

Aim

The aim of this project is to develop a deep learning model capable of segmenting different components in sandstone images. This includes identifying and classifying various regions such as clay, quartz, and pyrite within the sandstone structure.

Objectives

1. Preprocess the dataset of sandstone images and their corresponding masks.
2. Develop and train a U-Net model for semantic segmentation of sandstone images.
3. Evaluate the performance of the trained model using accuracy and Intersection over Union (IoU) metrics.
4. Visualize the segmentation results on test images.

Summary

This project involves building a U-Net model to segment sandstone images into different classes, such as clay, quartz, and pyrite. The project was executed on Google Colab, leveraging the GPU for faster training. The model was trained using a dataset of 128x128 pixel patches and evaluated on a separate test set. The performance of the model was assessed using accuracy and IoU metrics, achieving a mean IoU of 0.8665 and an accuracy of 96.37%.

Tools and Libraries Used

- Python
- Google Colab
- Keras (TensorFlow backend)
- OpenCV
- NumPy
- Matplotlib
- scikit-learn

Procedure

1. Data Preprocessing

- **Splitting Multi-Page TIFF Files:** The multi-page TIFF files containing images and masks were split into individual 128x128 pixel patches.
- **Resizing Images:** The images and masks were resized to 128x128 pixels.
- **Encoding Labels:** The mask values were encoded into classes (background, clay, quartz, pyrite).
- **Normalizing Images:** The image pixel values were normalized to range between 0 and 1.

- **Splitting Data:** The dataset was split into training and test sets, with a further split of the training set for quick testing.

CODE:

```
# Example code snippets for the steps above

def split_tiff(input_path, output_dir):
    tiff_image = Image.open(input_path)
    for i in range(tiff_image.n_frames):
        tiff_image.seek(i)
        output_path = os.path.join(output_dir, f'page_{i + 1}.tif')
        tiff_image.save(output_path)
```

```
# Data splitting and encoding
input_tiff_path = '/path/to/tiff'
output_directory = '/path/to/output'
split_tiff(input_tiff_path, output_directory)
```

2. Model Development

- **U-Net Architecture:** The U-Net model was constructed with a contraction path (encoder) and an expansive path (decoder) for precise localization.
- **Compiling the Model:** The model was compiled with the Adam optimizer and categorical cross-entropy loss function.

CODE:

```
def multi_unet_model(n_classes=4, IMG_HEIGHT=256, IMG_WIDTH=256, IMG_CHANNELS=1):
    inputs = Input((IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS))
    s = inputs

    # Contraction path
    c1 = Conv2D(16, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(s)
    c1 = Dropout(0.1)(c1)
    c1 = Conv2D(16, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(c1)
    p1 = MaxPooling2D((2, 2))(c1)

    # Additional layers
```

```

# ...

outputs = Conv2D(n_classes, (1, 1), activation='softmax')(c9)

model = Model(inputs=[inputs], outputs=[outputs])

return model

model = multi_unet_model(n_classes=4, IMG_HEIGHT=128, IMG_WIDTH=128, IMG_CHANNELS=1)

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

```

3. Training and Evaluation

- **Training:** The model was trained for 50 epochs with a batch size of 16.
- **Evaluation:** The model's performance was evaluated using accuracy and IoU metrics.

CODE:

```

history = model.fit(X_train, y_train_cat, batch_size=16, epochs=50, validation_data=(X_test, y_test_cat),
shuffle=False)

_, acc = model.evaluate(X_test, y_test_cat)

print("Accuracy is = ", (acc * 100.0), "%")

```

4. Visualization and Results

- **Plotting Loss and Accuracy:** The training and validation loss and accuracy were plotted over epochs.
- **IoU Calculation:** The IoU for each class was calculated and displayed.

CODE:

```

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)

plt.plot(epochs, loss, 'y', label='Training loss')

plt.plot(epochs, val_loss, 'r', label='Validation loss')

plt.legend()

plt.subplot(1, 2, 2)

plt.plot(epochs, acc, 'y', label='Training Accuracy')

plt.plot(epochs, val_acc, 'r', label='Validation Accuracy')

plt.legend()

```

```
plt.show()
```

Highlights

- **U-Net Model:** The U-Net architecture with both contraction and expansion paths enabled precise segmentation.
- **Class Balancing:** The use of class weights ensured balanced training despite class imbalances.
- **High Accuracy:** Achieved a high accuracy of 96.37% and a mean IoU of 0.8665.
- **IoU Calculation:** Detailed IoU calculation for each class provided insights into model performance.

Conclusion

The project successfully developed a U-Net model for the segmentation of sandstone images, achieving high accuracy and mean IoU. The model's performance indicates its potential for precise identification and classification of different sandstone components. Future work can explore further optimization and the application of this model to larger and more diverse datasets.