---|---|
| Total Faults | 39 | 52 | 63 | 154 |
| True Positives | 32 | 38 | 30 | 100 |
| False Positives | 2 | 2 | 2 | 6 |
| False Negatives | 5 | 12 | 31 | 48 |
| F1 Score | 90.1% | 84.4% | 64.5% | 78.7% |
| Jaccard Index | 82.1% | 73.1% | 47.6% | 64.9% |

TABLE II
METRICS ON SLICE (B) IN FIG. 18.

| | Shallow | Intermediate | Deep | All |
|---|---|---|---|---|
| Total Faults | 55 | 64 | 68 | 187 |
| True Positives | 42 | 50 | 38 | 130 |
| False Positives | 5 | 1 | 3 | 9 |
| False Negatives | 8 | 13 | 27 | 48 |
| F1 Score | 86.6% | 87.7% | 71.7% | 82.0% |
| Jaccard Index | 76.4% | 78.1% | 55.9% | 69.5% |

## VI. DISCUSSION

We observe that all the baseline representations are fundamentally flawed. The lack of continuity in the Z-Aligned-Vector, Dip 90° Strike 360°, and Dip 180° Strike 180° is problematic because a small change in orientation causes a big change in the representation's values. This generates large errors in the predicted orientation near areas of discontinuity on the toy dataset, as seen in Fig. 12. It also causes a larger overall error on the toy dataset performance (Fig. 13, 14). On the synthetic data these representations also perform poorly in comparison to our proposed representations (Fig. 15, 16).

The only existing representation in literature that is continuous is Saxena's representation. However, it is difficult and computationally expensive to invert this representation, making it infeasible for use on the seismic data.

Out of the four novel representations that we propose, we single out the Projection-Doubleangle representation as the most appropriate for predicting the orientation of planes in 3D. It performs the best on both the toy task (Fig. 13, 14) and the synthetic seismic data for predicting the orientation

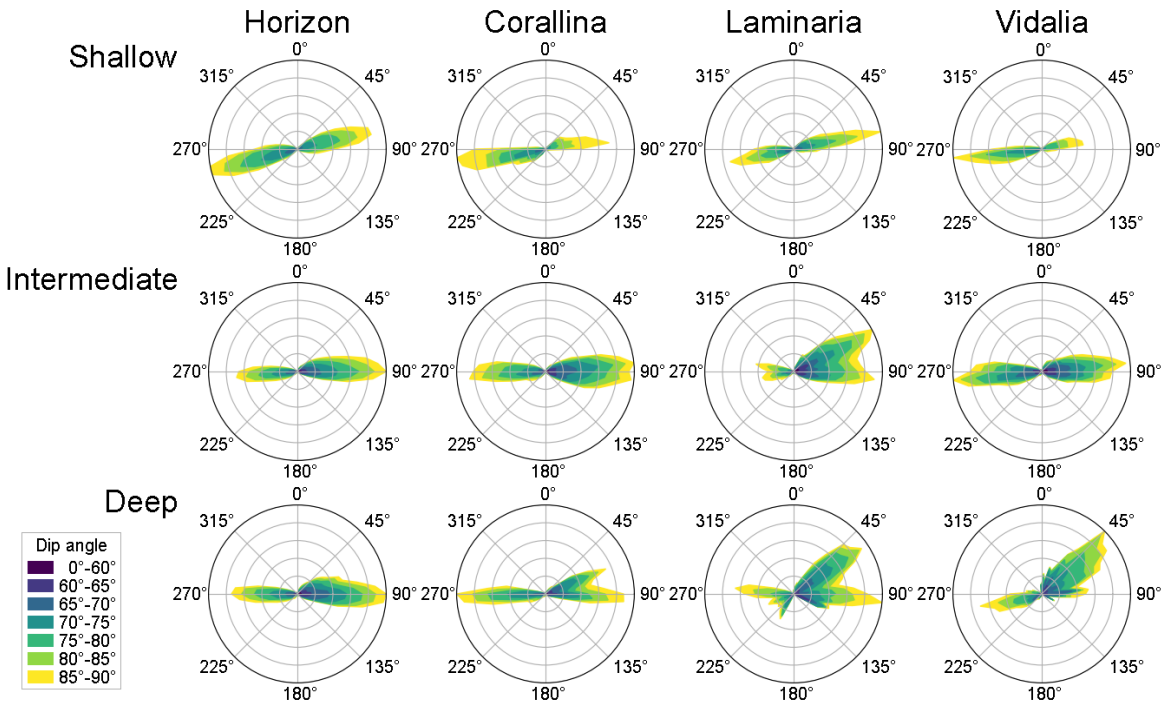


Fig. 19. Rose diagrams showing the distribution of the strike and dip angles in the following areas: Horizon spans the whole area of the volume at shallow horizon H25 level, intermediate horizon H75 and deep horizon H80 respectively. Corallina, Laminaria and Vidalia span an area with a radius of 2km and ±100ms from the respective horizon in TWT, as seen in Fig. 17 in yellow.

of faults (Fig. 15, 16). Additionally, it requires a mere 6 values to represent the orientation, which is the least out of all the proposed representations. However, the other proposed representations do provide some advantages.

Our Piecewise-Aligned representation has the advantage of easily scaling to n-dimensional vectors instead of being limited to 3D vectors. We can also make use of the representation to find an average orientation by averaging the values of multiple orientations represented using the Piecewise-Aligned representation. We make use of this property in our post-processing algorithm that separates faults into non-overlapping instances.

The Classification Dip-Strike and Classification Icosahedron representations have the advantage of being trained using a cross-entropy loss on top of a softmax activation function. This allows them to train more quickly at the beginning, as seen in Fig. 14, since a rough prediction brings the output into the neighbourhood of the closest category. This contrasts with the Piecewise-Aligned and Projection-Doubleangle representations that require finer adjustments before the prediction becomes useful. The Classification Icosahedron representation is advantageous because it only requires 6 values to encode an orientation, albeit at the expense of more complicated equations. The Classification Dip-Strike representation requires 10 values.

Using the Projection-Doubleangle representation we apply the model to the Laminari 3D seismic volume, where we demonstrate the utility of predicting the orientation of the faults by using it as one of the key elements for separating faults into non-intersecting fault instances or fault segments. Our post-processing fault-separation algorithm only uses local operations, making it easily parallelisable on the GPU. It makes the visualisation of the predicted faults in 3D much simpler, by selectively hiding some fault segments that may be obscuring other faults of interest. We can also sort the fault segments by their size, only displaying larger faults while applying smoothing to the predicted fault orientations over neighbouring voxels of the same fault segment.

We further analyse the prediction on the Laminaria 3D seismic volume where we observe that the model performs well at shallow levels. There is good agreement with a high F1 score and Jaccard Index at the shallow levels in Table I and II from Fig. 18. In 2D map view, the structures match well in terms of location (Fig. 17) and orientation (Fig. 19) with previous manual interpretations by Phillips et al. [1] and Çiftçi and Langhi [26]. The trends in Fig. 19 for the Corallina, Laminaria and Vidalia areas at the shallow level are the same ENE-WSW trends as shown in the previous work. At the deep level both E-W trending and a large number of ENE-WSW fault major structures were identified by the model, however only E-W structures were identified in previous work. This is possibly because the ENE-WSW structures are short and were considered to be secondary faults by the authors. Note the prominent structural feature where the two trends meet, as seen in Fig. 17 in horizon Shallow H25, which was labelled as 'intersection point' and 'outboard en-echelon faults' by Phillips et al. [1].

There is some drop off in performance between the model and interpreter at intermediate level (Fig. 18, Table I, II), but overall the agreement remains relatively high. At deep levels, the amount of correspondence between the interpreter and the model is relatively low (Fig. 18, Table I, II). The main

reason for the poorer model performance is the number of false negatives, i.e. structures that the model has not recognised that the interpreter thinks should have been there. Possible explanations for this poorer performance at deep levels might have to do with the high levels of ambient noise at depth in the Laminaria 3D seismic volume and the presence of more low dip angle structures, both of which are missing from the synthetic training data set. We can see from our modelled dip values that there are more gently dipping faults at the deep and intermediate levels compared to the shallow level. In contrast, at the deep and intermediate levels many more faults are picked by the model than shown in the published interpretations. We do not, however, consider these to be false positives because they were not identified as such in our own interpretation (Fig. 18). It is likely that in previous studies [1] the fault interpretation has been implicitly simplified at intermediate and deep levels to focus on primary structures, i.e., those that bound the main high structures. Our model has picked out both the major and minor (secondary) faults. This illustrates the difficulty in comparing Deep Learning representations of faults with published interpretations.

## VII. Conclusion

Up until now, there have been no representations for negation invariant vectors in 3D that can be predicted by a neural network, are continuous, and are easily parallelisable on a GPU. We propose four novel representations with all of these properties and use them to predict the orientation of planes in 3D, both on a toy task and applied to predicting the orientation of faults in seismic tomographic volumes. Our Projection-Doubleangle representation outperforms all other existing representations on this task. We train our model on synthetic data and deploy it on the Laminaria 3D volume, where we predict the location and orientation of faults. We further separate faults into individual non-intersecting segments using their orientation in a post-processing algorithm on the GPU. We observe that the model performs well in shallow regions. In deeper levels with more ambient noise, the model misses more faults. In intermediate levels the model detects many more correct faults that were missed or ignored by previous studies. In contrast to previous work, our model is capable of predicting the orientation of the faults and use this information to separate the predicted faults into fault segments.

## References


[1] T. B. Phillips, K. McCaffrey, and L. Magarinos, "Influence of variable decoupling between vertically separated fault populations on structural inheritance–the laminaria high, nw shelf of australia," *Basin Research*, vol. 34, no. 1, pp. 440–456, 2022.

[2] J. O'Connell, Z. Li, J. Hanson, R. Heffernan, J. Lyons, K. Paliwal, A. Dehzangi, Y. Yang, and Y. Zhou, "Spin2: Predicting sequence profiles from protein structures using deep neural networks," *Proteins: Structure, Function, and Bioinformatics*, vol. 86, no. 6, pp. 629–633, 2018.

[3] J. Lyons, A. Dehzangi, R. Heffernan, A. Sharma, K. Paliwal, A. Sattar, Y. Zhou, and Y. Yang, "Predicting backbone c$\alpha$ angles and dihedrals from protein sequences by stacked sparse auto-encoder deep neural network," *Journal of computational chemistry*, vol. 35, no. 28, pp. 2040–2046, 2014.

[4] H. Li, J. Hou, B. Adhikari, Q. Lyu, and J. Cheng, "Deep learning methods for protein torsion angle prediction," *BMC bioinformatics*, vol. 18, no. 1, pp. 1–13, 2017.

[5] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 1316–1322.

[6] D. Kluvanec, T. B. Phillips, K. J. McCaffrey, and N. Al Moubayed, "Using orientation to distinguish overlapping chromosomes," in *International Conference on Artificial Neural Networks*. Springer, 2022, pp. 391–403.

[7] G. Pitteri, M. Ramamonjisoa, S. Ilic, and V. Lepetit, "On object symmetries and 6d pose estimation from images," in *2019 International conference on 3D vision (3DV)*. IEEE, 2019, pp. 614–622.

[8] T. Hodan, D. Barath, and J. Matas, "Epos: Estimating 6d pose of objects with symmetries," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 703–11 712.

[9] A. Saxena, J. Driemeyer, and A. Y. Ng, "Learning 3-d object orientation from images," in *2009 IEEE International conference on robotics and automation*. IEEE, 2009, pp. 794–800.

[10] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM review*, vol. 53, no. 2, pp. 217–288, 2011.

[11] X. Wu, L. Liang, Y. Shi, and S. Fomel, "Faultseg3d: Using synthetic data sets to train an end-to-end convolutional neural network for 3d seismic fault segmentation," *GEOPHYSICS*, 2019.

[12] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015.

[13] X. Wu and D. Hale, "3d seismic image processing for faults," *Geophysics*, vol. 81, 2016.

[14] X. Wu, Z. Geng, Y. Shi, N. Pham, S. Fomel, and G. Caumon, "Building realistic structure models to train convolutional neural networks for seismic structural interpretation," *Geophysics*, vol. 85, 2020.

[15] K. Gao, L. Huang, and Y. Zheng, "Fault detection on seismic structural images using a nested residual u-net," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–15, 2022.

[16] Y. Dou, K. Li, J. Zhu, T. Li, S. Tan, and Z. Huang, "Efficient training of 3d seismic image fault segmentation network under sparse labels by weakening anomaly annotation," *ArXiv*, vol. abs/2110.05319, 2021.

[17] R. Feng, D. Grana, and N. Balling, "Uncertainty quantification in fault detection using convolutional neural networks," *Geophysics*, vol. 86, 2021.

[18] G. Hu, Z. Hu, J. Liu, F. Cheng, and D. Peng, "Seismic fault interpretation using deep learning-based semantic segmentation method," *IEEE Geoscience and Remote Sensing Letters*, 2020.

[19] Y. An, J. Guo, Q. Ye, C. Childs, J. Walsh, and R. Dong, "A gigabyte interpreted seismic dataset for automatic fault recognition," *Data in Brief*, vol. 37, 2021.

[20] ——, "Deep convolutional neural network for automatic fault recognition from 3d seismic datasets," *Computers & Geosciences*, vol. 153, p. 104776, 2021.

[21] T. Wrona, I. Pan, R. E. Bell, R. L. Gawthorpe, H. Fossen, and S. Brune, "3d seismic interpretation with deep learning: A brief introduction," *The Leading Edge*, vol. 40, no. 7, pp. 524–532, 2021.

[22] X. Wu, Y. Shi, S. Fomel, L. Liang, Q. Zhang, and A. Yusifov, "Faultnet3d: Predicting fault probabilities, strikes, and dips with a single convolutional neural network," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, pp. 9138–9155, 2019.

[23] X. Wu, L. Liang, Y. Shi, Z. Geng, and S. Fomel, "Multitask learning for local seismic image processing: fault detection, structure-oriented smoothing with edge-preserving, and seismic normal estimation by using a single convolutional neural network," *Geophysical Journal International*, 2019.

[24] Z. Bi and X. Wu, "Improving fault surface construction with inversion-based methods," *GEOPHYSICS*, 2021.

[25] D. Hale, "Methods to compute fault images, extract fault surfaces, and estimate fault throws from 3d seismic images," *Geophysics*, vol. 78, 2013.

[26] N. B. Çiftçi and L. Langhi, "Evolution of the hourglass structures in the laminaria high, timor sea: Implications for hydrocarbon traps," *Journal of Structural Geology*, vol. 36, pp. 55–70, 2012.

[27] Australian Geological Survey Organisation (AGSO) North West Shelf Study Group, "Deep reflections on the north west shelf: changing perspectives of basin formation," pp. 63–74, 1994.

[28] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *2016 fourth international conference on 3D vision (3DV)*. Ieee, 2016, pp. 565–571.

[29] P. Chao, C.-Y. Kao, Y.-S. Ruan, C.-H. Huang, and Y.-L. Lin, "Hardnet: A low memory traffic network," 2019.