1. set_seeds

```
def set_seeds(random_seed: int=42, torch_seed: int=42)
```

Description: Sets random seeds for Python, PyTorch CPU, and PyTorch GPU operations to ensure reproducibility.

Parameters:

- `random_seed` (int, optional): Seed value for Python's `random` module. Defaults to 42.
- `torch_seed` (int, optional): Seed value for PyTorch's CPU and GPU operations. Defaults to 42.

---

2. download_data

```
def download_data(source: str, destination: str, remove_source: bool = True) -> Path
```

Description: Downloads a zipped dataset from the provided source URL and unzips it into the specified destination folder.

Parameters:

- `source` (str): URL to the zipped dataset.
- `destination` (str): Folder path to unzip the dataset.
- `remove_source` (bool, optional): Whether to remove the source zip file after extraction. Defaults to True.

Returns:

- `Path` : Path to the destination directory where data is unzipped.

Example Usage:

```
download_data(source="https://example.com/dataset.zip", destination="my_dataset")
```

---

3. print_train_time

```
def print_train_time(start, end, device: torch.device = "cuda" if
torch.cuda.is_available() else "cpu")
```

Description: Prints the training duration in seconds and identifies the device (CPU or GPU).

Parameters:

- `start` (float): Start time of the training.
- `end` (float): End time of the training.
- `device` (torch.device, optional): Device used for training. Defaults to `"cuda"` if GPU is available, otherwise `"cpu"` .

Returns:

- `float` : The time difference between start and end.

---

4. accuracy_fn

```python
def accuracy_fn(y_true, y_pred)
```

Description: Calculates accuracy as a percentage between true and predicted labels.

Parameters:

- `y_true` (torch.Tensor): True labels.
- `y_pred` (torch.Tensor): Predicted labels.

Returns:

- `float`: The accuracy percentage.

---

5. plot_loss_curves

```python
def plot_loss_curves(results)
```

Description: Plots training and validation loss and accuracy curves over epochs.

Parameters:

- `results` (dict): A dictionary containing:
    - `train_loss` (list): List of training loss values.
    - `train_acc` (list): List of training accuracy values.
    - `test_loss` (list): List of validation loss values.
    - `test_acc` (list): List of validation accuracy values.

---

6. plot_confusion_matrix

```python
def plot_confusion_matrix(y_true, y_pred, class_names=None)
```

Description: Plots a confusion matrix using Seaborn heatmap.

Parameters:

- `y_true` (torch.Tensor or list): Ground truth labels.
- `y_pred` (torch.Tensor or list): Predicted labels.
- `class_names` (list, optional): List of class names for axis labels.

---

7. plot_decision_boundary

```python
def plot_decision_boundary(model, X, y, class_names, device='cuda' if
torch.cuda.is_available() else 'cpu')
```

Description: Plots the decision boundary of a model across multiple features by reducing the feature space to 2D using PCA.

Parameters:

- `model` (torch.nn.Module): The trained model.
- `X` (torch.Tensor): Input feature data.
- `y` (torch.Tensor): True labels.

- `class_names` (list): List of class names for the plot.
- `device` (str, optional): Device to run computation on. Defaults to `"cuda"` if available, otherwise `"cpu"` .

---

8. plot_pred_image

```
def plot_pred_image(model: torch.nn.Module, image_path: str, class_names: List[str] = None, transform=None, device: torch.device = "cuda" if torch.cuda.is_available() else "cpu")
```

Description: Loads an image, performs a prediction using a trained model, and displays the image with the predicted label and probability.

Parameters:

- `model` (torch.nn.Module): Trained PyTorch image classification model.
- `image_path` (str): Path to the target image.
- `class_names` (list, optional): List of class names for prediction.
- `transform` (optional): Transformations to apply to the image before prediction.
- `device` (torch.device, optional): Device to run the prediction on.

---

9. train_and_test

```
def train_and_test(model: torch.nn.Module, train_loader: torch.utils.data.DataLoader, test_loader: torch.utils.data.DataLoader, loss_fn: torch.nn.Module, optimizer: torch.optim.Optimizer, accuracy_fn, epochs: int, device: torch.device = "cuda" if torch.cuda.is_available() else "cpu")
```

Description: Trains and tests a model for a specified number of epochs, returning the training and testing loss and accuracy over each epoch.

Parameters:

- `model` (torch.nn.Module): The model to train and test.
- `train_loader` (torch.utils.data.DataLoader): DataLoader for training data.
- `test_loader` (torch.utils.data.DataLoader): DataLoader for testing data.
- `loss_fn` (torch.nn.Module): Loss function.
- `optimizer` (torch.optim.Optimizer): Optimizer for updating model weights.
- `accuracy_fn` (function): Function to calculate accuracy.
- `epochs` (int): Number of epochs to train.
- `device` (torch.device, optional): Device for computation. Defaults to `"cuda"` if available.

Returns:

- `dict` : A dictionary containing lists of training and testing loss and accuracy.

---

10. eval_model

```
def eval_model(model: torch.nn.Module, data_loader: torch.utils.data.DataLoader, loss_fn: torch.nn.Module, accuracy_fn, device: torch.device = "cuda" if torch.cuda.is_available() else "cpu")
```

Description: Evaluates a model on a test dataset and returns the loss, accuracy, and predictions.

Parameters:

- `model` (torch.nn.Module): The model to evaluate.
- `data_loader` (torch.utils.data.DataLoader): DataLoader for evaluation data.
- `loss_fn` (torch.nn.Module): Loss function.
- `accuracy_fn` (function): Function to calculate accuracy.
- `device` (torch.device, optional): Device to run the evaluation on.

Returns:

- `dict`: A dictionary containing model name, loss, accuracy, and predictions.
