National Tsing Hua University
Fall 2023 11210IPT 553000
Deep Learning in Biomedical Optical Imaging
Homework 4

AUTHOR

柯志明

Student ID: 111003804

1. Task A: Model Selection (20 pts)

Model Choice (5 pts)

The two pre-trained models selected from `torchvision.models` for transfer learning tasks are:

1. **ResNet50**: A variant of the Residual Network family with 50 layers, using the weights `ResNet50_Weights.IMAGENET1K_V1`.
2. **EfficientNet-B0**: The baseline model of the EfficientNet family, which provides a good trade-off between accuracy and efficiency, using weights `EfficientNet_B0_Weights.IMAGENET1K_V1`.

Explanation (15 pts)

(1) ResNet50

Architecture Complexity: ResNet50 is a medium-sized model in the ResNet family, which implements residual learning with skip connections to allow for the training of deeper neural networks without the vanishing gradient problem. The use of residual blocks makes the network easier to optimize and enables the training of deeper networks for better performance.

Pre-trained Performance: When pre-trained on large datasets like ImageNet, ResNet50 demonstrates excellent performance on a wide range of image classification tasks. Its structure enables it to learn complex features that are applicable to a variety of visual recognition tasks, which makes it highly adaptable to new datasets and problems through transfer learning.

Computation Time: While not the most lightweight architecture, ResNet50 is considered to have a moderate computational cost, providing a balance between accuracy and

computational efficiency. This makes it suitable for scenarios where both are a priority, and it can be fine-tuned relatively quickly on modern hardware.

(2) EfficientNet-B0

Architecture Complexity: EfficientNet-B0 introduces a new scaling method for convolutional networks that uniformly scales all dimensions of depth/width/resolution using a compound coefficient. This allows the model to scale up in a more structured manner. EfficientNet-B0, the baseline model, was optimized to achieve high accuracy while being efficient with respect to computational resources.

Pre-trained Performance: EfficientNet models, including the B0 variant, achieve state-of-the-art accuracy on ImageNet and other benchmarks, while using fewer parameters and FLOPs (floating-point operations per second) compared to other models of similar performance levels. This efficiency translates into better adaptability for transfer learning tasks, especially where computational resources are limited.

Computation Time: EfficientNet-B0 is designed to be efficient, hence the name. It is particularly well-suited for environments where computational resources, such as memory and processing power, are constrained. Its pre-trained performance does not come at the cost of longer training times, making it ideal for rapid prototyping and deployment.

2. Task B: Fine-tuning the ConvNet (30 pts)

Fine-tuning of ResNet50 (15 pts)

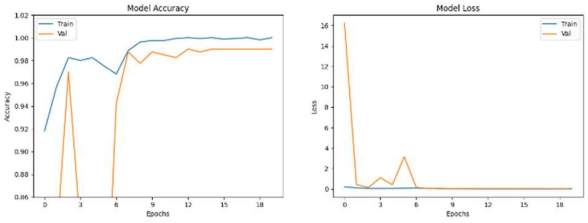

Fig. 1 presents the comparison of validation (in orange) and training (in blue) performance for the **ResNet50-based** transfer model, showcasing both the accuracy curve (a) and the loss curve (b), with **fine-tuning** applied using `ResNet50_Weights.IMAGENET1K_V1`.

During the fine-tuning of ResNet50 on the chest x-ray training dataset, both the accuracy and loss curves rapidly converged by approximately the 10th epoch. The fine-tuned model achieved 100% accuracy on the training dataset and 99% accuracy on the validation dataset.

The small gap between the training and validation losses indicates minimal overfitting. The performance on both the training and validation datasets is significantly better than that of ANNs or CNNs newly trained without transfer learning, as seen in HW2 and HW3. However, the inference accuracy on the testing dataset is only 78.25%, which is not a substantial improvement. Economically speaking, the training time for this fine-tuning process was 102.45 seconds.

Fine-tuning of EfficientNet-B0

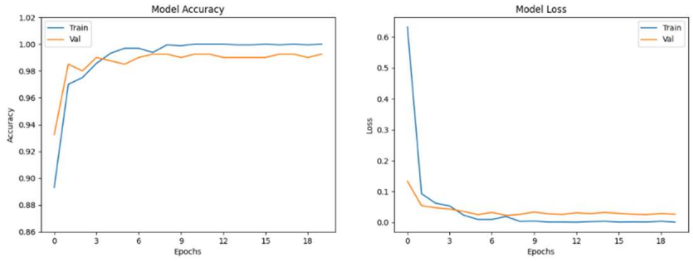

Fig. 2 presents the comparison of validation (in orange) and training (in blue) performance for the **EfficientNet-B0-based** transfer model, showcasing both the accuracy curve (a) and the loss curve (b), with **fine-tuning** applied using `EfficientNet_B0_Weights.IMAGENET1K_V1`.

Examining Figure 2, we observe more stable learning curves during the fine-tuning of EfficientNet-B0 compared to those of ResNet50. The fine-tuned model achieved 100% accuracy on the training dataset and 99.25% accuracy on the validation dataset. However, the inference accuracy on the test dataset was only 76.75%, which is lower than that of ResNet50. True to its name, the fine-tuning process for EfficientNet-B0 took only 68.18 seconds, which is significantly shorter than the time required for ResNet50.

3. Task C: ConvNet as Fixed Feature Extractor (30 pts)

ResNet50 as Fixed Feature Extractor (15 pts)

The learning curves for the ResNet50 model with a fixed feature extractor also converge rapidly, albeit with performance lower than that of fine-tuned models. The training and validation losses tend to approach the same limit value, indicating that there is likely no overfitting on the training dataset. This approach achieves a testing accuracy of 83.75%, while maintaining training and validation accuracies at around 95%. It seems to offer better generalization on the test dataset. Moreover, since the parameters in the fixed feature layers

are not trainable, the training time for this process is only 45.33 seconds, significantly less than that required for fine-tuning.

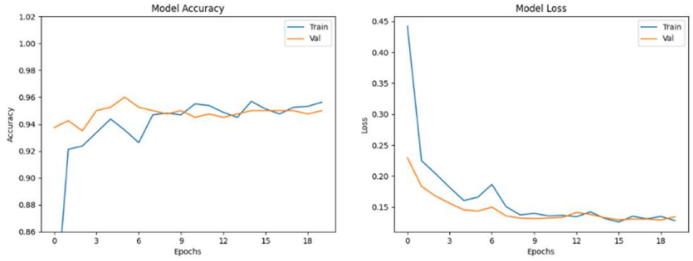

Fig. 3 compares the validation (in orange) and training (in blue) performance of the **ResNet50-based** model configured as a **fixed feature extractor**, illustrating the accuracy curve (a) and the loss curve (b) with `ResNet50_Weights.IMAGENET1K_V1`.

EfficientNet-B0 as Fixed Feature Extractor (15 pts)

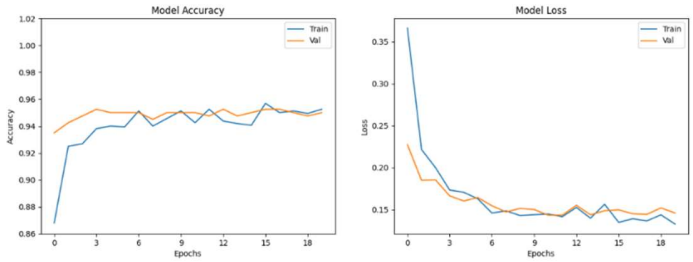

Fig. 4 compares the validation (in orange) and training (in blue) performance of the **EfficientNet-B0-based** model configured as a **fixed feature extractor**, illustrating the accuracy curve (a) and the loss curve (b) with `EfficientNet_B0_Weights.IMAGENET1K_V1`.

Similar to the previous case, the learning curves for the EfficientNet-B0 model with a fixed feature extractor converge without a considerable gap between training and validation. It achieves almost the same performance as the ResNet50 when used as a fixed feature extractor, with a testing accuracy of 83.50% and comparable training and validation accuracies at around 95%. The training time for this process is only 27.52 seconds, which is notably efficient.

4. Task D: Comparison and Analysis (10 pts)

In comparing the outcomes of Tasks B and C (see Table 1.), where ResNet50 and EfficientNet-B0 were fine-tuned versus used as fixed feature extractors, we can derive some

insightful contrasts in performance and adaptability of the models. Here's an analysis of the differences observed:

Table 1. The Accuracies and Training Time

	Pre-trained Model	Trian Acc.	Val Acc.	Test Acc.	Time (sec.)
Fine-Tuning	ResNet50	**100%**	99%	78.25%	102.45
	EfficientNet-B0	**100%**	**99.25%**	76.75%	68.18
Fixed Feature	ResNet50	95.63%	95%	**83.75%**	45.33
	EfficientNet-B0	95.25%	95%	83.5%	**27.52**

Performance Differences:

(1) **Training and Validation Accuracy:**

- Fine-tuned models (Task B) achieved near-perfect accuracies on the training dataset, indicating that they were able to learn the specific details of the training data to a high degree. This is likely due to the fact that fine-tuning allows for adjustments in all the layers of the network, which makes the model more flexible and better suited to the nuances of the new dataset.

- As fixed feature extractors (Task C), both models exhibited slightly lower training and validation accuracies compared to their fine-tuned counterparts. This is expected since the pre-trained patterns learned from the original dataset are not modified, which might not be perfectly suited for the new dataset.

(2) **Testing Accuracy:**

- There's an interesting reversal when it comes to testing accuracy. Fine-tuned models showed lower testing accuracy compared to when the models were used as fixed feature extractors. This suggests that fine-tuning might lead the model to overfit the training data, even though the validation accuracy was high, which did not generalize as well on unseen test data.

- Fixed feature extractors maintained a higher testing accuracy, which indicates better generalization. This could be due to the robust features learned from the original large dataset that are useful across different datasets.

Adaptability Differences:

(1) **Training Time:**

- Fine-tuning required significantly more time as well as more computational resources because all the weights in the network were being updated. This longer training time might not be suitable for time-constrained applications.

- Using the models as fixed feature extractors was much more time-efficient, which would be preferable in situations where computational resources or time are limited.

(2) **Transferability of Learned Features:**

- Fine-tuning might lead to learning features that are too specific to the training set, especially when the dataset is not diverse enough.

- Fixed feature extraction tends to preserve the generality of the features learned from the original dataset, which can be more easily transferred to new tasks.

5. Task E: Test Dataset Analysis (10 pts)

When training and validation accuracies are exceptionally high (close to 100%), but the testing accuracy significantly drops (to around 80%), overfitting is a primary suspect. However, if overfitting has been ruled out or minimized through regularization, data augmentation, and other techniques, we need to explore other potential causes:

a. **Data Distribution Mismatch:** There might be a mismatch in the distribution of data between training/validation and testing datasets. If the test data contain examples that are not well-represented in the training set (out-of-distribution samples), the model will likely perform poorly on those samples.

b. **Different Domains**: The model may have been trained and validated on data from one domain but tested on data from a different domain. If there's a domain shift, the model's performance can degrade.

c. **Label Noise**: Inconsistent or incorrect labeling in the test set (label noise) can also result in lower performance metrics. The model might be performing correctly according to the true label, but if the label is wrong, it will be penalized unjustly.

d. **Data Leakage**: There could be a possibility of data leakage in the training and validation datasets, leading to artificially high performance. Ensuring the test set is isolated from any data preprocessing or feature engineering steps that include knowledge of the training/validation data is crucial.