ECE 157A/272A
Homework 3: Neural Network and Foundation Model

Due Date: Friday, December 6th, 2024, 11:59PM

Start early, this is longer than Homework 2!

Introduction

In Homework 3, you will get your first experience with neural networks and foundation models. We will introduce you to the PyTorch framework, a popular deep learning acceleration framework used in both industry and academia for building both simple and cutting-edge neural networks.

For this assignment, you will working with the following datasets:

- The MIR-WM811K dataset[2], which you used in Homework 2
- The HAM10000 dataset[3]
- The CalTech 256 dataset[1]

You will be tasked with the following:

1. Implement neural networks from scratch and train them for classification.
2. Fine-tune the foundation model VGG16 for classification.
3. Analyze the performance differences between training a model from scratch and fine-tuning a foundation model for the same task. We will be revisiting feature-based training in this analysis.
4. Considering the specific classification task, the domain of the data and the properties of each dataset, determine when it is appropriate to use feature-based training, train a custom neural network from scratch, or fine-tune foundational models.

Dataset

This time, we have provided you with datasets from three different domains: the WM-811K dataset from the semiconductor wafer testing domain, the HAM10000 dataset from the medical domain, and the CalTech-256 dataset from the natural image domain.

WM-811K Dataset

The WM-811K dataset is the same dataset from the previous homework. It includes 2746 real-world wafers for training, with five failure pattern labels: `Center`, `Edge-Loc`, `Scratch`, `Donut`, and `Near-full`. The dataset attributes are summarized in Table 1.

Column	**Type**	**Description**
dieSize	float	Number of dice on a wafer
failureType	string	Type of failure pattern (5 categories: “Center,” “Edge-Loc,” “Scratch,” “Donut,” “Near-full”)
lotName	string	Manufacturing lot name of a wafer
trainTestLabel	string	Denotes whether the sample is for training or testing in the original dataset
waferIndex	integer	Denotes the wafer index within a lot
waferMap	numpy array	Contains wafer maps of varying sizes, where 0s denote no dice, 1s denote passing dice, and 2s denote failing dice

Table 1: WM-811K Dataset Attributes

HAM10000 Dataset

The HAM10000 dataset consists of 10,015 training images for detecting pigmented skin lesions. It was collected to train neural networks for the automated diagnosis of pigmented skin lesions, using dermatoscopic images acquired from various populations. This dataset exhibits a pronounced class imbalance for one of the classes, which can impact our analysis of the models. To address this concern, we have selected a subset of the data, resulting in 4,510 training images.

The dataset is provided in the `HAM10000` folder, which includes both a `train` and a `test` folder containing the training and testing images, respectively. Each of these folders has subfolders where the names indicate the classes of the images contained within. In the code, we will introduce some convenient PyTorch functions and classes for loading data organized in this manner.

The dataset consists of 7 labels: `AKIEC`, `BCC`, `BKL`, `DF`, `MEL`, `NV`, and `VASC`. An in-depth description of the datasets and labels can be found here [3].

CalTech-256 Dataset

This dataset comprises 30,607 images used for classifying 256 object categories. It was collected to train neural networks for classifying natural images of objects. The dataset includes images of object categories obtained from Google Images. Given the dataset’s numerous classes and images, to simplify the classification task and reduce training time, we have selected 10 classes from the initial set of 256 object categories. This selection resulted in 1,439 training images.

The dataset is provided in the `CalTech-256` folder, similar to the `HAM10000` folder structure, with `train` and `test` folders containing the respective images. Subfolders within these folders indicate the classes of the images.

The dataset consists of ten labels: `011.billiards`, `092.grapes`, `101.head-phones`, `105.horse`, `126.ladder`, `138.mattress`, `148.mussels`, `159.people`, `168.raccoon`, and `227.treadmill`.

Tasks 1.1 - Custom Neural Network for Wafer Map Classification

In the previous homework, we explored building traditional machine learning models by training on manually extracted features from wafer maps, utilizing our domain knowledge. In this homework, we will focus on training neural network models, specifically convolutional neural networks (CNNs), capable of taking images as input and automatically learning features from the training data to achieve the specified classification objectives.

PyTorch provides numerous modules for use, including convolutional layers, fully-connected layers, pooling layers, dropout layers, and activation functions. Most importantly, all assembled models are themselves modules, allowing you to construct your model from other models or pre-trained subsections of other models. We recommend training for at least 10 epochs to visualize the training process over time. It may be necessary to train for more than 20 epochs to achieve good convergence.

For a typical Convolutional Neural Network (CNN), the initial layers of your model should consist of convolutional layers, as they are well-suited for feature extraction from image data, such as wafer maps. Remember to include an activation function after each linear or convolutional layer to enable the model to perform more than a linear transformation. Pooling layers can help control the number of parameters at the end of the network and reduce overfitting by discarding less relevant information at each step. Additionally, the use of dropout layers can further mitigate overfitting.

For this section, **please report the following**:

- The data preprocessing steps, the data split ratio, the structure of your model (including the types of layers used and their associated parameters,) and the optimization parameters chosen for your model.
- A visualization of the model's loss and accuracy over the training process, as well as the accuracy achieved by your best model on the training, validation, and test datasets. The training accuracy is reported per batch instead of per epoch, so providing an approximate average accuracy for the epoch based on the accuracy visualization is acceptable. The test accuracy is obtained by submitting the prediction to the Gradescope auto-grader. Make sure to keep the prediction file name as specified in the code.

Tasks 1.2 - VGG16 for Wafer Map Classification

In this task, we will utilize the former ImageNet state-of-the-art model, VGG16, to classify the wafer map dataset.

VGG16 is trained on the ImageNet dataset using a supervised learning approach. ImageNet is a large-scale natural image dataset containing millions of labeled images, covering a wide range of object categories and scenes commonly found in the natural environment. VGG16 learns to recognize and classify objects and scenes in these real-world images, fine-tuning its parameters through extensive training to develop the ability to recognize a diverse array of objects and scenes.

PyTorch's torchvision package includes a pre-trained VGG16 model with `VGG16 Weights.IMAGENET1K_V1` that we will use. However, we need to modify the model to fit our dataset. The model expects 1000 classes as output, but we only have 5. To adapt the model, we will replace the last layer of the model's classifier with a new layer that has 5 outputs, as shown below:

```
model.classifier[6] = nn.Linear(in_features=4096, out_features=num_classes)
```

Note: The VGG16 model is deep, which means it may suffer from numerical instability during training. To mitigate this, choose a small learning rate for your optimizer, typically around `0.001`, and add `momentum=0.9` to help prevent any single step from being too large.

Note: VGG16 is a relatively small model but may still require substantial computational resources. Training the model may take upwards of 25 minutes, even on a GPU. Be sure to record your results.

For this section, please report the following:

- The data preprocessing steps, data split ratio, and the optimization parameters chosen for your model.
- A visualization of the model's loss and accuracy over the training process, as well as the accuracy achieved by your best model on the training, validation, and test datasets. The training accuracy is reported per batch instead of per epoch, so providing an approximate average accuracy for the epoch based on the accuracy visualization is acceptable. The test accuracy is obtained by submitting the prediction to the Gradescope auto-grader. Make sure to keep the prediction file name as specified in the code.

Tasks 1.3 - Wafer Map Classification Model Analysis

For section 1.3, please answer the following questions:

- Considering the domain of the data (semiconductor wafer testing), what are the specific challenges and advantages that make one model more suitable than the other? Hint: information that already exists or lacking in the model before training or fine-tuning.
- When accounting for trade-offs between data availability, data characteristics, the models' applicability, and your domain knowledge of wafer map patterns, which of the three methods (feature-based model from homework 2, custom neural network, foundation models) would you prefer to train a model?

Tasks 2.1-2.3 - Custom Neural Network and VGG16 for Pigmented Skin Lesion Classification

In this part of the homework, we will work with the HAM10000 dataset, which is part of the International Skin Imaging Collaboration (ISIC) dataset. It contains a large collection of skin images used for the classification of skin lesions. Your initial task is to review the images in this dataset. (Keep in mind that you will later be asked to compare how the classification in this dataset differs from the other two datasets in subsequent parts of the homework.) You will then need to modify and run the code to observe how the Convolutional Neural Network (CNN) trained from scratch compares to the fine-tuned VGG16 model for this specific dataset. (Keep in mind that you will be asked to compare the performance of the two models between the HAM10000 dataset and the other dataset.)

For the coding portion, your main objectives are as follows:

- Modify the data preprocessing functions, dataset class, and data loaders to work with the HAM10000 dataset in the provided code skeleton in this task's notebook. Please note that while the wafer dataset is loaded from a pickle file as a pandas dataframe, the HAM10000 dataset consists of images within directories. The directory structure is organized so that the name of the parent directory corresponds to the label of the images. **PyTorch provides methods to handle this dataset organization.**
- Setup the custom model and the VGG16 model to work with this dataset.

After you have obtained your best models for custom neural network and VGG16, please report the following items for each tasks: For section 2.1, please report the following:

- The data preprocessing steps, the data split ratio, the structure of your model (including the types of layers used and their associated parameters,) and the optimization parameters chosen for your model.
- A visualization of the model's loss and accuracy over the training process, as well as the accuracy achieved by your best model on the training, validation, and test datasets. The training accuracy is reported per batch instead of per epoch, so providing an approximate average accuracy for the epoch based on the accuracy visualization is acceptable. The test accuracy is obtained by submitting the prediction to the Gradescope auto-grader. Make sure to keep the prediction file name as specified in the code.

For section 2.2, please report the following:

- The data preprocessing steps, the data split ratio, and the optimization parameters chosen for your model.
- A visualization of the model's loss and accuracy over the training process, as well as the accuracy achieved by your best model on the training, validation, and test datasets. The training accuracy is reported per batch instead of per epoch, so providing an approximate average accuracy for the epoch based on the accuracy visualization is acceptable. The test accuracy is obtained by submitting the prediction to the Gradescope auto-grader. Make sure to keep the prediction file name as specified in the code.

For section 2.3, please answer the following question:

- Considering the domain of the data (pigmented skin lesions), what are the specific challenges and advantages that make one model more suitable than the other? Hint: information that already exists or lacking in the model before training or fine-tuning.

Tasks 3.1-3.3 - Custom Neural Network and VGG16 for CalTech-256

In this part of the homework, we will work with a subset of the CalTech-256 dataset, a widely recognized image dataset comprising a diverse collection of 256 object categories. This subset contains 10 specific classes from the CalTech-256 dataset. Your initial task is to review the images in this dataset. (Keep in mind that you will later be asked to compare how the classification in this dataset differs from the other two datasets in subsequent parts of the homework.) You will then need to adjust and execute the code to observe how a Convolutional Neural Network (CNN) trained from scratch performs in comparison to a fine-tuned VGG16 model when applied to this specific dataset. (Keep in mind that you will be asked to compare the performance of the two models between the HAM10000 dataset and the other dataset.)

For the coding task, your main objectives are as follows:

- Modify the data preprocessing functions, dataset class, and data loaders to be compatible with the CalTech-256 dataset in the provided code skeleton in this task's notebook. Please be aware that, unlike the HAM10000 dataset, the CalTech-256 dataset contains images of varying sizes.
- Setup the custom model and the VGG16 model to work with this dataset.

After you have obtained your best models for custom neural network and VGG16, please report the following items for each tasks: For section 3.1, please report the following:

- The data preprocessing steps, the data split ratio, the structure of your model (including the types of layers used and their associated parameters,) and the optimization parameters chosen for your model.
- A visualization of the model's loss and accuracy over the training process, as well as the accuracy achieved by your best model on the training, validation, and test datasets. The training accuracy is reported per batch instead of per epoch, so providing an approximate average accuracy for the epoch based on the accuracy visualization is acceptable. The test accuracy is obtained by submitting the prediction to the Gradescope auto-grader. Make sure to keep the prediction file name as specified in the code.

For section 3.2, please report the following:

- The data preprocessing steps, the data split ratio, and the optimization parameters chosen for your model.
- A visualization of the model's loss and accuracy over the training process, as well as the accuracy achieved by your best model on the training, validation, and test datasets. The training accuracy is reported per batch instead of per epoch, so providing an approximate average accuracy for the epoch based on the accuracy visualization is acceptable. The test accuracy is obtained by submitting the prediction to the Gradescope auto-grader. Make sure to keep the prediction file name as specified in the code.

For section 3.3, please answer the following question:

- Considering the domain of the data (natural images of objects), what are the specific challenges and advantages that make one model more suitable than the other? Hint: information that already exists or lacking in the model before training or fine-tuning.

Task 4 - Summarize Analysis

Please answer the following question:

- Is one of the three datasets "easier" to classify than the other two? Please explain your answer. Additionally, please provide your definition for "easier" in the context of this question.

What to turn in

- Code and Test Data Predictions: Submit your code in notebook format (.ipynb). Make sure to retain all outputs.
- Report: A report containing a brief overview of the task objective, an overview of what you have done, all the responses to questions and report items requested in the Tasks section. Export and submit the report as a PDF document. The report should be a separate document from the code.

References

[1] Gregory Griffin, Alex Holub, and Pietro Perona. *Caltech 256*. Apr. 2022. DOI: `10.22002/D1.20087`.

[2] Jyh-Shing R. Jang Ming-Ju Wu and Jui-Long Chen. *MIR-WM811K*. 2015. URL: `http://mirlab.org/dataset/public/`.

[3] Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. *The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions*. DOI: `10.1038/sdata.2018.161`.