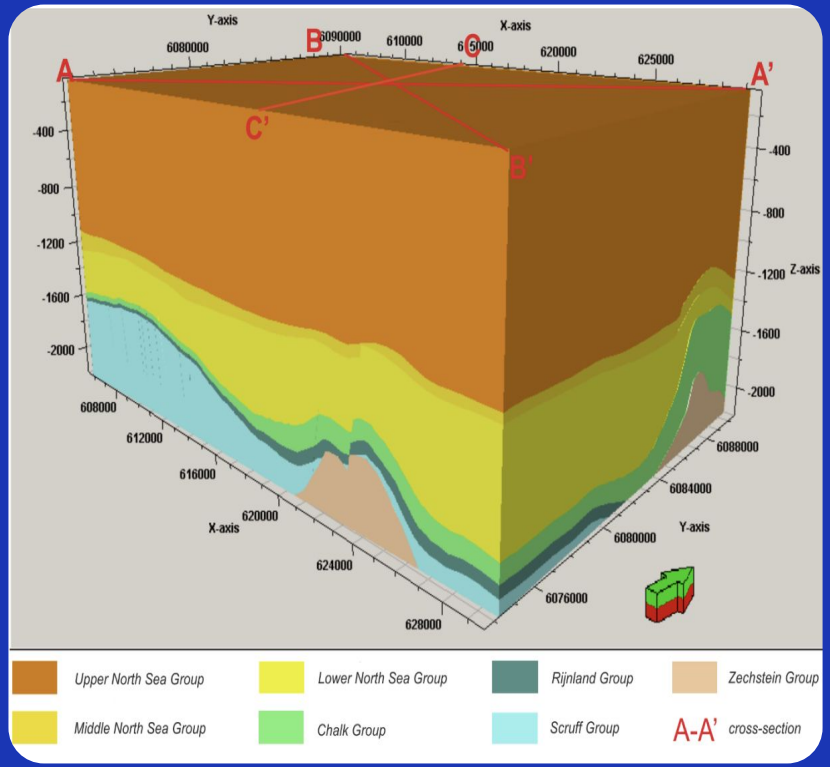
LIVE AI

FACIES CLASSIFICATION

Omar Ashraf
Mazen Hassan



Approach

01. Download Dataset

02. Data Configuration & Augmentation

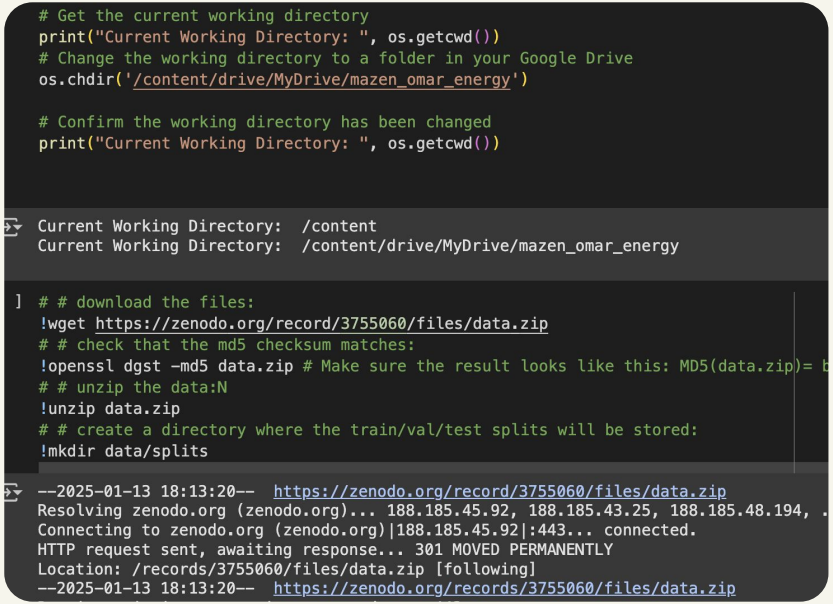
03. Neural Network Architectures

04. Training & Testing

05. Flask Web App

Downloading the Dataset

- We downloaded the **Dutch F3** dataset from the link provided in the task description.
- Extracted the files and saved it.
- The data is now ready for loading and splitting.



```
# Get the current working directory
print("Current Working Directory: ", os.getcwd())
# Change the working directory to a folder in your Google Drive
os.chdir('/content/drive/MyDrive/mazen_omar_energy')

# Confirm the working directory has been changed
print("Current Working Directory: ", os.getcwd())
```

```
Current Working Directory:  /content
Current Working Directory:  /content/drive/MyDrive/mazen_omar_energy
```

```
# # download the files:
!wget https://zenodo.org/record/3755060/files/data.zip
# # check that the md5 checksum matches:
!openssl dgst -md5 data.zip # Make sure the result looks like this: MD5(data.zip)= b
# # unzip the data:N
!unzip data.zip
# # create a directory where the train/val/test splits will be stored:
!mkdir data/splits
```

```
--2025-01-13 18:13:20--  https://zenodo.org/record/3755060/files/data.zip
Resolving zenodo.org (zenodo.org)... 188.185.45.92, 188.185.43.25, 188.185.48.194, .
Connecting to zenodo.org (zenodo.org)|188.185.45.92|:443... connected.
HTTP request sent, awaiting response... 301 MOVED PERMANENTLY
Location: /records/3755060/files/data.zip [following]
--2025-01-13 18:13:20--  https://zenodo.org/records/3755060/files/data.zip
```

Data Configuration & Augmentation

This step is divided into loading labels, splitting data into training and validations sets, augmenting data, and initializing data loaders.

01

LOADING

Loaded the ‘train_labels.npy’ file for patch-based splitting.

02

SPLITTING

Created inline and crossline patches for training and validation. Then split the dataset for relative splits (80:20).

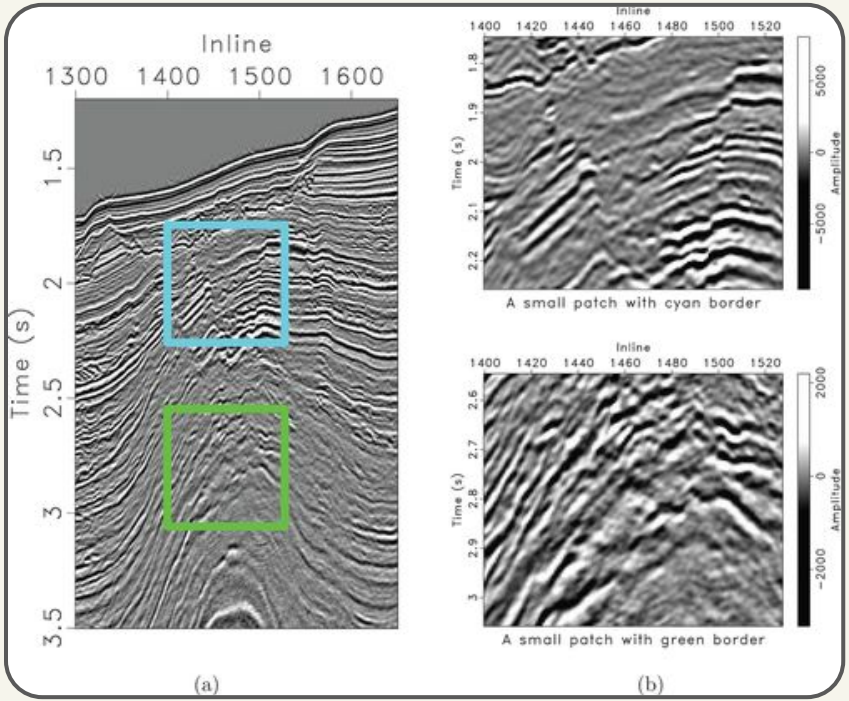
03

AUGMENTATION

Augmented the data using .py scripts in an attempt to reduce overfitting and class imbalance.

Augmentation Plan

- The aim for this approach was to decrease **overfitting** and **class imbalance**.
- Data augmentation was applied to the training set **only**.
- The data was then shuffled to make sure **memorization** doesn't occur.
- We created **data loaders** for both the training and validation sets.



Neural Networks Architecture

U-Net

Utilizes an **encoder-decoder** structure with **skip connections**. The encoder **extracts** features from the seismic data and the skip connections **preserve** the spatial details. This architecture is very **effective** for this type of feature extraction.

ResNet-34

34-layer residual network with **shortcut connections** to mitigate **vanishing gradients**. Pre-trained model, and is **fine-tuned** on the seismic data to adapt to such patterns. It is **lightweight** making it ideal for moderate dataset sizes.

ResNet-50

Deeper 50-layer variant with **bottleneck** layers. Enhanced **feature representation** due to increased depth; captures the **complex** seismic structures. As a result, higher computational cost but **improved accuracy**.

InceptionV3

Employs **parallel convolutions** (Inception modules) for **multi-scale feature extraction**. This architecture is **optimized** for efficiency, and excels at feature detection making it **critical** for **diverse** seismic data.

Hyperparameter Tuning

ResNet-34

Parameters Explored:

Batch size: 16, 32, 64

Learning rate: 0.01, 0.001, 1e-4

Search Strategy: Grid

ResNet-50

Parameters Explored:

Batch size: 16, 32, 64

Learning rate: 0.01, 0.001, 1e-4

Dropout: 0.1, 0.2, 0.3, 0.4, 0.5

Weight Decay: 1e-5

Search Strategy: Random

InceptionV3

Parameters Explored:

Batch size: 16, 32, 64

Learning rate: 0.01, 0.001, 1e-4

Search Strategy: Random

Optimal Findings and Challenges

Optimal Findings:

ResNet-34
Batch size: 64
Learning rate: 1e-4

ResNet-50
Batch size: 64
Learning rate: 1e-4
Weight Decay: 1e-5

InceptionV3
Batch size: 64
Learning rate: 1e-4

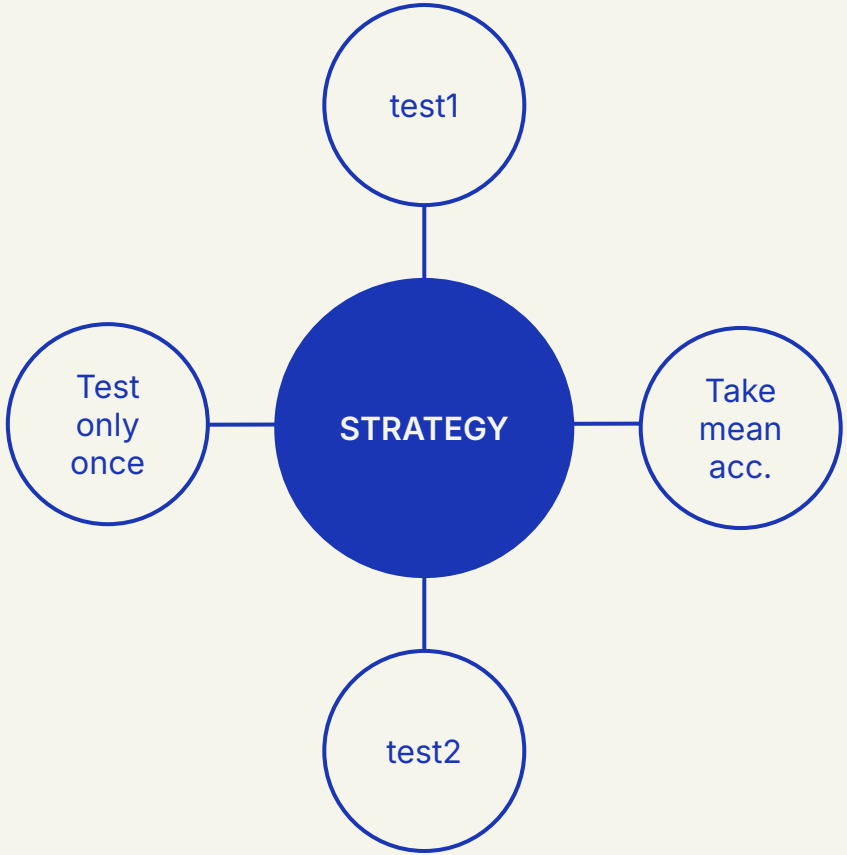
Challenges:

ResNet-50:

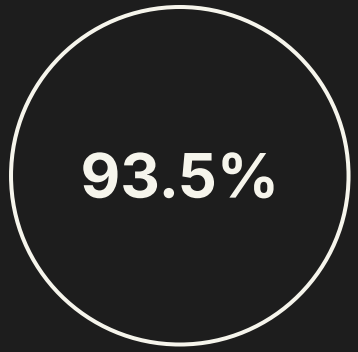
- Prolonged training time and unusually high training/validation losses.
- Dropout implementation provided little improvement.
- Solution was introducing L2 Regularization and weight decay in the loss function, which helped reduce overfitting

Testing Strategy

- We have **two** test sets.
- Test the model on each set **only once**.
- Take **average** overall accuracy of each set as the accuracy of the model.

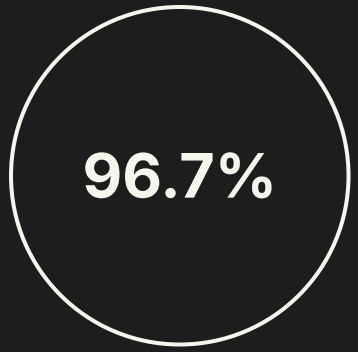


Model Testing Accuracies



ResNet-34

Offers a simpler architecture and faster training times, but lacking that extra accuracy.



ResNet-50

Achieves higher accuracy but requires more computational resources.

InceptionV3

Provides the best overall performance, balancing speed and accuracy effectively.

Flask Web App

Downloaded .pth files and integrated into Flask app.

Created a simple responsive interface which lets users choose a model.

The selected model segments the image, and results are displayed

Web App Content

Home page

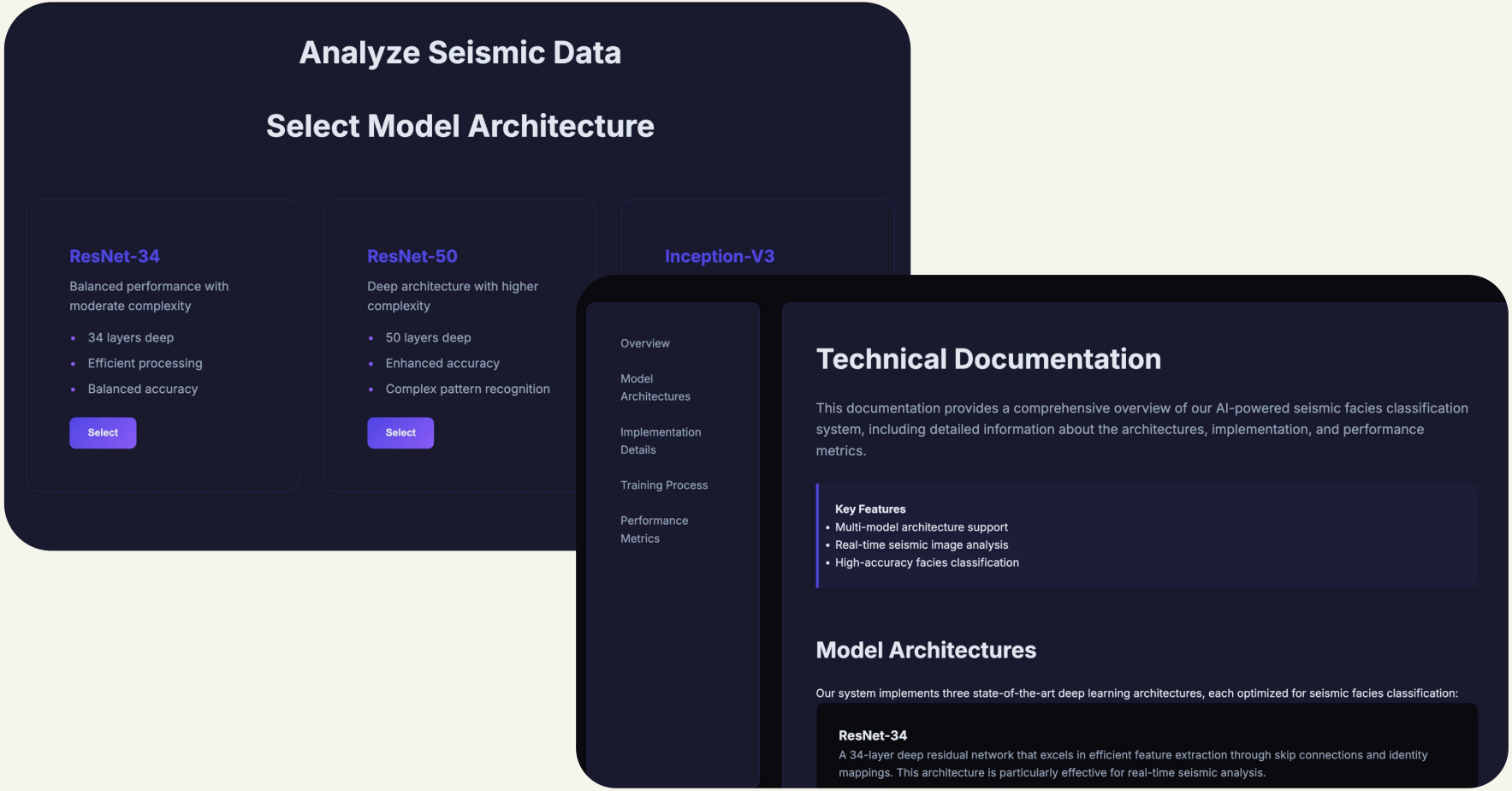
This section features a central title and two call-to-action buttons: one that directs users to the models page and another that opens the documentation.

Models page

This section displays three cards, each representing a model with a short description. When a user selects a model, they can upload an image, and the classification results are shown afterwards.

Documentation page

The documentation page offers detailed information on the models and explains how the system works for those who want to learn more.


Analyze Seismic Data
Select Model Architecture
ResNet-34
Balanced performance with moderate complexity
34 layers deep
Efficient processing
Balanced accuracy
Select
ResNet-50
Deep architecture with higher complexity
50 layers deep
Enhanced accuracy
Complex pattern recognition
Select
Inception-V3
Overview
Model Architectures
Implementation Details
Training Process
Performance Metrics
Technical Documentation
This documentation provides a comprehensive overview of our AI-powered seismic facies classification system, including detailed information about the architectures, implementation, and performance metrics.
Key Features
Multi-model architecture support
Real-time seismic image analysis
High-accuracy facies classification
Model Architectures
Our system implements three state-of-the-art deep learning architectures, each optimized for seismic facies classification:
ResNet-34
A 34-layer deep residual network that excels in efficient feature extraction through skip connections and identity mappings. This architecture is particularly effective for real-time seismic analysis.

Summary

- **Dataset Preparation:** Utilized the Dutch F3 dataset, processed through extraction, splitting, and augmentation to address class imbalance and overfitting.
- **Model Architectures:** Explored U-Net (backbone), ResNet-34, ResNet-50, and InceptionV3, focusing on feature extraction and performance optimization.
- **Hyperparameter Tuning & Training:** Applied grid and random search methods; fine-tuned batch sizes, learning rates, dropout, and weight decay for optimal results.
- **Testing Key Findings:** InceptionV3 emerged as the best-performing model with 97.2% accuracy, finding an ideal balance between computational efficiency and accuracy.
- **Flask Integration:** Created a responsive web app using best model paths, offering user-friendly interface, real-time image classification results, and informative documentation for users to learn more.

Any Questions?

Contact us:
Omar Ashraf
oomaraashrafaabdou@gmail.com

Mazen Sakr
mazensakr396@gmail.com

Thank You!