Multi-Class Nuclei Segmentation Problem Report

Jiashan XU

I. Introduction

The project is a multi-class nuclei segmentation problem using deep learning techniques. The goal is to develop models capable of accurately classifying nuclei images into different types. The dataset used for this task consists of input images (`x`) and corresponding labels (`y`) indicating the types of nuclei present in each image.

II. Data Preparation

First, I load the input images and labels from the provided files using NumPy. The dataset is then split into training and testing sets with a 8:2 ratio using the `train_test_split` function. To facilitate training, the labels (`y_train` and `y_test`) are encoded using one-hot encoding. First, the labels are integer encoded using `LabelEncoder` from scikit-learn, and then one-hot encoded using `OneHotEncoder`. This ensures that the labels are represented as binary vectors suitable for multi-class classification. Finally, the training set is further split into training and validation sets with a 9:1 ratio using `train_test_split`.

III. Model

Model 1: Convolutional Neural Network (CNN)

The first model we use for the nuclei segmentation problem is a Convolutional Neural Network (CNN). The architecture of this model consists of several convolutional layers followed by max pooling layers. The flattened output is then passed through fully connected layers with dropout for regularization. The final layer uses softmax activation to produce class probabilities.

The model is compiled using the SGD optimizer with a learning rate of 0.001 and categorical cross-entropy loss. The training data is normalized by dividing by 255 to scale the pixel values between 0 and 1.

The model is trained on the training data for 35 epochs with a batch size of 8. The validation data is used for monitoring the training progress and avoiding overfitting. The training history is stored in the `history` variable.

After training, the model is evaluated on the testing data. The predicted labels are obtained using `argmax` to convert the predicted probabilities to class indices. The accuracy is then calculated using the `accuracy_score` function from scikit-learn.

Model 2: FCN

The second model is the FCN model. The model is compiled using the Adam optimizer, a learning rate of 0.001, and categorical cross-entropy loss. The pixel values of the training, validation, and testing data are normalized by dividing by 255.0. The model is trained on the training data for 35 epochs with a batch size of 8. The validation data is used for monitoring the training progress. After training, the model is evaluated on the testing data, and the test accuracy is calculated.

Model 3: ResNet50

Next, I explore the use of a pre-trained model called ResNet50 for nuclei segmentation. The ResNet50 model is initialized with pre-trained weights on the ImageNet dataset. We remove the top layers of the ResNet50 model and add our custom layers for nuclei classification.

The model is compiled with the SGD optimizer, a learning rate of 0.001, and categorical cross-entropy loss.

Similar to the CNN model, the data is normalized by dividing by 255, and the model is trained on the training data for 35 epochs with a batch size of 8. The validation data is used for monitoring the training progress.

After training, the model is evaluated on the testing data, and the test accuracy is calculated using the `accuracy_score` function.

Model 4: VGG19

For the last model, I employ the VGG19 architecture, another pre-trained model. The VGG19 model is loaded without the top layers, and custom layers are added for nuclei classification. The weights of the VGG19 layers are frozen to prevent them from being updated during training.

The model is compiled with the SGD optimizer, a learning rate of 0.0001, and categorical cross-entropy loss.

Data augmentation is applied using the `ImageDataGenerator` from Keras, which helps to artificially increase the size of the training dataset. The data is also normalized by dividing by 255.

The model is trained using the augmented training data for 40 epochs with a batch size of 8. The validation data is used for monitoring the training progress.

After training, the model is evaluated on the testing data, and the test accuracy is calculated.

IV. Evaluation

Model Name	Training Accuracy	Validation Accuracy	Test Accuracy
CNN	0.5050	0.5023	0.4756
FCN	0.9178	0.3991	0.3496
ResNet50	0.9859	0.6808	0.6654
VGG19	0.3121	2.5629	0.1485

The CNN model demonstrates relatively low accuracy across all three metrics. The potential reasons for this performance are as follows:

- Model Complexity: The CNN model has a simple architecture with a few convolutional and pooling layers, followed by fully connected layers. This limited complexity may not be sufficient to capture the intricate features and patterns required for accurate nuclei segmentation.
- Hyperparameters: The hyperparameters, such as the learning rate, batch size, and number of epochs, may not be optimized for this specific problem. For example, the number of epochs maybe should be increased to improved performance.

The FCN model shows relatively high training accuracy, but lower validation and test accuracies. Several factors could contribute to this performance:

- Model Architecture: FCN is specifically designed for image segmentation tasks, such as nuclei segmentation. Its architecture includes a contracting path to capture contextual information and an expanding path for precise localization.
- Overfitting: The high training accuracy and low validation/test accuracies indicate potential overfitting. Overfitting occurs when a model learns to fit the training data too closely, resulting in poor generalization to unseen data. Regularization techniques like dropout or data augmentation methods could help alleviate overfitting.

The ResNet50 model exhibits significantly higher accuracy than the CNN model, particularly in training. However, there is a notable drop in performance between training and validation/test accuracies, indicating potential overfitting. The reasons for these observations are as follows:

- Model Capacity: ResNet50 is a deep and complex architecture that can capture more intricate features and patterns compared to the CNN model. This high capacity allows it to learn a rich representation of nuclei segmentation. However, the model's capacity can also lead to overfitting, especially if the dataset is not large enough to support the model's complexity.

- Overfitting: The large difference between training and validation/test accuracies suggests that the model may have memorized the training data too well and failed to generalize to new, unseen data. Techniques like regularization (e.g., dropout) or reducing model complexity could help alleviate overfitting.

The VGG19 model demonstrates the lowest performance among the three models, with very low accuracy across all metrics. The potential reasons for this poor performance are as follows:

- Model Complexity: VGG19 is a deep and complex architecture known for its performance in image classification tasks. However, the nuclei segmentation problem may require a different model design or architecture that can better capture the specific features and patterns associated with nuclei. The complexity of VGG19 might not be suitable for this task.

V. Conclusion

In this report, I have explored various deep learning models for the multi-class nuclei segmentation problem. I experimented with different architectures, including CNN, FCN, ResNet50, and VGG19. The models were trained and evaluated on the provided dataset, with the test accuracy reported for each model.

Based on the results, the best-performing model for this task is the ResNet50. This model demonstrates the ability to accurately classify nuclei images into their respective types. Further analysis and fine-tuning of the models can be done to improve their performance.