Methods

This section highlights specific data preparation, design experiments and architectures, and model evaluation for this chapter.

Data Preparation

Image Generation in Pre-Processing: FMS Image Log and Core Image

The FMS image logs used in this study underwent data restoration by the IODP team, which included speed, button, EMEX voltage corrections, equalization, and depth-shifting (Isern et al., 2002; Davies et al., 1991; Betzler et al., 2017). The depth-shifting process aligned the logging runs to a common scale by matching distinctive features of the natural gamma log (Isern et al., 2002; Davies et al., 1991; Betzler et al., 2017).

Custom Python code was used for further pre-processing. The code, implemented in Python version 3.8 within a Jupyter Notebook environment, utilized libraries such as NumPy (version 1.23.5), pandas (version 1.5.1), OpenCV-python (version 4.7.0.68), and Dlisio (version 0.3.7). The code is publicly available on GitHub https://github.com/ssaira267/fms_to_image.

Data Labelling Based on Dunham Classification

The core and corresponding FMS images were labelled into five textural classes using the modified Dunham classification (Figure 4.1; Dunham, 1962; Embry and Klovan,1971) based on the depositional texture, grain content, and lime mud content.

Data Combination

Wireline log data with different vertical resolutions and sampling rates, and FMS images are combined to create a uniform dataset with consistent data points across all logs at specified depths. To ensure data integrity and completeness missing values were dealt with. This is crucial as missing data can lead to biased and inaccurate analysis (Emmanuel et al., 2021). I used the '*dropna*' function from the *pandas* library to remove any incomplete or undefined entries from the dataset. The '*dropna*' function scans through the dataset and drops entire rows (i.e. observations at a given depth below the seafloor) containing missing values thereby ensuring that only complete and valid data points are retained for the training and testing.

Data Split and Balancing

The datasets were split into a training set (80% of the data) and a testing set (20%) using a geographic train-test-split to minimize spatial co-linearity between the training and testing sets. The training set is further divided into training (75% of the data) and validation set (25% of the data). Since the dataset is imbalanced where wackestone and rudstone are minority classes, I performed data balancing using '*RandomOverSampler*' (ROS) on both FMS images and wireline log data. The ROS provided in the '*imbalanced-learn*' library, is a technique used for dealing with imbalanced datasets by oversampling the minority classes. The minority class samples are randomly selected and duplicated until the class distribution is balanced (Batista et al., 2005; Zheng et al., 2015; Shelke et al., 2017; Mohammed et al., 2020). As the wireline log data and FMS images are tied together based on their respective depth below the seafloor, oversampling was performed for both image data and tabular data to ensure that the oversampling process maintains the correspondence between the two modalities. This approach ensures that both FMS image data and wireline log data have the same number of samples.

Design Experiments and Architectures

Objective 1: Performance of Dunham textural classification from FMS data using transfer learning on a VGG19 architecture

Transfer learning using the VGG19 model was implemented. The pre-trained weights of the feature extraction portion of the network were used to extract generic features from the FMS images. The original classification layer of the VGG19 model was replaced with a new classifier, designed, and trained using the FMS image datasets. This new classification head comprised a global average pooling layer, two fully connected layers with 4096 and 1024 hidden units, respectively, utilizing ReLU activation, a dropout rate of 0.2 for regularization, and a final fully connected SoftMax layer due to the multiclass nature of our task. The feature embedding layer thus has a dimension of 4096 neurons, which is what we also used later in XGBoost (see objective 3). The model was set to train for a total of 200 epochs with *Early Stopping* implemented to prevent overfitting by monitoring validation loss with a patience parameter of 5 (Prechelt, L., 2012; Srivastava et al., 2014). During initial training, only the classification head was trained using a batch size of 128 and an Adam optimizer with a learning rate (Lr) of 0.001 while keeping the rest of the layers frozen. Subsequently, fine-tuning was performed, including unfreezing the uppermost two layers of the feature extractor followed by conducting adjustments using different learning rates (0.0001 and 0.00001).

Objective 2: Performance of Dunham texture classification with XGBoost from wireline log data

A total of 2 XGBoost models were trained based on 2 different datasets as described in section 5.4.1. Although the dataset is a combination of wireline log data and FMS images, in this study, only wireline log data are fed into the XGBoost model. The rock classes were encoded using the label encoder from *scikit-learn*. The tree-related parameters were set to their default settings during the training with a maximum depth of 6 for each tree in the ensemble, a learning rate of 0.3 for each boosting iteration, and a gamma set of 0 denoting no minimum loss reduction made to further partition on a leaf note (Chen and Guestrin, 2016). In addition, Early stopping was used to prevent overfitting the training data. The parameter *'early_stopping_rounds'* was set to '10', which means that the model's performance on a validation dataset is monitored after each boosting round. If there is no improvement in performance for 10 consecutive rounds, training will be stopped, and the best model observed during this monitoring period will be returned.

Objective 3: Performance of Dunham texture classification with XGBoost using FMS image data embedding from VGG19

XGBoost was used to analyse FMS images in comparison to objectives 1 and 2. While XGBoost is highly effective for classification and regression with tabular data, applying it directly to raw image data without further processing is generally not feasible. Therefore, features from the FMS images were extracted using the pre-trained VGG19 model, similar to the approach used in Objective 1. These features were then inputted into the XGBoost algorithm for final classification. The training parameters used are similar to Objective 2.

Evaluation

The VGG19- and XGBoost-trained models' performance was evaluated by monitoring the training and validation losses, as well as accuracy curves across each epoch during model training. The training loss indicates the error between predicted and true values on the training set, while the validation loss reflects the model's performance on a separate validation set. These loss curves offer valuable insights into how well the model is learning from the data and whether it risks overfitting or underfitting. Conversely, accuracy curves illustrate how correct overall predictions are. The training accuracy curve shows performance on the training set, while the validation accuracy curve demonstrates generalization capability to unseen data. Analysing these curves helps us understand how well our model learns throughout its training process; steady improvement in both can indicate effective learning, whereas a large gap may suggest overfitting.

In addition, for XGBoost, we referred to the XGBoost log loss and XGBoost classification error log. The log loss is a measure of how well the model's predicted probabilities align with the true class labels. The goal during training is to minimize this loss, essentially optimizing the model to make better probability estimates for each class. The XGBoost classification error is a metric that directly assesses the number of misclassifications made by the model. It is the ratio of incorrectly classified instances to the total number of instances. Evaluation metrics, such as accuracy, precision, recall and F1-score, were used to quantitatively assess the model's performance on the test dataset.