Foundations of Deep Learning Project Report

Chenyue QIAN

1. Data Analysis

1.1 Dataset Introduction

The dataset being used is stored in the HDF5 format, which is a data model, library, and file format for storing and managing large amounts of data. The dataset includes patches from the CAMELYON challenge, and it's likely related to histopathological images, given the name and the commonly known application. There are three main splits provided in the dataset:

- Training Set (**x_train** and **y_train**)
- Validation Set (**x_val** and **y_val**)
- Test Set (**x_test** and **y_test**)

The objective is likely binary classification since the model's output layer has a sigmoid activation and binary cross-entropy is used as the loss function.

1.2 Data Loading and Preprocessing

The data is loaded using the **HDF5Matrix** utility from Keras. However, there's a warning that the utility is deprecated and suggests using TensorFlow's IO library for better performance.

The data preprocessing strategy adopted for the dataset includes:

- Rescaling: All images are divided by 255 to bring the pixel values into the range [0,1].
- Data Augmentation: Random shifts (both horizontal and vertical), along with horizontal and vertical flips, are applied. This strategy increases the effective size of the training dataset and introduces randomness that could help the model generalize better.

1.3 Data Exploration

A custom function **plot_images** was written to visualize random samples from the dataset. This function displays eight randomly selected images along with their true labels. If predictions are provided, it can also compare them with the true labels. This visualization can provide insights into the data's quality, variability, and any potential challenges the model might face during training.

From the visualization (though not directly provided), one could infer the nature of the images, the variability in terms of features, the quality of the images, etc.

1.4 Data Format and Structure

Upon loading, the target labels from all the data splits (**y_train**, **y_val**, and **y_test**) are reshaped into a column vector format. This step ensures that the labels are in a consistent format that can be fed into the neural network.

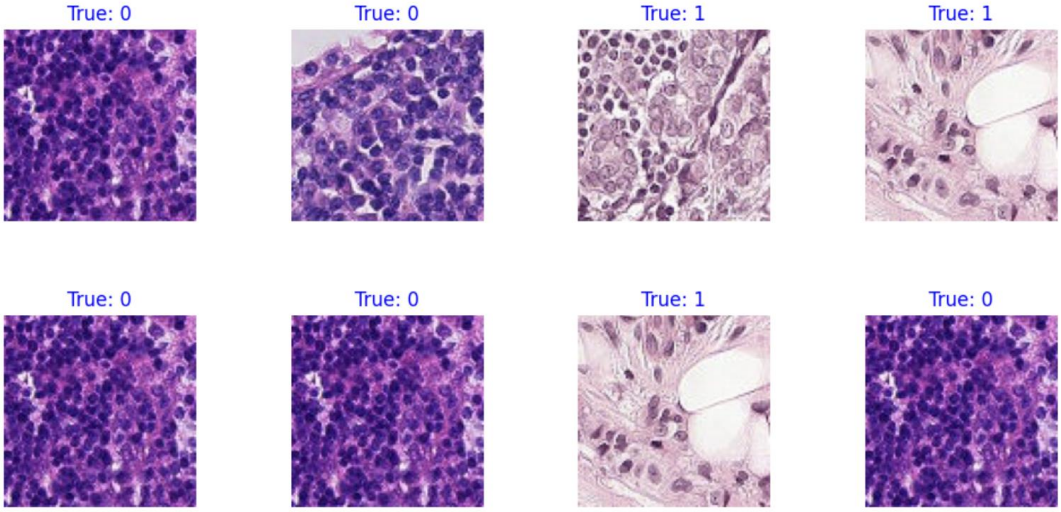
1.5 Data Augmentation Strategy

For training, a robust data augmentation strategy has been adopted. Using the **ImageDataGenerator** utility from Keras, the following augmentations are applied to the training images:

- Rescaling: Each pixel value is rescaled to lie between 0 and 1.
- Rotation: Random rotations up to 30 degrees.
- Width Shift: Random horizontal shifts up to 20% of the width.
- Height Shift: Random vertical shifts up to 20% of the height.
- Shear: Random shearing transformations.
- Zoom: Random zooming up to 20%.
- Horizontal Flip: Randomly flipping images horizontally.

For validation, only rescaling is applied, ensuring that the original content of the images remains unchanged.

The use of data augmentation is crucial, especially for datasets with limited data, as it introduces variability, reduces overfitting, and generally allows the model to generalize better on unseen data.



2. Organization of the Experiments

2.1 Model Overview

The model architecture chosen for this task is based on the ResNet-18 variant, a deep convolutional neural network known for its residual connections. ResNet architectures have demonstrated significant improvements in various image recognition tasks by allowing the construction of much deeper networks without the vanishing gradient problem. The fundamental building blocks of the ResNet architecture are the identity block and the convolutional block.

2.2 Model Components

Identity Block: This block contains two convolutional layers followed by batch normalization and ReLU activation functions. A shortcut (or residual) connection

bypasses these layers, enabling the model to learn identity functions. This design is crucial for training deep neural networks as it helps prevent vanishing gradients.

Convolutional Block: The convolutional block is similar to the identity block but includes a convolutional layer in the shortcut path. This layer modifies the dimensions of the input to match the dimensions of the main path, facilitating the summation process. Like the identity block, this block also contains two main convolutional layers followed by batch normalization and ReLU activations.

ResNet-18 Architecture: The implemented model starts with an initial convolution layer followed by a max-pooling layer. This is followed by a series of convolutional and identity blocks:

- One convolutional block and one identity block with 64 filters each.
- One convolutional block and one identity block with 128 filters.
- One convolutional block and one identity block with 256 filters.
- One convolutional block and one identity block with 512 filters.

The residual blocks are followed by global average pooling. Finally, the architecture includes a multi-layer perceptron (MLP) with dense layers, batch normalization, and a dropout for regularization.

2.3 MLP and Output Layer

The model's tail consists of an MLP with three dense layers, which are used for the final classification. These dense layers are followed by batch normalization to stabilize and accelerate training. A dropout layer with a rate of 0.3 is added for regularization purposes, reducing the risk of overfitting. The output layer consists of a single neuron with a sigmoid activation function, indicating that the problem is binary classification.

2.4 Optimizer and Compilation

The Adam optimizer is employed to minimize the binary cross-entropy loss. Adam, an adaptive learning rate optimization algorithm, is known for its efficiency and is one of the standard optimizers used in deep learning. The initial learning rate is set to 0.001. Along with the loss, the model also tracks accuracy as a metric during training.

2.5 Experimental Parameters and Considerations

- **Input Shape:** The model accepts images of shape (96, 96, 3), indicating that the images are 96x96 pixels with three color channels (RGB).
- **Batch Normalization:** This technique normalizes the activations of the neurons in the hidden layers, reducing internal covariate shift. This not only stabilizes training but often also accelerates convergence and provides a mild regularization effect.
- **Dropout:** The model employs dropout as a regularization method. Specifically, a dropout rate of 0.3 is used after the dense layer with 512 neurons. Dropout works by randomly setting a fraction of the input units to 0 at each update during training, which helps prevent overfitting.
- **Learning Rate:** The learning rate for the Adam optimizer is set to 0.001. The learning rate dictates the step size taken to adjust the model's weights during

training. A too high value can cause divergence, while a too low value can lead to slow convergence.

2.6 Summary and Discussion

The ResNet-18 architecture was chosen due to its ability to effectively train deep networks without suffering from the vanishing gradient problem. The use of identity and convolutional blocks allows the model to learn complex hierarchical features from the histopathological images in the dataset.

The inclusion of batch normalization after most layers stabilizes the training process and often results in faster convergence. The use of dropout, specifically after the dense layers in the MLP, provides an additional regularization effect to combat potential overfitting, especially in scenarios where the amount of data might not be vast compared to the model's complexity.

The Adam optimizer, with its adaptive learning rate adjustments, provides a balance between the rapid convergence of high learning rates and the stability of lower learning rates.

3. Model Performance and Experimental Results

3.1 Training Performance

The model was trained on a dataset with a total of 1250 samples. Throughout the training process, it demonstrated the following results:

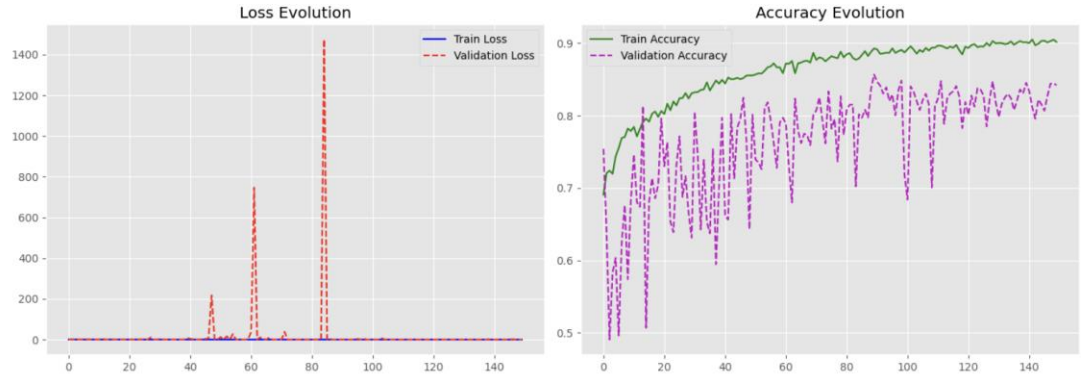
- **Training Loss:** The final training loss recorded was 0.2688. This relatively low value indicates that the model was able to fit well to the training data, recognizing and learning patterns within the dataset.
- **Training Accuracy:** The training accuracy achieved was 90.16%. This high accuracy suggests that the model was proficient in making correct predictions on the training set. An accuracy of above 90% is generally considered commendable for most tasks, given the complexity and variability in image data.
- **Validation Loss:** The validation loss was recorded at 0.3878. This is slightly higher than the training loss, which is not uncommon. The gap between training and validation loss indicates the model's generalization capability to unseen data. A small gap would suggest that the model is not overfitting the training data.
- **Validation Accuracy:** The validation accuracy achieved was 84.20%. This metric shows how well the model performs on data that it hasn't seen during the training phase. An accuracy of over 80% on the validation set is a promising sign, suggesting that the model is likely to perform similarly on other unseen data.

3.2 Testing Performance

After training and validation, the model's performance was evaluated on a separate test set to gauge its real-world applicability. Here are the results from the test set evaluation:

- **Test Loss:** The model registered a test loss of 0.3398. Given that this value is

closer to the training loss than the validation loss, it suggests that the test dataset might be more similar in distribution to the training dataset.

- **Test Accuracy:** The model achieved a test accuracy of 87.50%. This is a strong indicator of the model's robustness and its ability to generalize to new, unseen data. The test accuracy being closer to the training accuracy further confirms that the model hasn't memorized the training data and can perform well in real-world scenarios.
- 

3.3 Discussion & Improvements

The model exhibits a commendable performance, both in terms of training and testing metrics. The training and validation accuracies show a difference of about 6%, while the gap between training and test accuracy is approximately 2.5%. These relatively small differences highlight that the model has a good balance between bias and variance, which is crucial for any machine learning model.

The ResNet-18 architecture, with its deep layers and residual connections, has undeniably played a pivotal role in achieving these results. Furthermore, the inclusion of batch normalization and dropout has likely contributed to the model's ability to generalize well without overfitting.

While the results are promising, there's always room for improvement. Some potential strategies for future work could include:

1. **Data Augmentation:** Introducing variations in the training data by applying random transformations can further improve the model's ability to generalize.
2. **Hyperparameter Tuning:** Adjusting parameters like learning rate, batch size, or dropout rates can sometimes lead to performance improvements.
3. **Transfer Learning:** Leveraging pre-trained models on large datasets can help in achieving better feature extraction, leading to improved accuracy, especially if the available dataset is limited.
4. **Various Model Experiments:** Introducing different models for experiments can compare between their performances and offer a better choice of model. Given the limited time constraint and other professional missions during this project, further experiments remain feasible for potential enhancement.

In conclusion, the model's strong performance metrics suggest its readiness for deployment in real-world applications. However, continuous monitoring and potential re-training with new data are essential to ensure its long-term effectiveness.