CISC3024 Pattern Recognition Project

by Zhang Huakang D-B9-2760-6

Outline

- 📝 **Data Loading** - loading data from file, build datasets and dataloaders
- 🎨 **Plot Example Image** - show some exmaple image of each class
- 👷 **Model Design** - *AlexNet*
- 🤹 **Model Training** - find the best lossing rate, and training the model
- 📽️ **Model Performance Evaluation** - built-in recording and camera view

Data Loading

```
train_tran = transforms.Compose([
                    transforms.RandomRotation(5),
                    transforms.RandomHorizontalFlip(0.5),
                    transforms.RandomCrop(64, padding=2),
                    transforms.ToTensor(),
                ])
tran = transforms.Compose([
                    transforms.ToTensor(),
                ])
```

```
class SatalliteDataset(Dataset):
    def __init__(self,images,labels, is_train):
        self.images_list=images
        self.labels_list=labels
        self.is_train=is_train

    def __len__(self):
        return len(self.labels_list)

    def __getitem__(self,idx):
        if self.is_train:
            return train_tran(self.images_list[idx]).to(device), self.labels_list[idx]
        return tran(self.images_list[idx]).to(device), self.labels_list[idx]
```

Data Loading

```
images,labels = shuffle(loaddata(datapath,label_names), random_state=64)

X_train, X_Test, y_train, y_Test = train_test_split(images,labels, test_size=0.23, random_state=42)

X_val, X_test, y_val, y_test = train_test_split(X_Test, y_Test, test_size=0.23, random_state=42)

train_iterator = data.DataLoader(SatalliteDataset(X_train,y_train, True), batch_size=BATCH_SIZE)

valid_iterator = data.DataLoader(SatalliteDataset(X_test,y_test, False), batch_size=BATCH_SIZE)

test_iterator = data.DataLoader(SatalliteDataset(X_val,y_val, False), batch_size=BATCH_SIZE)
```

Data Loading

```
images,labels = shuffle(loaddata(datapath,label_names), random_state=64)

X_train, X_Test, y_train, y_Test = train_test_split(images,labels, test_size=0.23, random_state=42)

X_val, X_test, y_val, y_test = train_test_split(X_Test, y_Test, test_size=0.23, random_state=42)

train_iterator = data.DataLoader(SatalliteDataset(X_train,y_train, True), batch_size=BATCH_SIZE)

valid_iterator = data.DataLoader(SatalliteDataset(X_test,y_test, False), batch_size=BATCH_SIZE)

test_iterator = data.DataLoader(SatalliteDataset(X_val,y_val, False), batch_size=BATCH_SIZE)
```

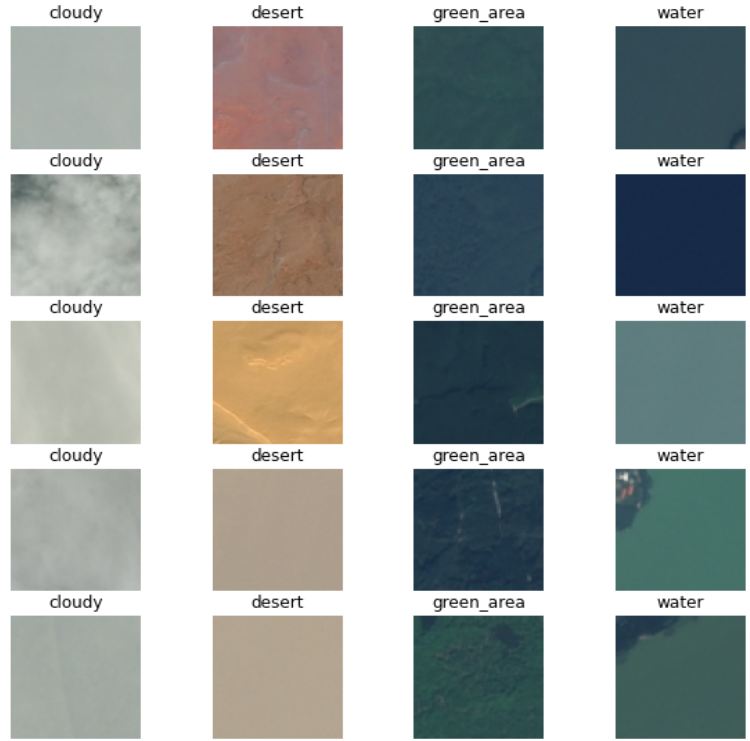
Data Loading

```
images,labels = shuffle(loaddata(datapath,label_names), random_state=64)

X_train, X_Test, y_train, y_Test = train_test_split(images,labels, test_size=0.23, random_state=42)

X_val, X_test, y_val, y_test = train_test_split(X_Test, y_Test, test_size=0.23, random_state=42)

train_iterator = data.DataLoader(SatalliteDataset(X_train,y_train, True), batch_size=BATCH_SIZE)

valid_iterator = data.DataLoader(SatalliteDataset(X_test,y_test, False), batch_size=BATCH_SIZE)

test_iterator = data.DataLoader(SatalliteDataset(X_val,y_val, False), batch_size=BATCH_SIZE)
```

Data Loading

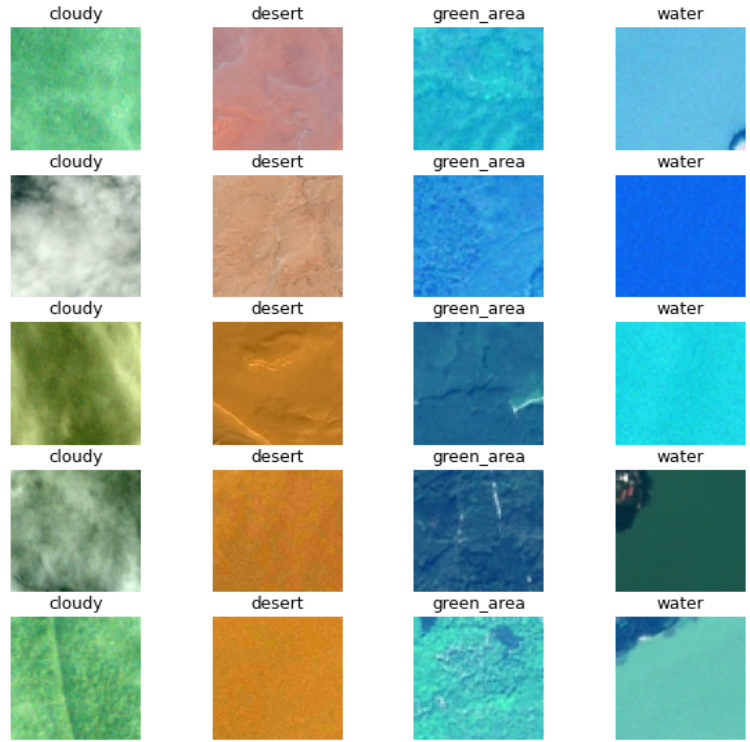
```
images,labels = shuffle(loaddata(datapath,label_names), random_state=64)

X_train, X_Test, y_train, y_Test = train_test_split(images,labels, test_size=0.23, random_state=42)

X_val, X_test, y_val, y_test = train_test_split(X_Test, y_Test, test_size=0.23, random_state=42)

train_iterator = data.DataLoader(SatalliteDataset(X_train,y_train, True), batch_size=BATCH_SIZE)

valid_iterator = data.DataLoader(SatalliteDataset(X_test,y_test, False), batch_size=BATCH_SIZE)

test_iterator = data.DataLoader(SatalliteDataset(X_val,y_val, False), batch_size=BATCH_SIZE)
```

Dataset Example



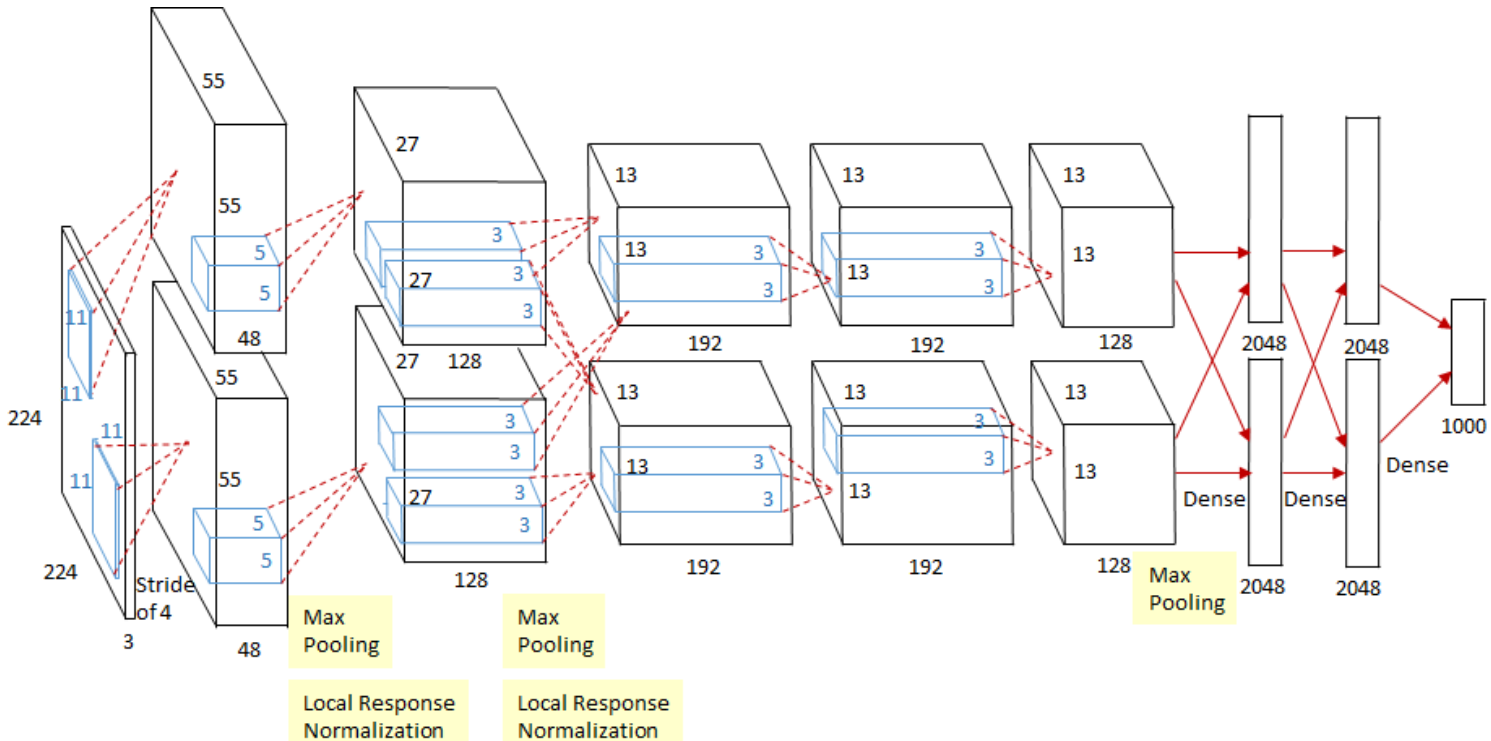
Dataset Example

with *normalization*



Model Design

AlexNet



Model Design

features

```
self.features = nn.Sequential(
    nn.Conv2d(3, 64, 3, 2, 1),
    nn.MaxPool2d(2),
    nn.ReLU(inplace=True),
    nn.Conv2d(64, 192, 3, padding=1),
    nn.MaxPool2d(2),
    nn.ReLU(inplace=True),
    nn.Conv2d(192, 384, 3, padding=1),
    nn.ReLU(inplace=True),
    nn.Conv2d(384, 256, 3, padding=1),
    nn.ReLU(inplace=True),
    nn.Conv2d(256, 256, 3, padding=1),
    nn.MaxPool2d(2),
    nn.ReLU(inplace=True)
    )
```

Model Design

classifier

```
self.classifier = nn.Sequential(
    nn.Dropout(0.5),
    nn.Linear(4096, 4096),
    nn.ReLU(inplace=True),
    nn.Dropout(0.5),
    nn.Linear(4096, 4096),
    nn.ReLU(inplace=True),
    nn.Linear(4096, output_dim),
    )
```

The model has **35,830,596** trainable parameters

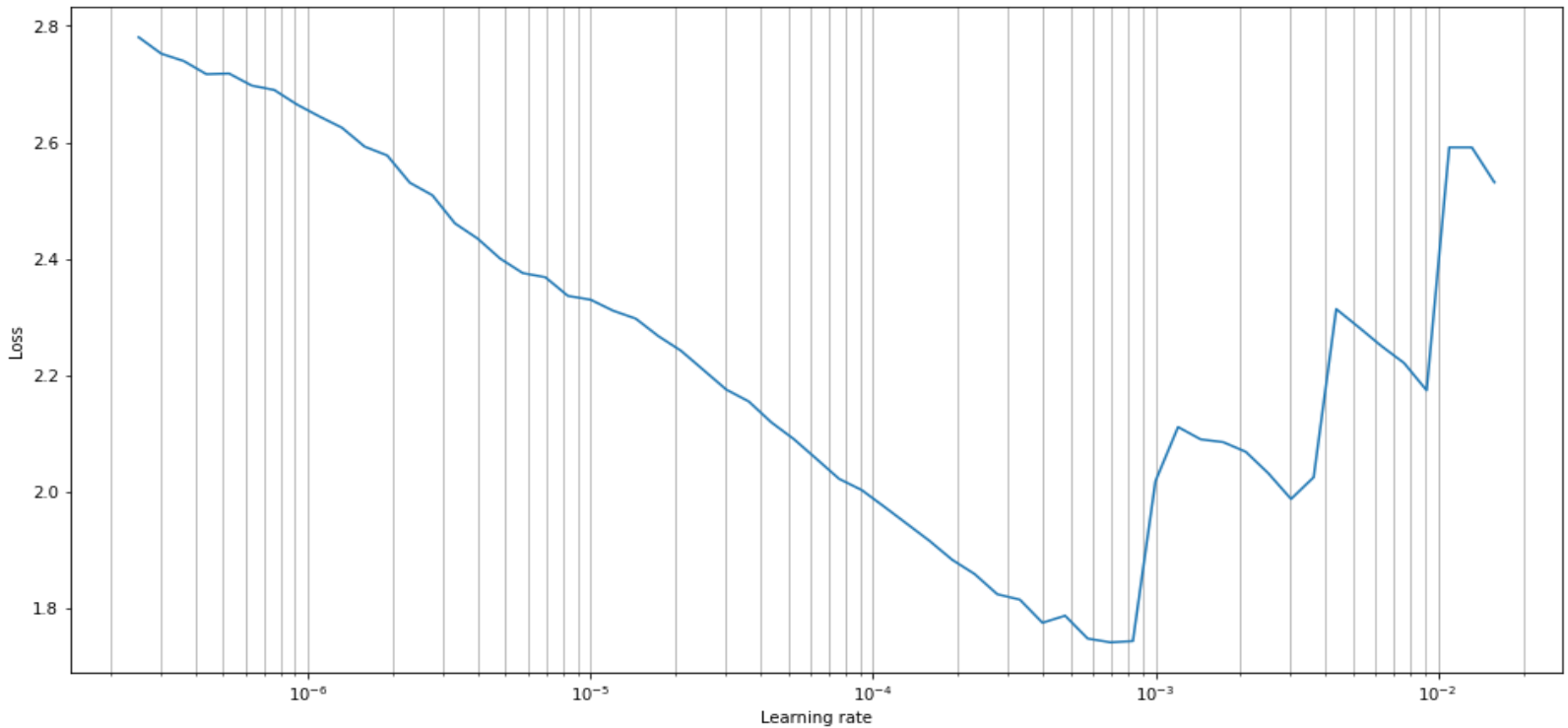
Model Training

Get the best **Learning Rate**, in `LRFinder` class

```
def range_test(self, iterator, end_lr=10, num_iter=100, smooth_f=0.05, diverge_th=5):
        lrs = []
        losses = []
        best_loss = float('inf')
        lr_scheduler = ExponentialLR(self.optimizer, end_lr, num_iter)
        iterator = IteratorWrapper(iterator)
        for iteration in range(num_iter):
            loss = self._train_batch(iterator)
            lrs.append(lr_scheduler.get_last_lr()[0])
            lr_scheduler.step()
            if iteration > 0:
                loss = smooth_f * loss + (1 - smooth_f) * losses[-1]
            if loss < best_loss:
                best_loss = loss
            losses.append(loss)
            if loss > diverge_th * best_loss:
                print("Stopping early, the loss has diverged")
                break
        model.load_state_dict(torch.load('init_params.pt'))
        return lrs, losses
```

Model Training

```
FOUND_LR = 1e-4
```

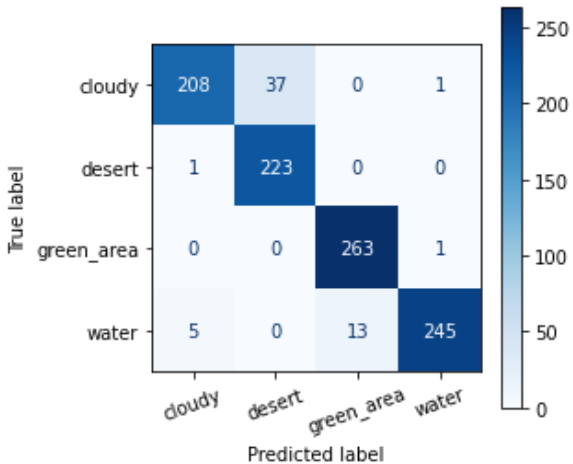


Model Training

```
for epoch in trange(EPOCHS, desc="Epochs"):
    start_time = time.monotonic()
    train_loss, train_acc = train(model, train_iterator, optimizer, criterion, device)
    valid_loss, valid_acc = evaluate(model, valid_iterator, criterion, device)
    end_time = time.monotonic()
    epoch_mins, epoch_secs = epoch_time(start_time, end_time)
    print(f'Epoch: {epoch+1:02} | Epoch Time: {epoch_mins}m {epoch_secs}s')
    print(f'\tTrain Loss: {train_loss:.3f} | Train Acc: {train_acc*100:.2f}%')
    print(f'\t Val. Loss: {valid_loss:.3f} |  Val. Acc: {valid_acc*100:.2f}%')
```

Model Performance Evaluation

```
Epoch: 25 | Epoch Time: 0m 5s
    Train Loss: 0.116 | Train Acc: 95.74%
     Val. Loss: 0.193 |  Val. Acc: 92.22%
Test Loss: 0.161 | Test Acc: 93.81%
```



Q&A