Explanation of User Data Loaders (UserImageDataset & UserTextDataset)

This section explains the classes used to load user-provided image and text data for model training and evaluation.

```
class UserImageDataset(Dataset):
    def __init__(self, image_folder, label_file, transfrom=None):
        self.transform = transform
        self.image_files = [f for f in os.listdir(image_folder) if f.endswith(('.png', '.jpg', '.jpeg'))]
        self.labels = pd.read_csv(label_file) if label_file else None
        self.image_folder = image_folder
```

- __init__: Initializes the dataset. Stores the transform, collects image file names, loads labels if provided, and stores the folder path.

```
    def __len__(self):
        return len(self.image_files)
```

- __len__: Returns the number of images in the dataset.

```
    def __getitem__(self, idx):
        img_path = os.path.join(self.image_folder, self.image_files[idx])
        image = Image.open(img_path).convert('L') #Grayscale for mnist-like, or rgb for others
        if image.size != (28, 28):
            image = image.resize((28, 28))
        image = transform.ToTensor()(image)
        if image.shape[0] == 3: #converts RGB to grayscale if needed
```

```
            image = image.mean(dim=0, keepdim=True)
        if self.transform:
            image = self.transform(image)
        if self.labels is not None:
            label = int(self.labels.iloc[idx]['label'])
        else:
            label = 0
        return image, label
```

- __getitem__: Loads and processes an image, applies transforms, and returns the image and its label.

```
class UserTextDataset(Dataset):
    def __init__(self, csv_file, vocab, max_len=50):
        self.data = pd.read_csv(csv_file)
        self.vocab = vocab
        self.max_len = max_len
```

- __init__: Loads the CSV file, stores the vocabulary and maximum sequence length.

```
    def __len__(self):
        return len(self.data)
```

- __len__: Returns the number of text samples.

```
    def __getitem__(self, idx):
        text = self.vocab.lookup_indicies(self.data.iloc[idx]['text'].split()[:self.max_len])
```

```
        text = text + [0] * (self.max_len - len(text))  # Padding
        label = int(self.data.iloc[idx]['label'])
        return torch.tensor(text, dtype=torch.long), label
```

- __getitem__: Tokenizes and pads the text, returns the tensor and its label.

Purpose:

- These classes allow user-supplied image and text data to be loaded and processed in a format compatible with PyTorch models.