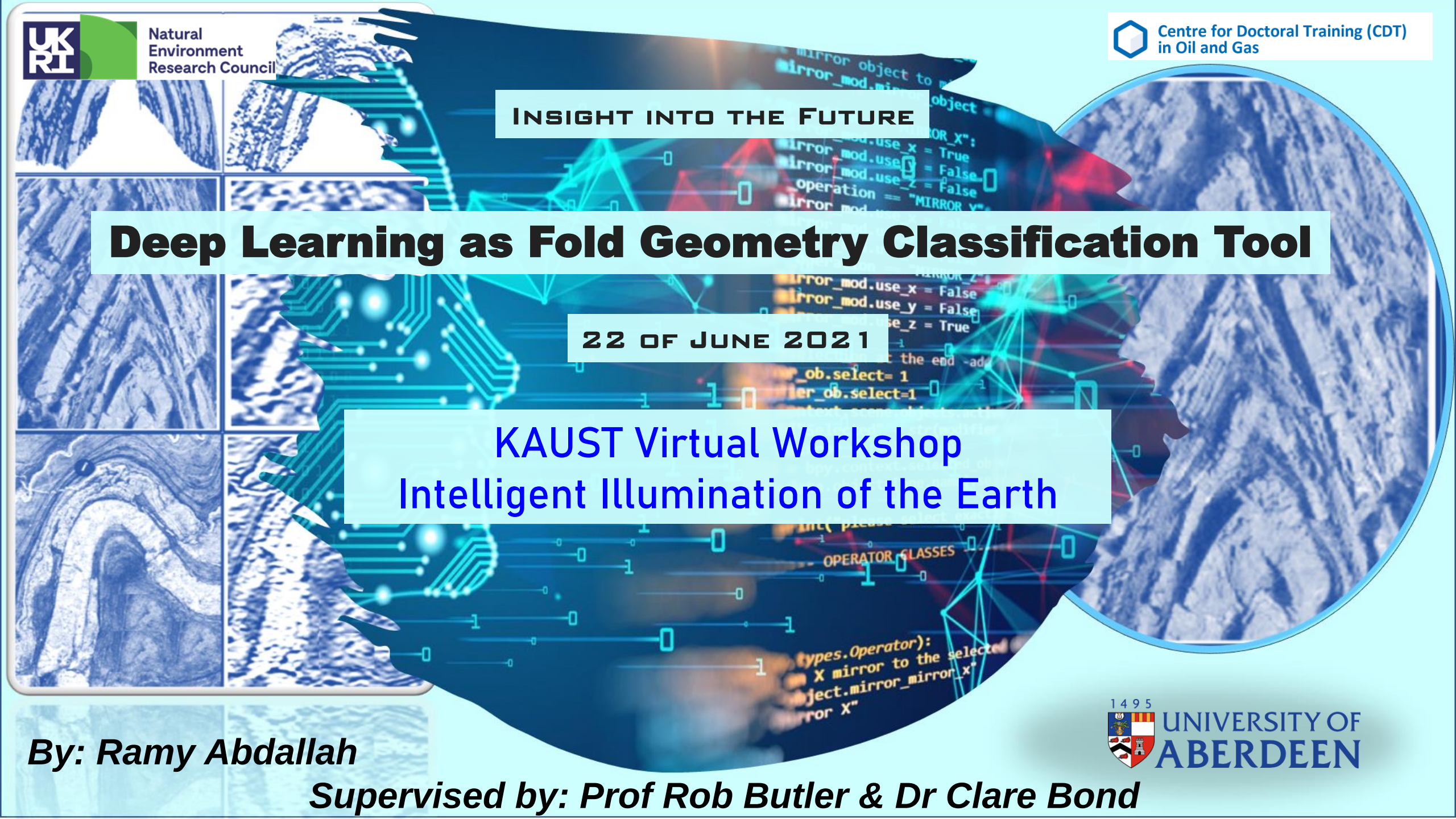


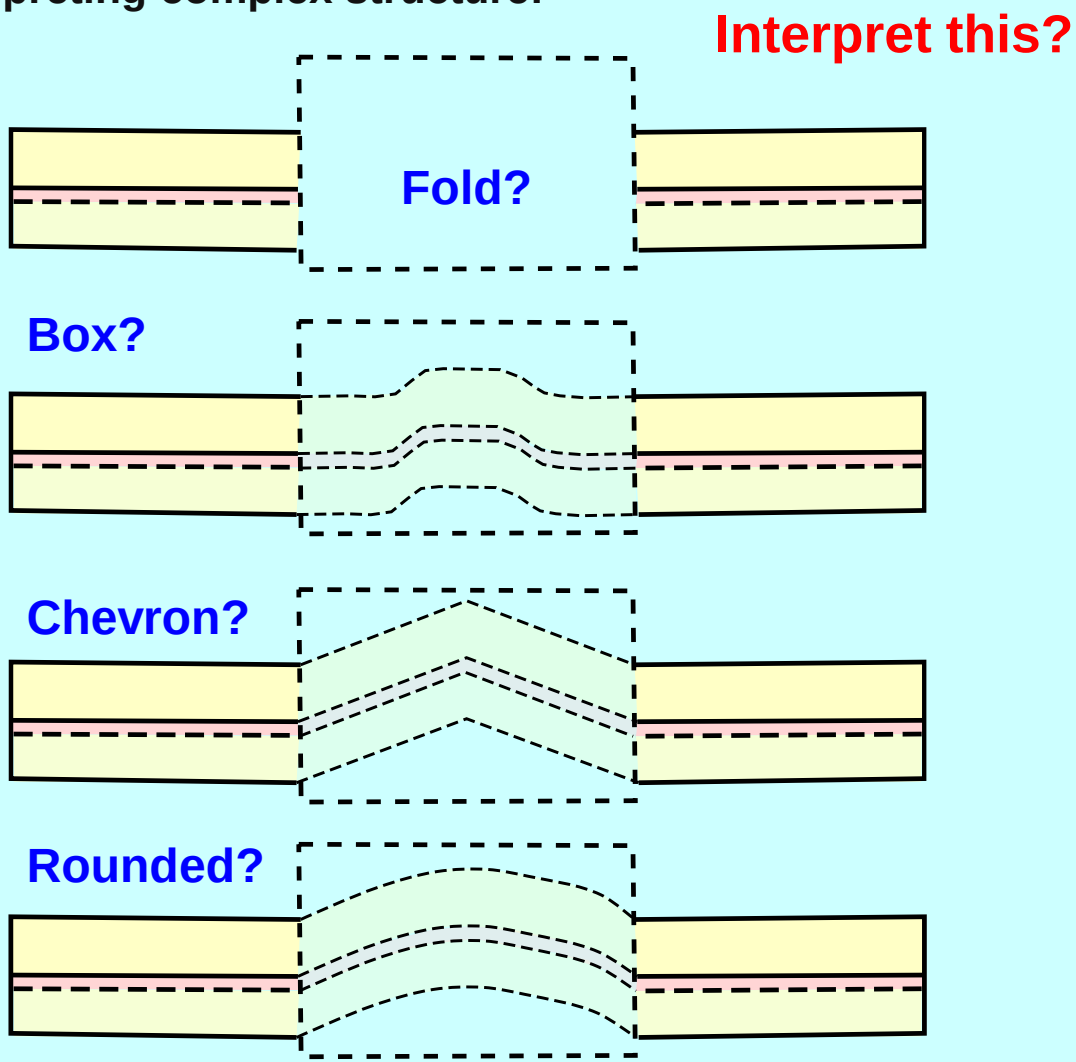

By: Ramy Abdallah

Supervised by: Prof Rob Butler & Dr Clare Bond

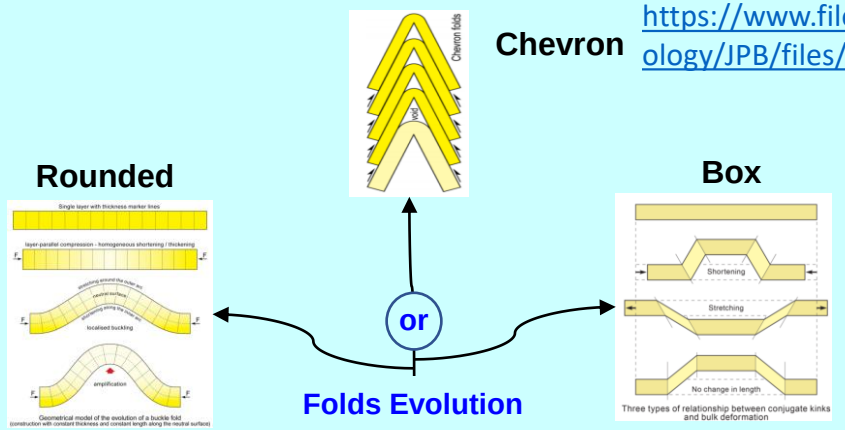
# Why perform fold classification?



**1. Provide additional information and constraints when interpreting complex structure:**

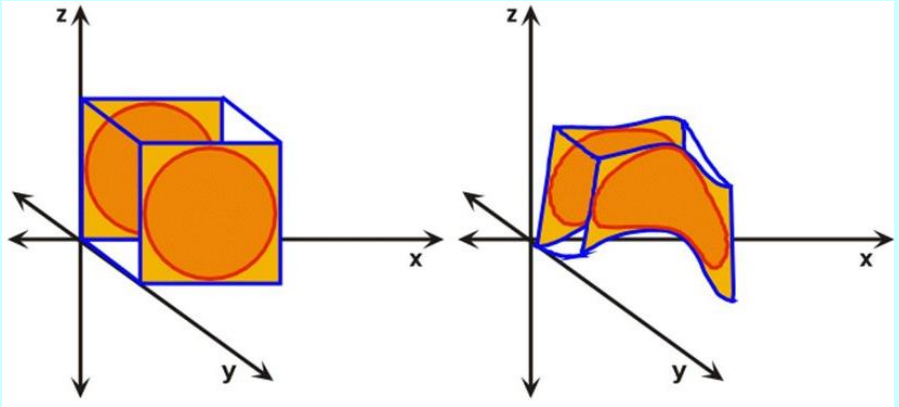



**2. Helps to understand folding mechanisms and development:**




**3. Helps to understand distribution and heterogeneity of strain:**

Heterogeneous or Inhomogeneous Strain

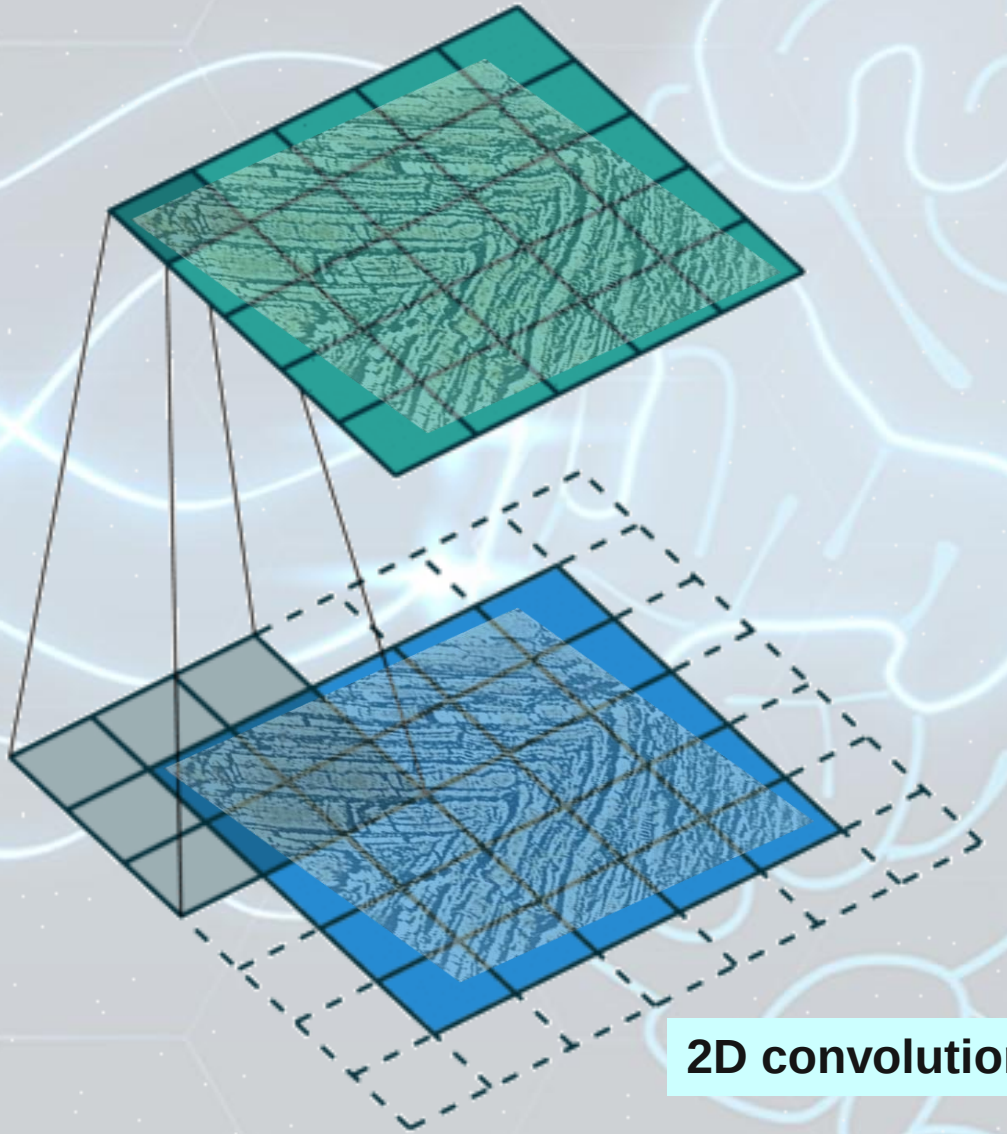


# Image Classification

## Using 2D Convolutional Neural Networks



https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

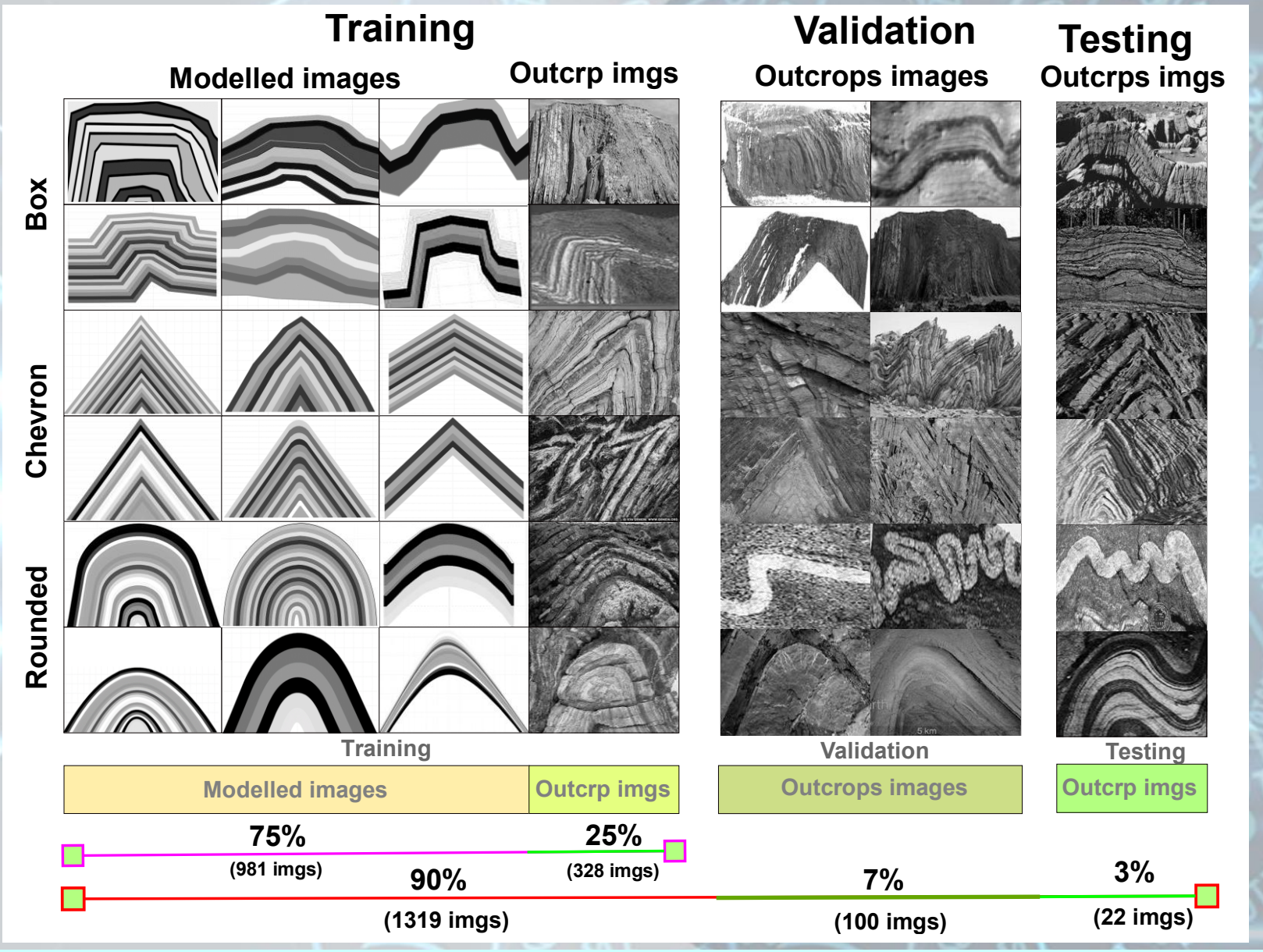


2D convolution operation

- Start with a kernel, which is simply a small matrix of weights.
- This kernel “slides” over the 2D input data, performing an elementwise multiplication with the part of the input it is currently on.
- Then summing up the results into a single output pixel.

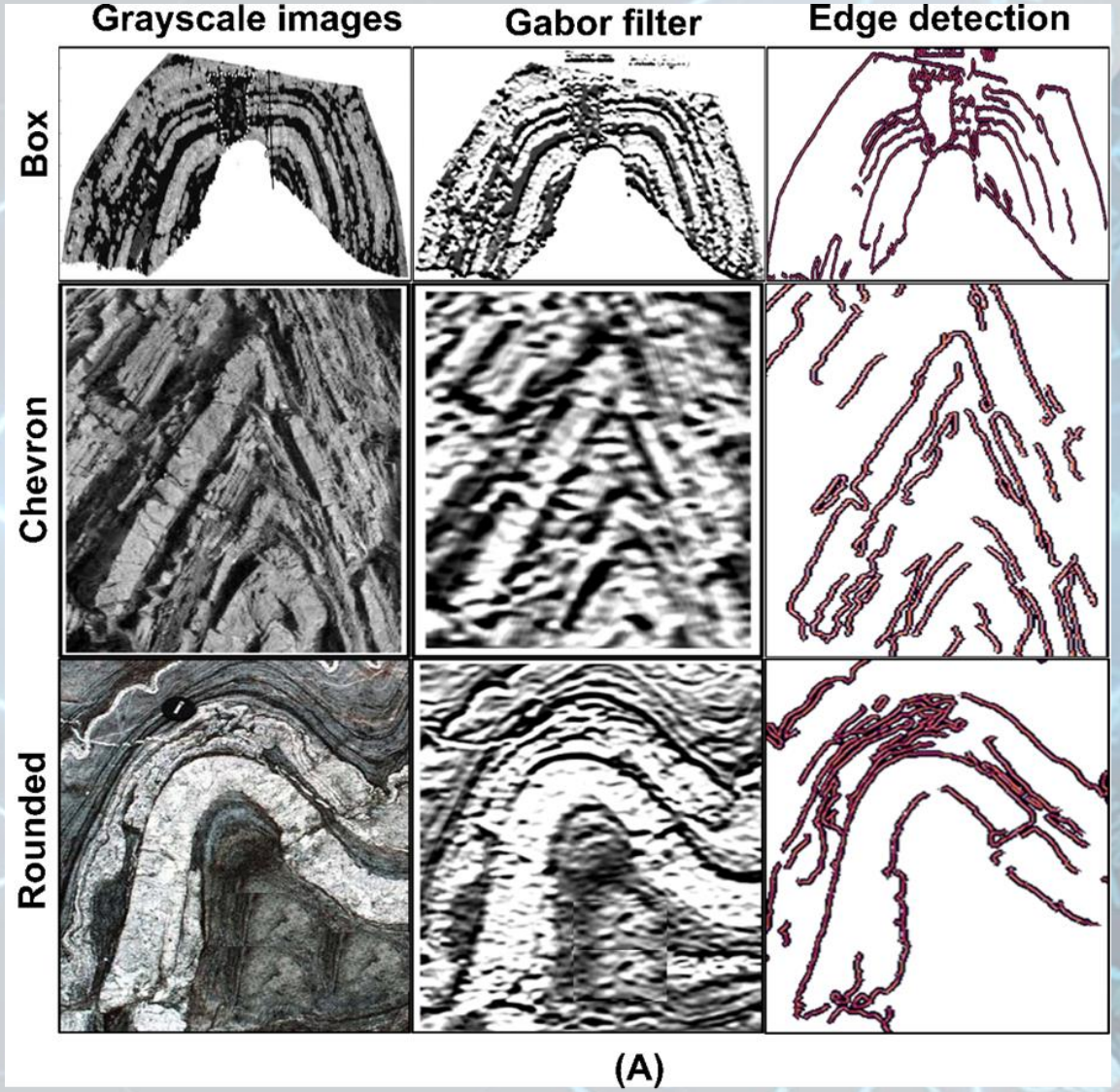
# Shape of Hinges Dataset

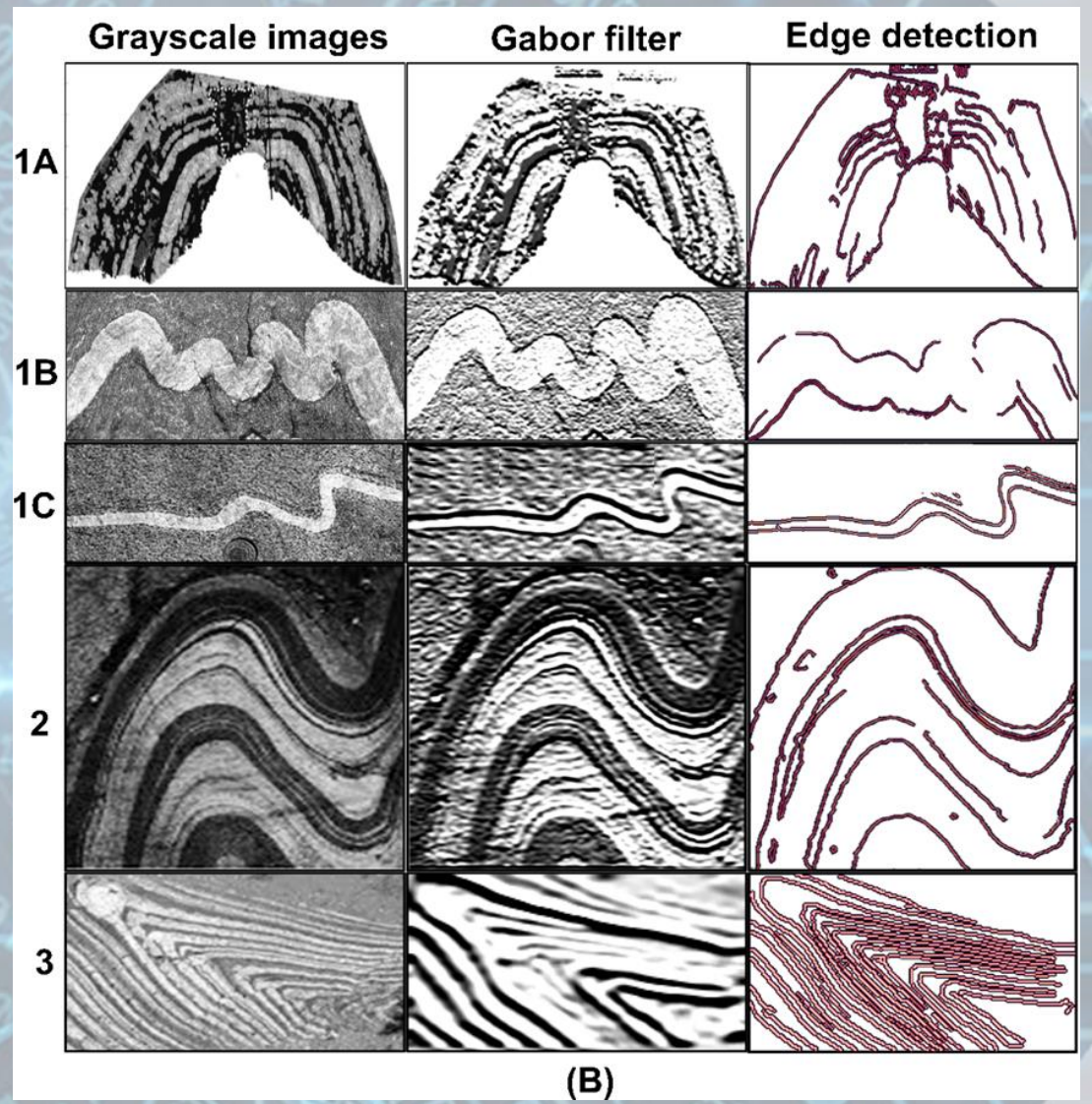




# Shallow Learning - Feature Extraction






Example of features extracted: Coloured, grayscale, original, Gabor, edge, noise, mean, std.

# Shallow Learning - Results



| Model | Accuracy | Balanced Accuracy | ROC AUC | F1 Score | Time Taken |
|---|---|---|---|---|---|
| RandomForestClassifier | 0.64 | 0.64 | None | 0.64 | 3.30 |
| ExtraTreesClassifier | 0.64 | 0.64 | None | 0.64 | 1.16 |
| LabelSpreading | 0.60 | 0.60 | None | 0.60 | 2.19 |
| LabelPropagation | 0.60 | 0.60 | None | 0.60 | 1.82 |
| LGBMClassifier | 0.60 | 0.60 | None | 0.60 | 1.87 |
| BaggingClassifier | 0.59 | 0.59 | None | 0.59 | 2.15 |
| XGBClassifier | 0.59 | 0.59 | None | 0.59 | 4.14 |
| ExtraTreeClassifier | 0.56 | 0.56 | None | 0.56 | 0.07 |
| KNeighborsClassifier | 0.50 | 0.50 | None | 0.50 | 0.27 |
| SGDClassifier | 0.49 | 0.49 | None | 0.46 | 0.29 |
| DecisionTreeClassifier | 0.49 | 0.49 | None | 0.49 | 0.41 |
| NuSVC | 0.48 | 0.48 | None | 0.48 | 2.78 |
| CalibratedClassifierCV | 0.48 | 0.48 | None | 0.48 | 10.73 |
| SVC | 0.47 | 0.47 | None | 0.47 | 1.62 |
| Perceptron | 0.47 | 0.47 | None | 0.46 | 0.08 |
| RidgeClassifier | 0.47 | 0.47 | None | 0.47 | 0.09 |
| LinearDiscriminantAnalysis | 0.46 | 0.46 | None | 0.46 | 0.16 |
| RidgeClassifierCV | 0.46 | 0.46 | None | 0.46 | 0.12 |
| LinearSVC | 0.46 | 0.46 | None | 0.46 | 2.94 |
| LogisticRegression | 0.45 | 0.45 | None | 0.45 | 0.41 |
| AdaBoostClassifier | 0.40 | 0.40 | None | 0.40 | 2.63 |
| QuadraticDiscriminantAnalysis | 0.39 | 0.39 | None | 0.39 | 0.14 |
| PassiveAggressiveClassifier | 0.38 | 0.38 | None | 0.32 | 0.14 |
| GaussianNB | 0.34 | 0.34 | None | 0.28 | 0.05 |
| NearestCentroid | 0.33 | 0.33 | None | 0.20 | 0.05 |
| BernoulliNB | 0.31 | 0.31 | None | 0.17 | 0.06 |
| DummyClassifier | 0.30 | 0.30 | None | 0.30 | 0.05 |

- **Validation dataset**

- **Maximum Accuracy of 64%**

| Model | Accuracy | Balanced Accuracy | ROC AUC | F1 Score | Time Taken |
|---|---|---|---|---|---|
| XGBClassifier | 0.63 | 0.63 | None | 0.63 | 4.02 |
| LGBMClassifier | 0.59 | 0.58 | None | 0.56 | 2.08 |
| ExtraTreesClassifier | 0.57 | 0.57 | None | 0.56 | 1.00 |
| RandomForestClassifier | 0.57 | 0.57 | None | 0.55 | 3.31 |
| DecisionTreeClassifier | 0.53 | 0.53 | None | 0.52 | 0.40 |
| BaggingClassifier | 0.53 | 0.53 | None | 0.50 | 2.28 |
| CalibratedClassifierCV | 0.53 | 0.52 | None | 0.52 | 11.40 |
| LinearSVC | 0.53 | 0.52 | None | 0.50 | 2.88 |
| LogisticRegression | 0.51 | 0.51 | None | 0.50 | 0.40 |
| SGDClassifier | 0.49 | 0.49 | None | 0.45 | 0.32 |
| NuSVC | 0.49 | 0.48 | None | 0.45 | 2.69 |
| LinearDiscriminantAnalysis | 0.49 | 0.48 | None | 0.46 | 0.12 |
| RidgeClassifierCV | 0.49 | 0.48 | None | 0.46 | 0.12 |
| RidgeClassifier | 0.49 | 0.47 | None | 0.45 | 0.08 |
| KNeighborsClassifier | 0.47 | 0.47 | None | 0.47 | 0.13 |
| QuadraticDiscriminantAnalysis | 0.47 | 0.47 | None | 0.47 | 0.11 |
| LabelPropagation | 0.47 | 0.47 | None | 0.46 | 1.56 |
| LabelSpreading | 0.47 | 0.47 | None | 0.46 | 2.12 |
| PassiveAggressiveClassifier | 0.46 | 0.46 | None | 0.42 | 0.12 |
| Perceptron | 0.47 | 0.46 | None | 0.43 | 0.08 |
| ExtraTreeClassifier | 0.46 | 0.45 | None | 0.44 | 0.06 |
| AdaBoostClassifier | 0.38 | 0.39 | None | 0.38 | 2.68 |
| SVC | 0.37 | 0.37 | None | 0.34 | 1.57 |
| NearestCentroid | 0.32 | 0.33 | None | 0.18 | 0.05 |
| BernoulliNB | 0.31 | 0.32 | None | 0.15 | 0.07 |
| DummyClassifier | 0.31 | 0.31 | None | 0.30 | 0.05 |
| GaussianNB | 0.29 | 0.30 | None | 0.25 | 0.04 |

- **Testing dataset**

# Deep Learning - Results

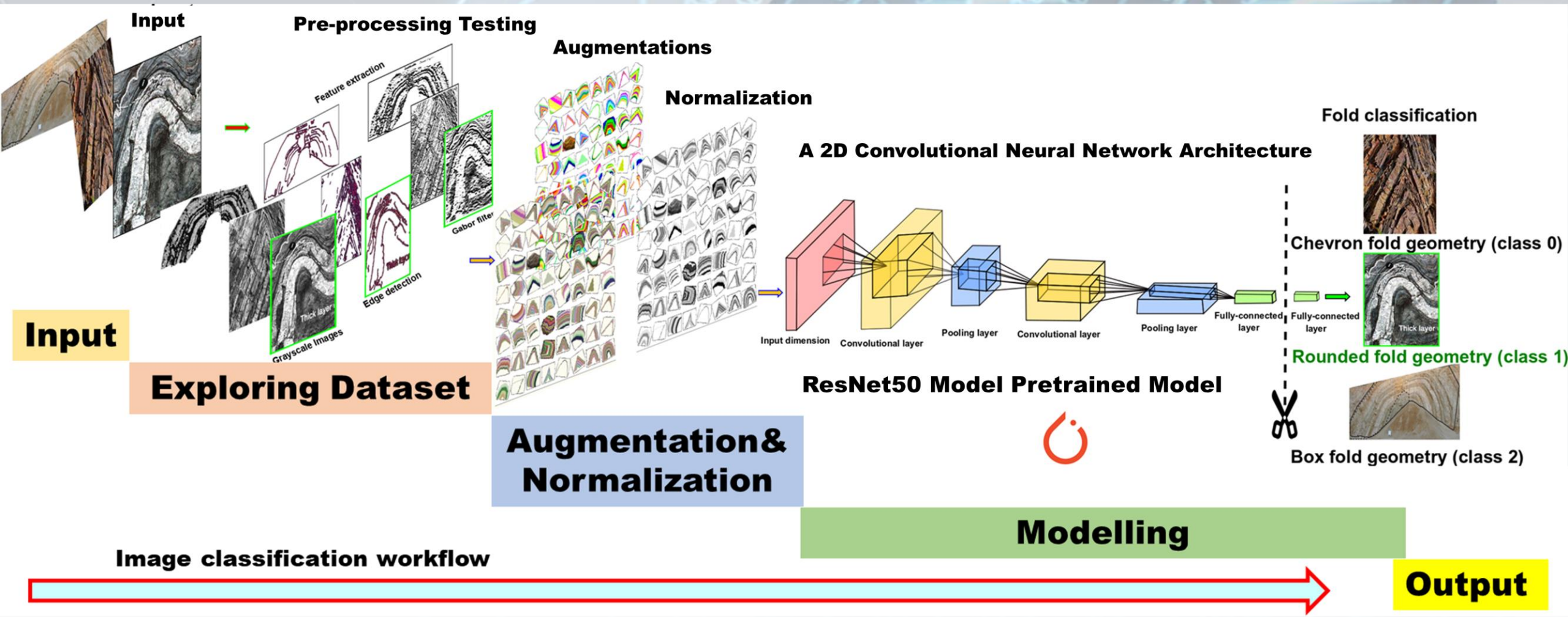


**ResNet50 Pretrained Model Accuracy over 70%**

| Learning | Model | Training | Types of Data | Accuracy (%) | Drive (CPU) - time | Colud (GPU) - time |
|---|---|---|---|---|---|---|
| Shallow | Random Forest | Untrained | Test | 57 | 13min 3s | - |
| | | | Val | 64 | | |
| Deep | ResNet9 | Untrained Greyscale | Test | 61.7 | 2h 12min 43s | 20min 35s |
| | | | Val | 55.7 | | |
| | | Untrained Coloured | Test | 69 | 2h 2min 14s | 22min 45s |
| | | | Val | 60 | | |
| Transfer | ResNet34 | Untrained | Test | 42.6 | 2h 4min 54s | 25min 15s |
| | | | Val | 45 | | |
| | | Pretrained | Test | 63 | 2h 7min 25s | 25min 9s |
| | | | Val | 60 | | |
| | ResNet50 | Untrained | Test | 33.8 | 3h 52min 48s | 3h 27min 2s |
| | | | Val | 45 | | |
| | | Pretrained | Test | **70.6** | 3h 14min 58s | 30min 35s |
| | | | Val | **62** | | |
| | ResNet101 | Untrained | Test | 39.7 | 3h 8min 15s | 32min 11s |
| | | | Val | 38.9 | | |
| | | Pretrained | Test | 61.7 | 4h 58min 34s | 44min 9s |
| | | | Val | 61 | | |
| | ResNet152 | Untrained | Test | 33.8 | 3h 33min 1s | 34min 7s |
| | | | Val | 36.7 | | |
| | | Pretrained | Test | 66.2 | 6h 1min 48s | 58min 6s |
| | | | Val | 61.3 | | |

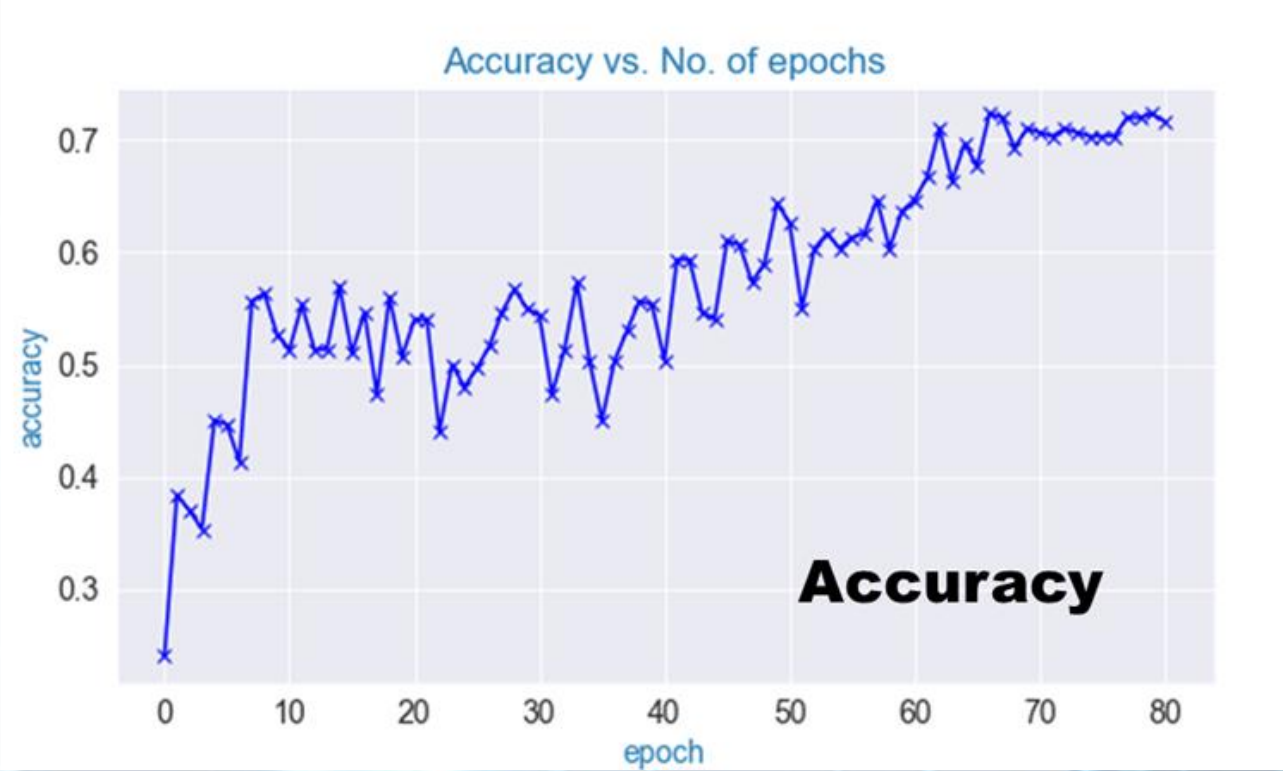
# Transfer Learning workflow

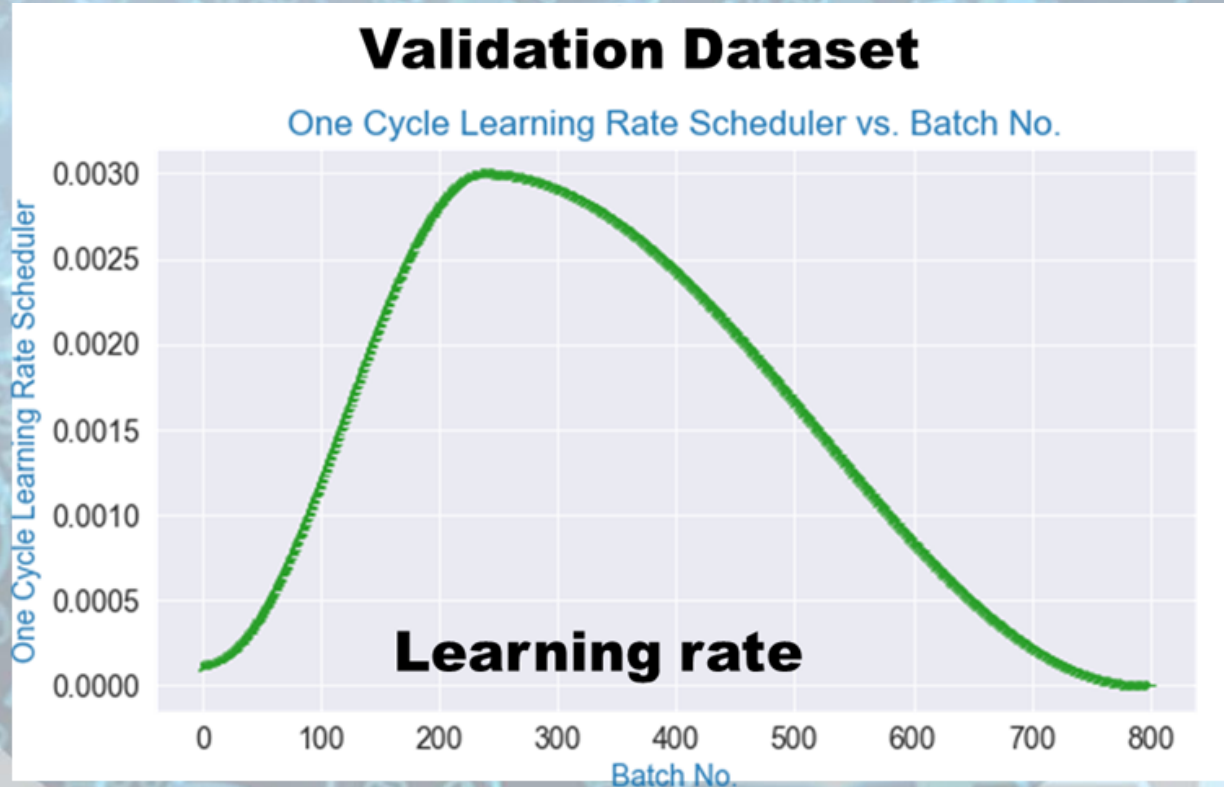




# Model Evaluation

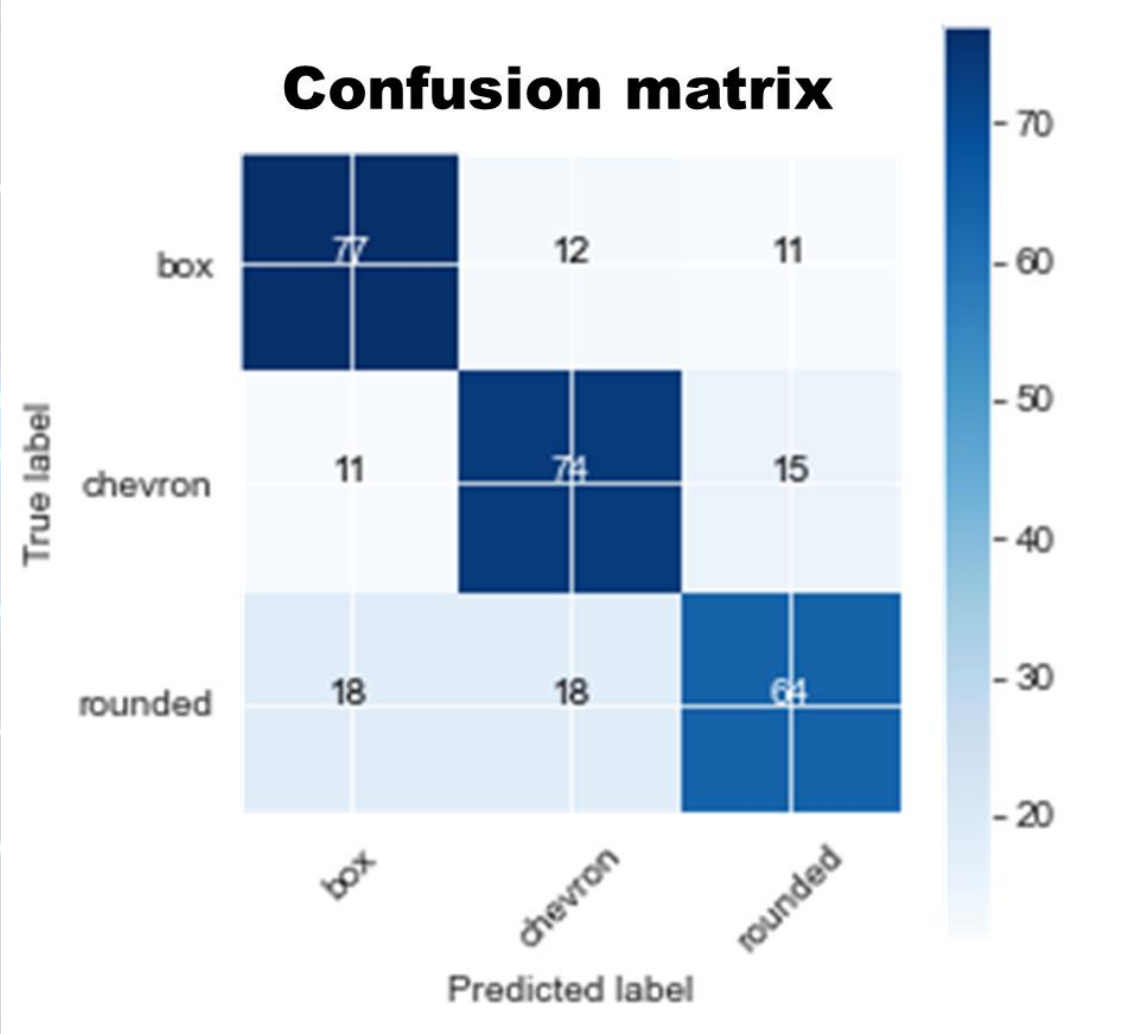


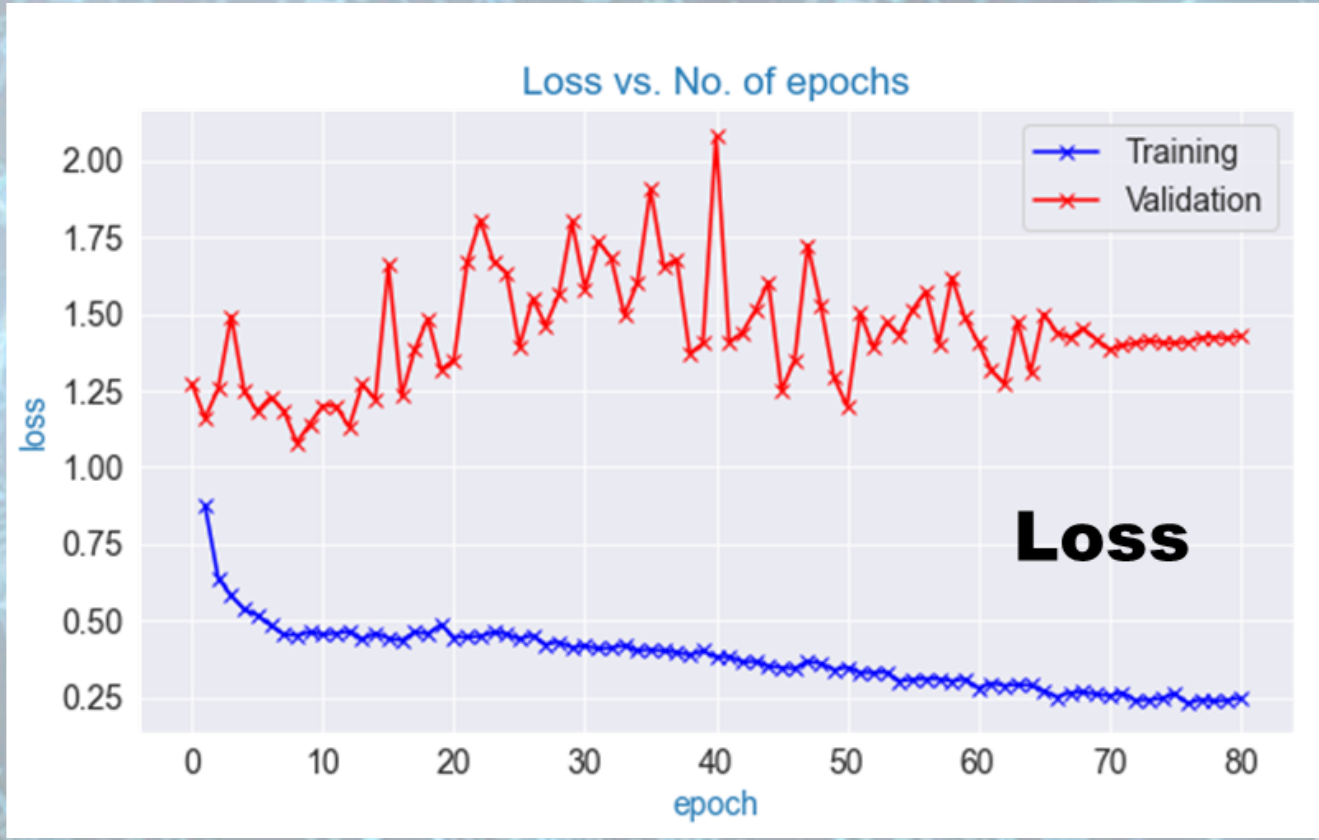


# Model Evaluation

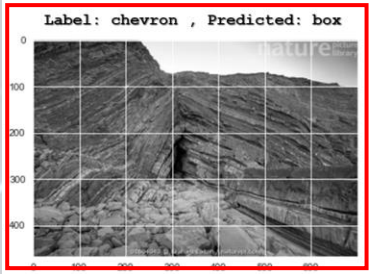


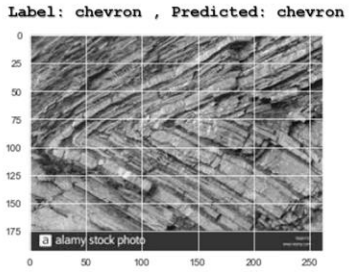


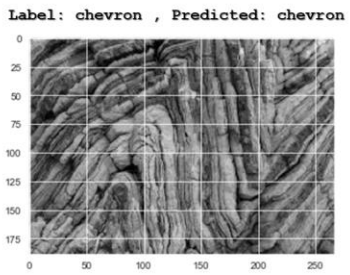



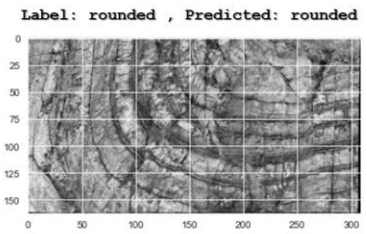
# Classification Examples

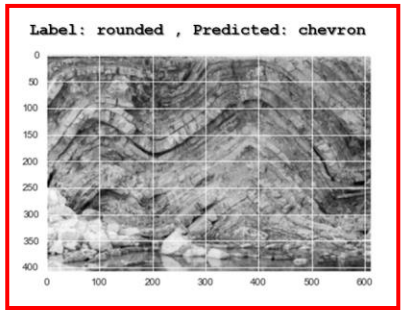


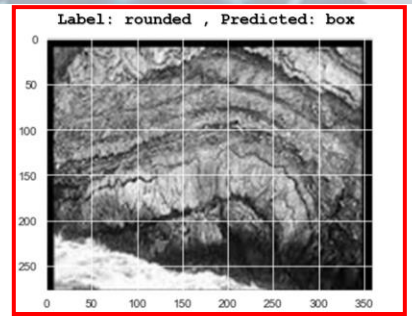


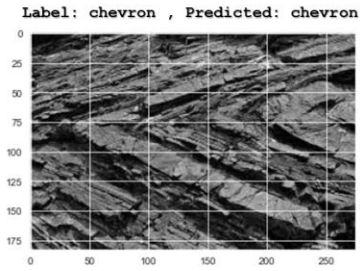



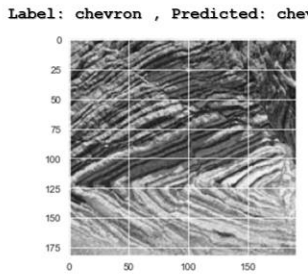



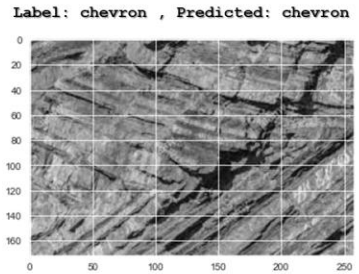



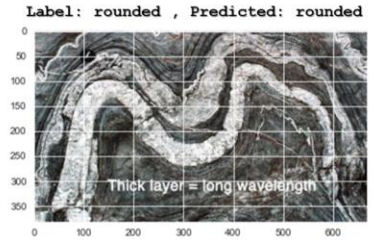



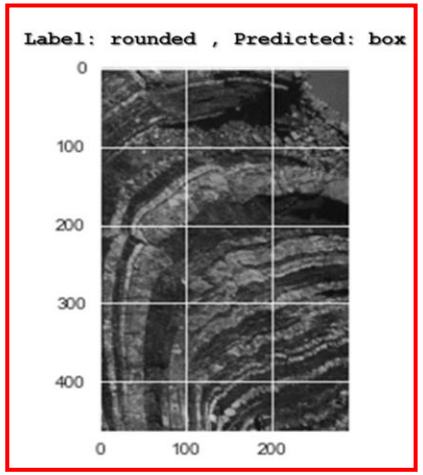



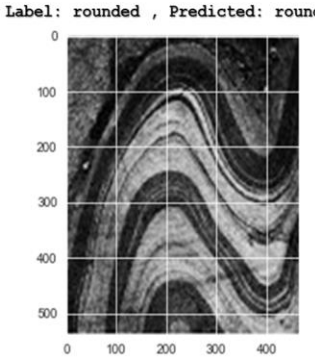



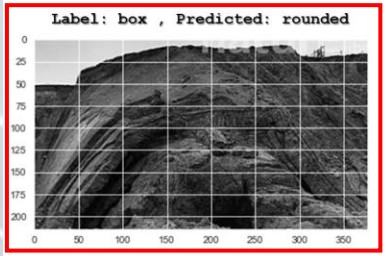



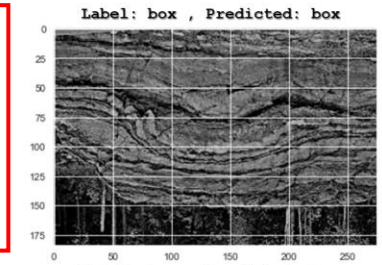



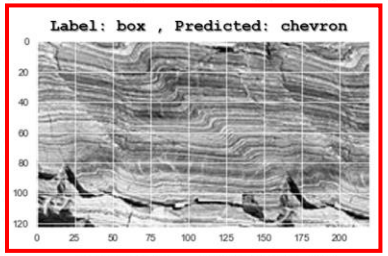



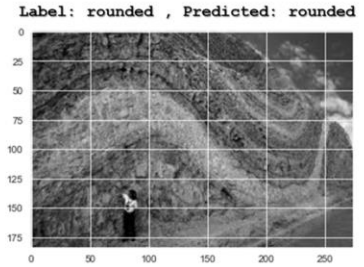



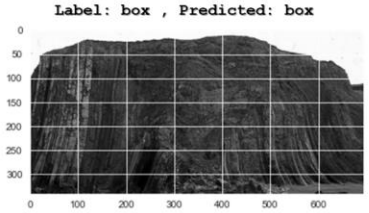



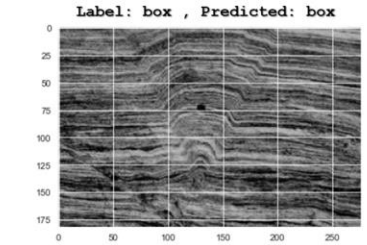



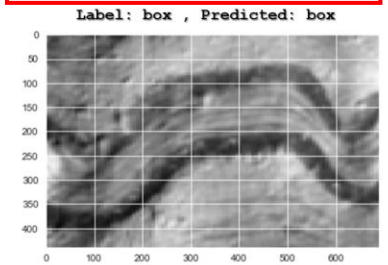



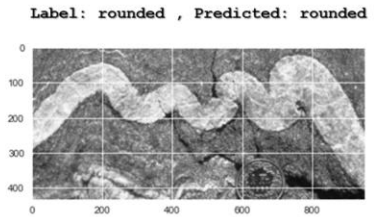



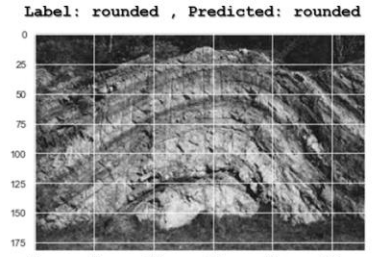



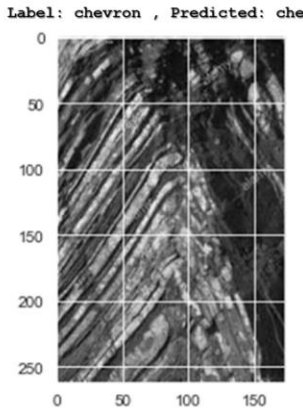



# Conclusions

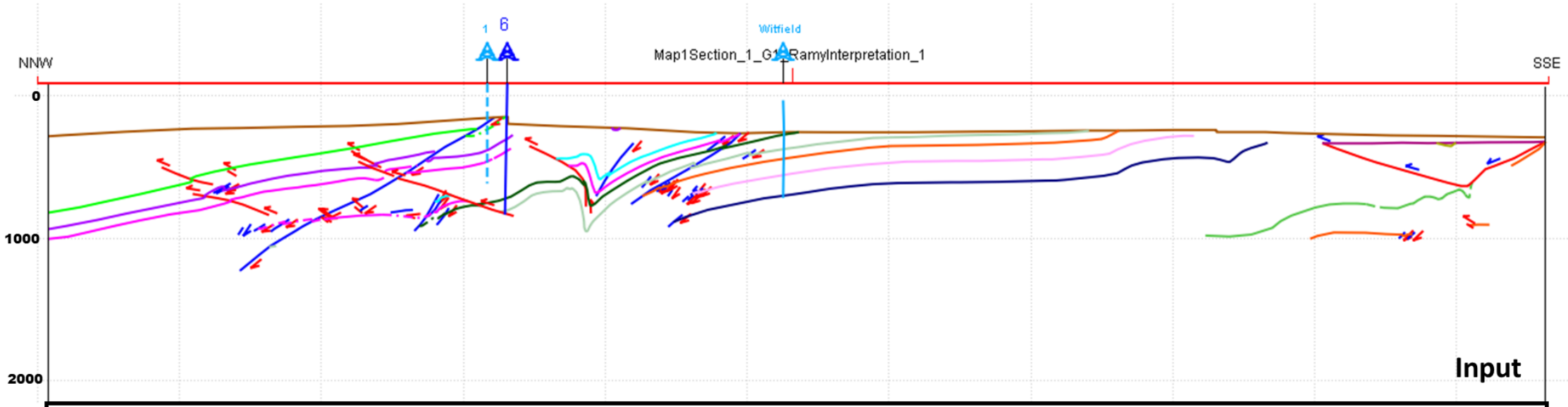


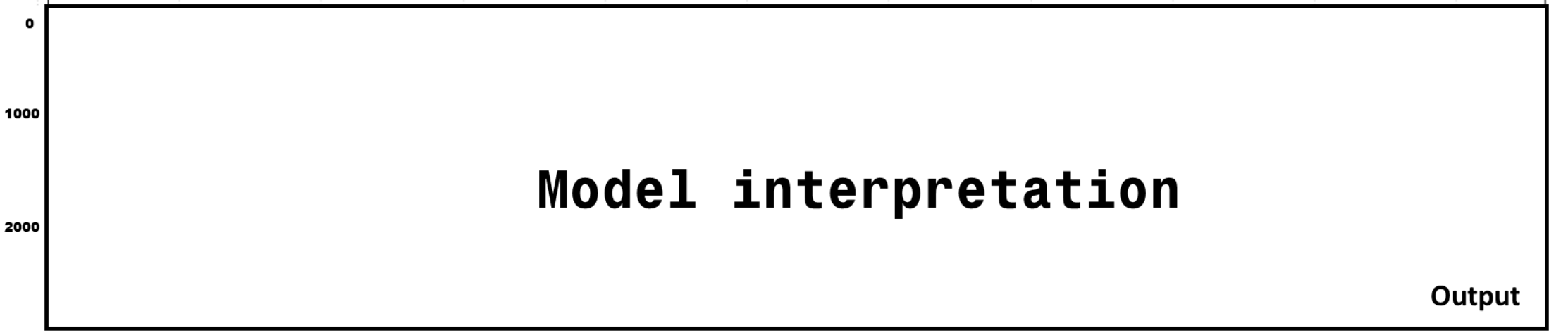
- We proposed modelled **data**, augmentation and normalization as methods to overcome geosciences data limitations.
- **Shallow learning** and feature extraction can be made prior to any deep learning study.
- **Convolutional Neural Networks** (CNNs) can help addressing geosciences classification problems.
- **Deep transfer learning** doing better in this case study.
- **Resnet50** model record the higher accuracy in our study.
- Can deductions be drawn from machine and deep learning studies?

# Future work



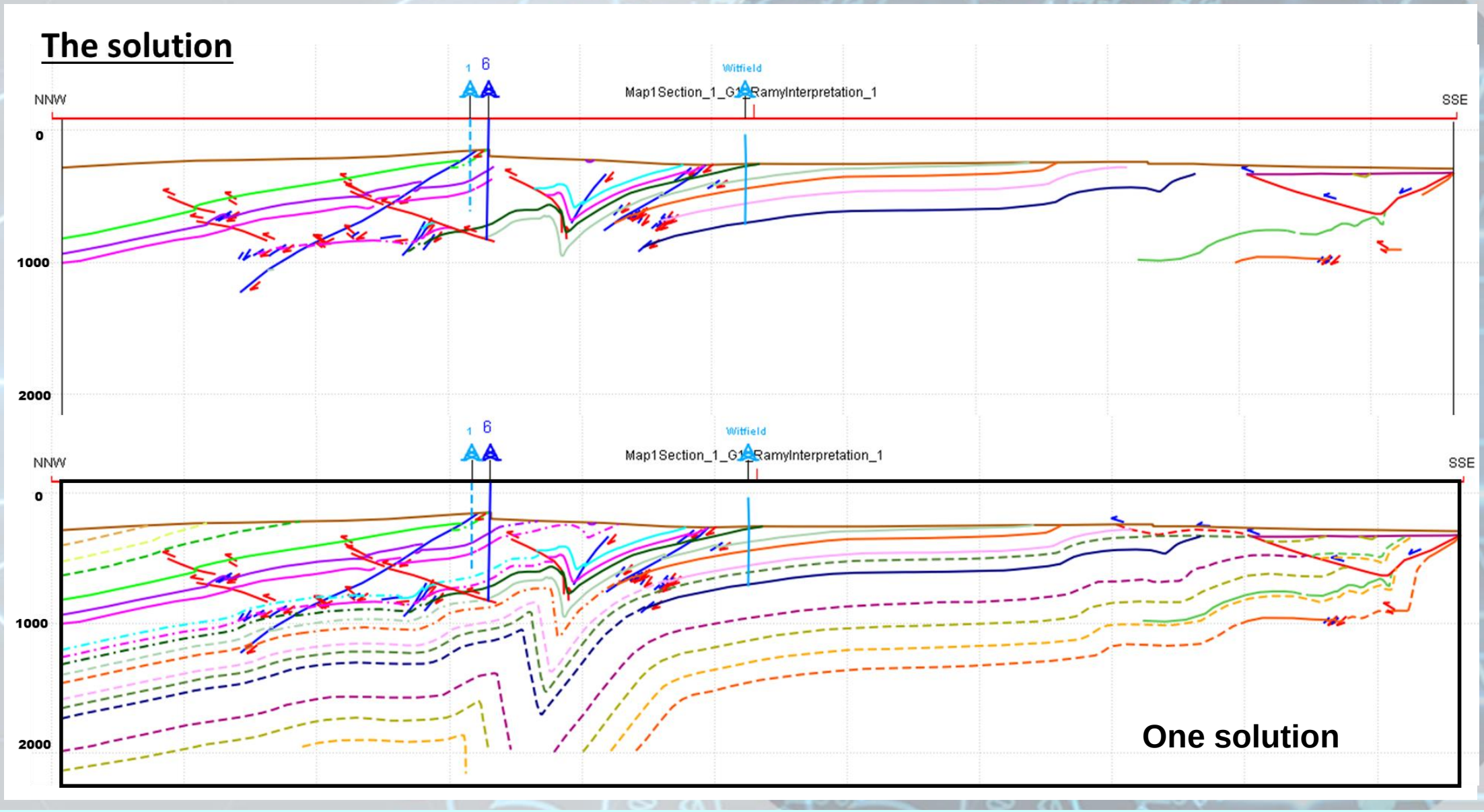
## Example of Test data



# Future work

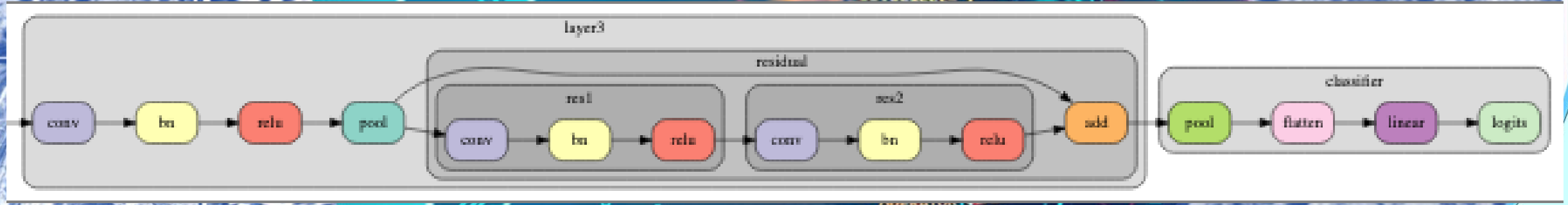



The solution
1
6
Witfield
Map1Section_1_G1RamyInterpretation_1
NNW
SSE
0
1000
2000
One solution


r.abdallah.18@abdn.ac.uk

Natural Environment Research Council

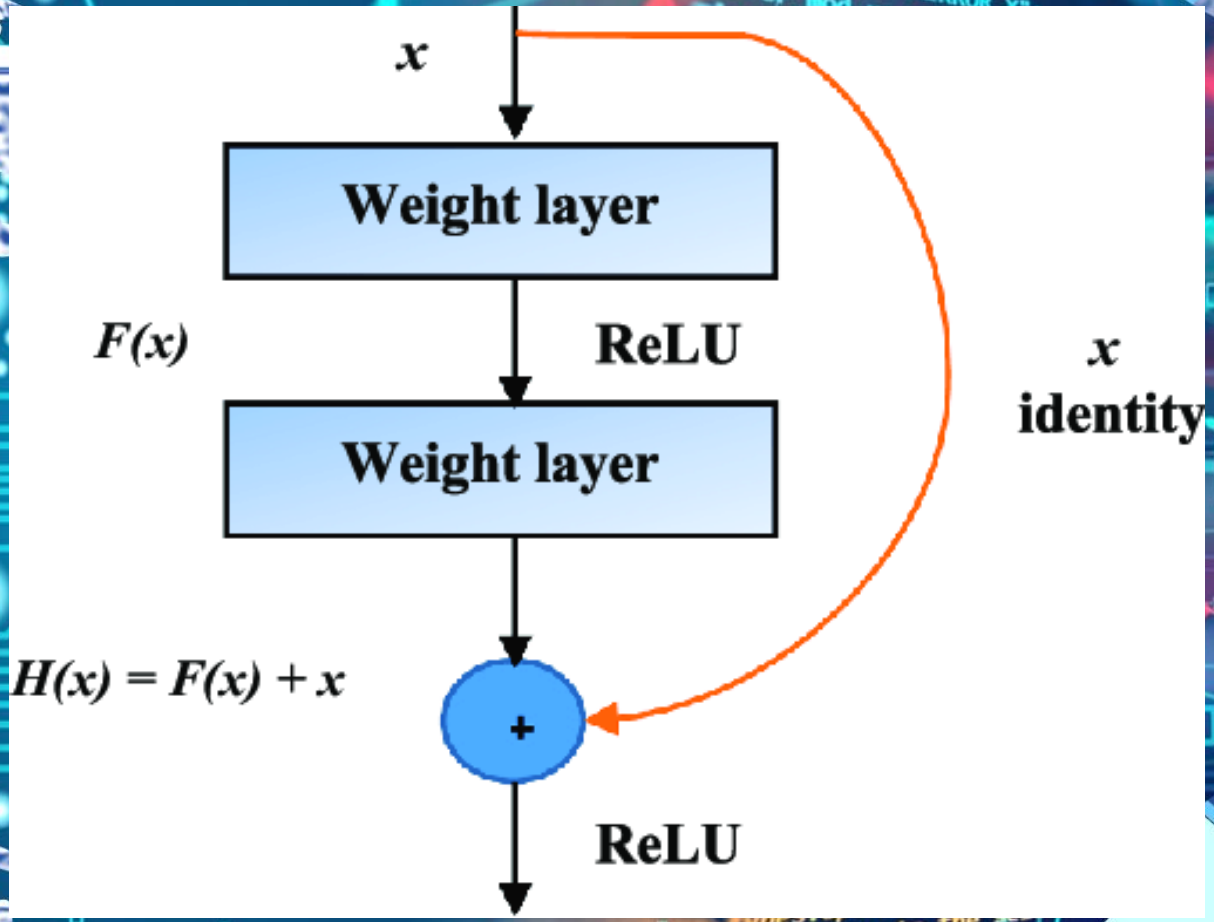
Centre for Doctoral Training (CDT) in Oil and Gas


1495 UNIVERSITY OF ABERDEEN

Natural Environment Research Council

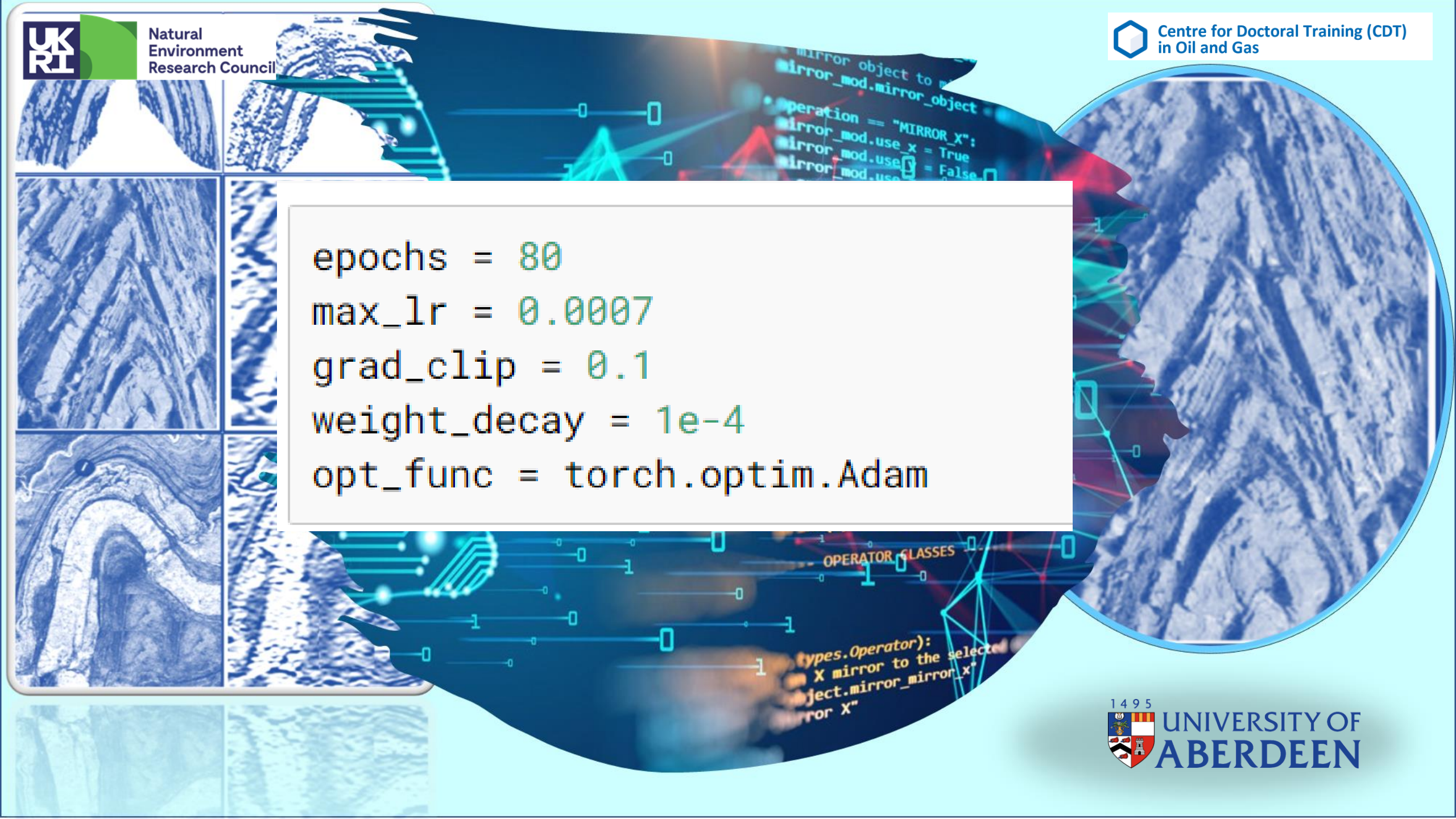
Centre for Doctoral Training (CDT) in Oil and Gas

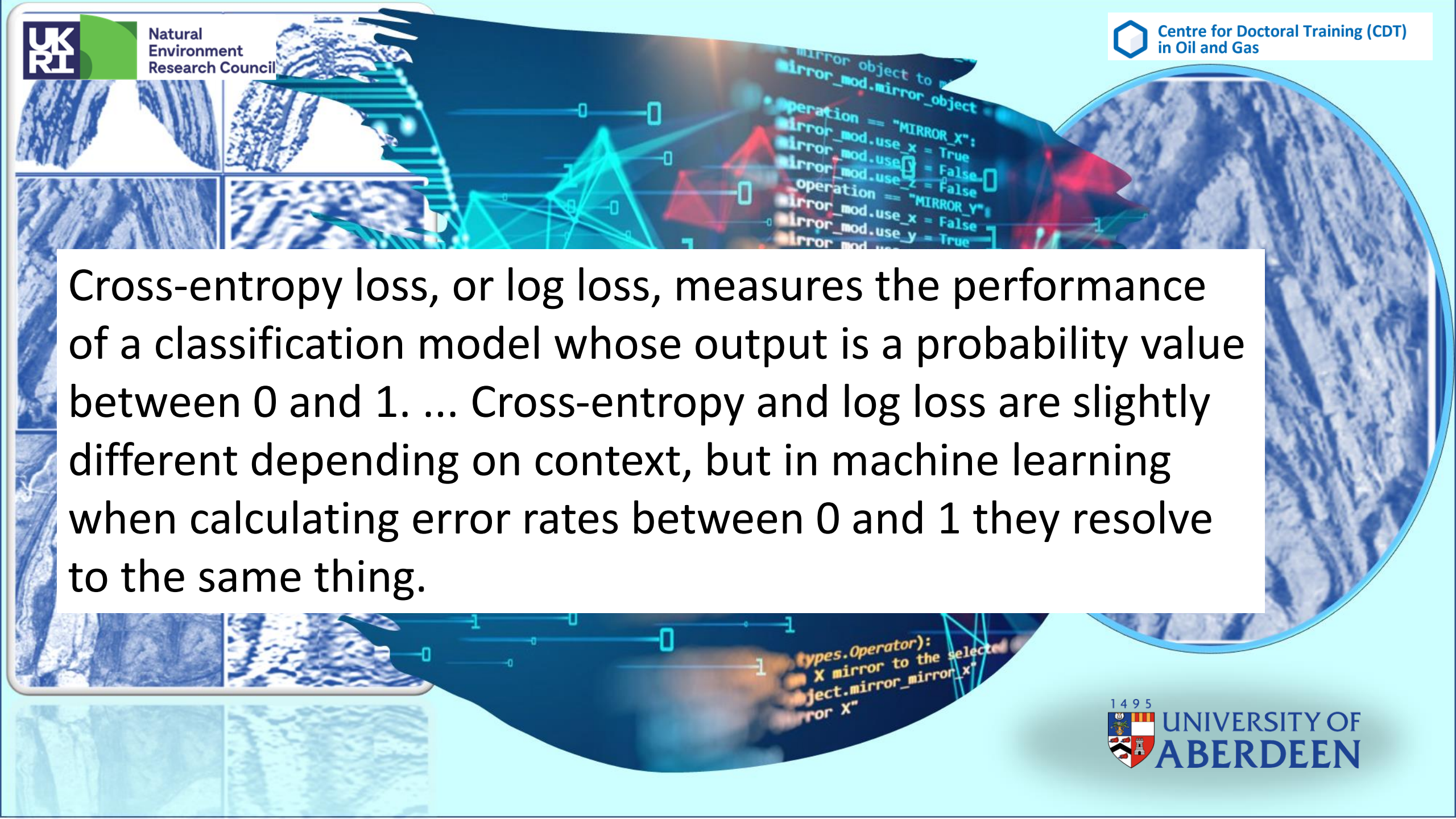



1495 UNIVERSITY OF ABERDEEN

```
# Data transforms (normalization & data augmentation) # tt.RandomCrop(64, padding=4, padding_mode='reflect')
stats = ((0.485, 0.456, 0.406), (0.229, 0.224, 0.225))
randomrotate = 10 # 0.175 rad
train_tfms = tt.Compose([tt.Resize((image_size, image_size)),
                         tt.RandomCrop(image_size, padding=4, padding_mode='reflect'),
                         tt.Resize(image_size),
                         tt.CenterCrop(image_size),
                         tt.RandomHorizontalFlip(),
                         tt.RandomVerticalFlip(),
                         tt.Pad((2, 5, 0, 5)),
                         #tt.RandomRotation(randomrotate),
                         tt.RandomAffine(0, translate=(0.4, 0.5)),
                         tt.RandomRotation(randomrotate, resample=PIL.Image.NEAREST, expand=False, center=(40,
                         #tt.RandomResizedCrop(256, scale=(0.5,0.9), ratio=(1, 1)),
                         tt.ColorJitter(brightness=0.1, contrast=0.1, saturation=0.1, hue=0.1), ##
                         tt.ToTensor(),
                         tt.Normalize(*stats,inplace=True)])
valid_tfms = tt.Compose([tt.Resize((image_size, image_size)), tt.ToTensor(), tt.Normalize(*stats)])

test_tfms = tt.Compose([tt.Resize((image_size, image_size)), tt.ToTensor(), tt.Normalize(*stats)])
```


Natural Environment Research Council
Centre for Doctoral Training (CDT) in Oil and Gas
epochs = 80
max_lr = 0.0007
grad_clip = 0.1
weight_decay = 1e-4
opt_func = torch.optim.Adam
1495 UNIVERSITY OF ABERDEEN

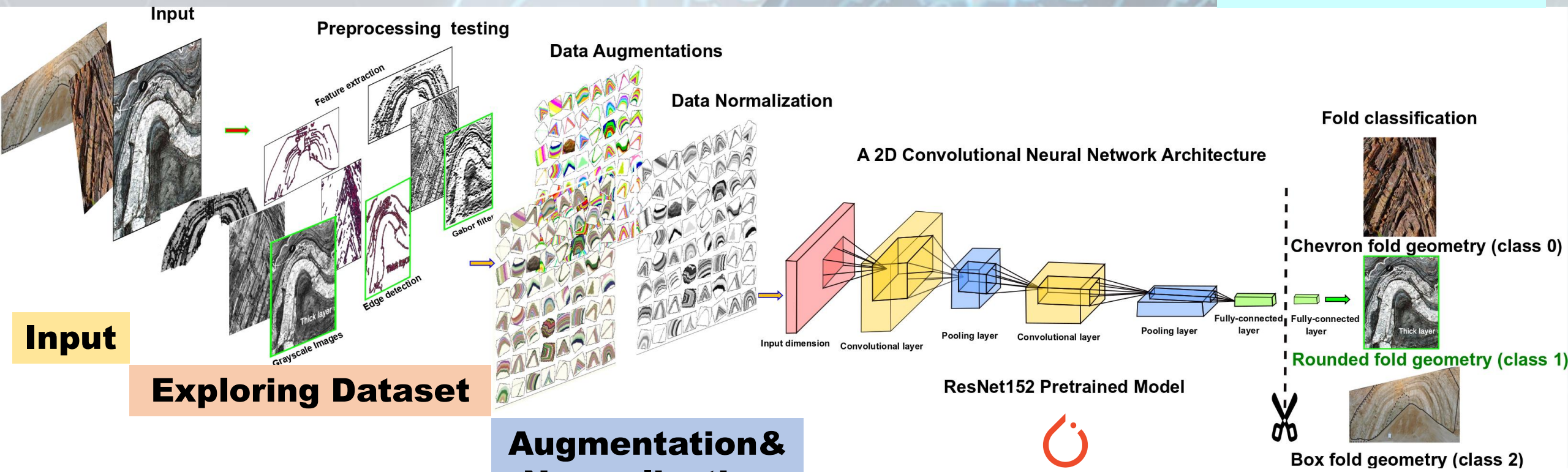

Natural Environment Research Council
Centre for Doctoral Training (CDT) in Oil and Gas
Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. ... Cross-entropy and log loss are slightly different depending on context, but in machine learning when calculating error rates between 0 and 1 they resolve to the same thing.
1495 UNIVERSITY OF ABERDEEN


UKRI
Natural Environment Research Council
Centre for Doctoral Training (CDT) in Oil and Gas
mirror_mod.mirror_object
operation == "MIRROR_X":
irror_mod.use_x = True
irror_mod.use_y = False
irror_mod.use_z = False
operation == "MIRROR_Y":
irror_mod.use_x = False
irror_mod.use_y = True
irror_mod.use_z = False
operation == "MIRROR_Z":
irror_mod.use_x = False
irror_mod.use_y = False
irror_mod.use_z = True
selection at the end -ad
_ob.select= 1
er_ob.select=1
ntext.scene.objects.acti
("Selected" + str(modifie
irror_ob.select = 0
bpy.context.selected_ob
ata.objects[one.name].se
int("please select exact
OPERATOR CLASSES
types.Operator):
X mirror to the selected
ject.mirror_mirror_x"
ror X"
1495
UNIVERSITY OF ABERDEEN

# Transfer Learning workflow



## ResNet50 Model




**Input**

**Exploring Dataset**

**Augmentation& Normalization**

**Modelling**

**Image classification workflow**

**Output**