Hands-on 9

CIE5141 Computer Vision in Construction

b07501113 林庭瑄

A. Image Classification

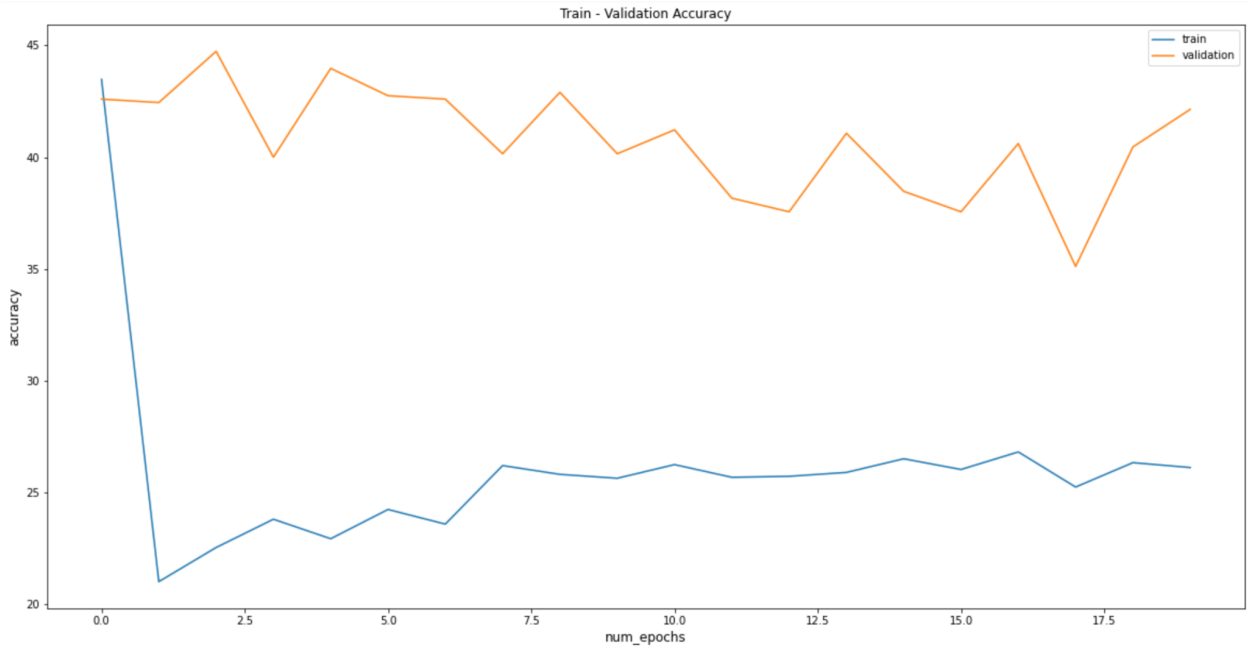
1. Report your results (50%)

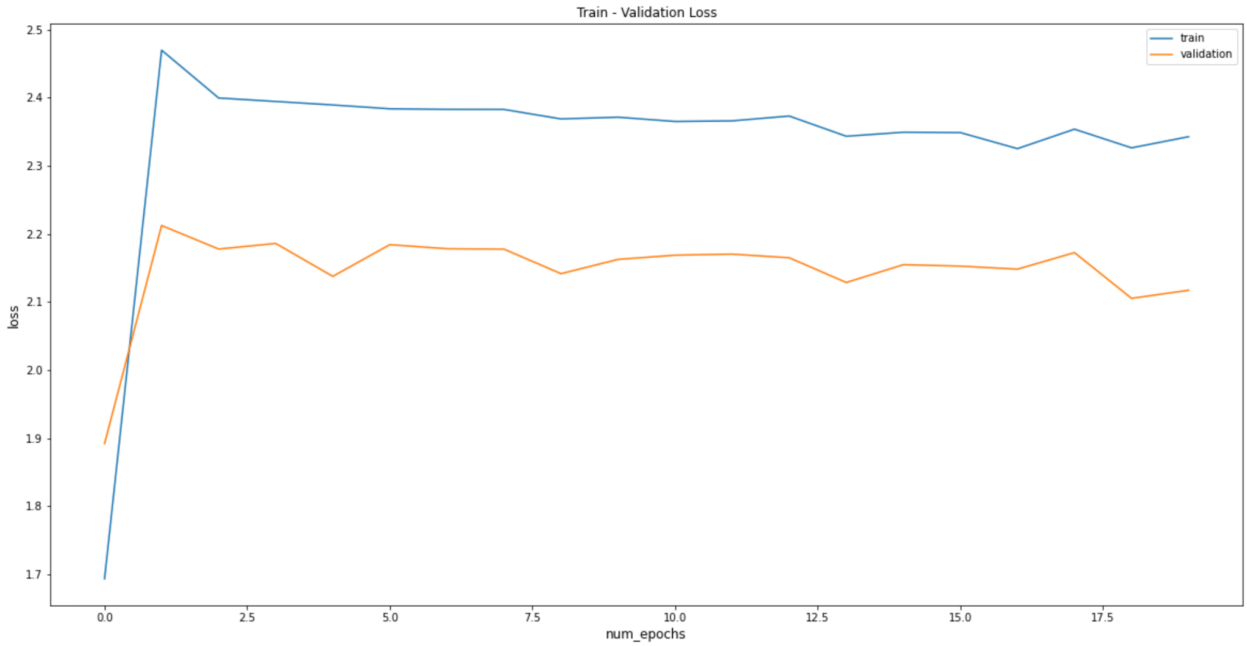
(i) Report your accuracy

```
correct=0
total=0
for batch_idx, (data_, target_) in enumerate(test_loader):
 data_, target_ = data_.to(device), target_.to(device)# on GPU
 outputs = model(data_)
 _,pred = torch.max(outputs, dim=1)
 correct += torch.sum(pred==target_).item()
 total += target_.size(0)
print("Accuracy:",correct/total*100)
```

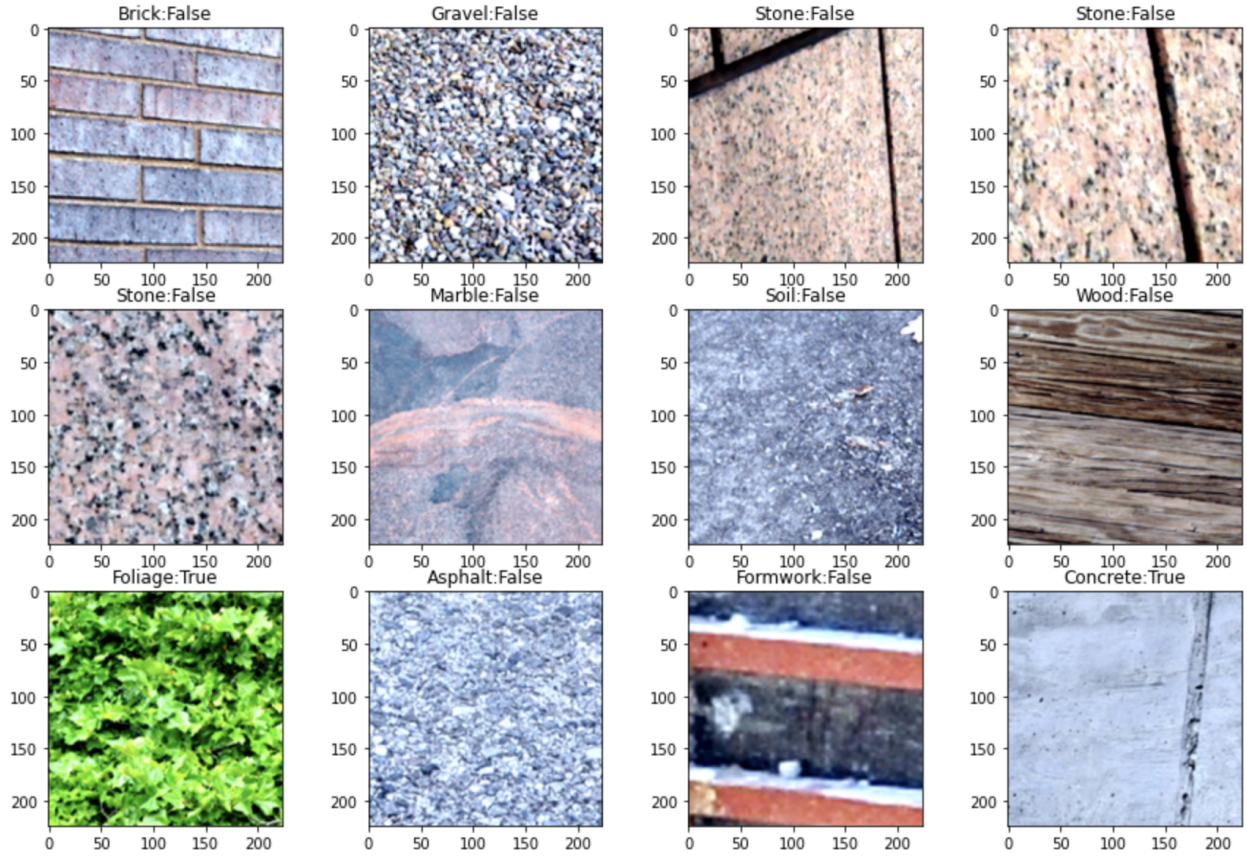
```
Accuracy: 45.25993883792049
```

(ii) Paste here the accuracy and loss curves





(iii) Paste here some prediction results



(iv) Mentions the steps taken to improve the accuracy

Data:

70% train
20% validate
10% test

Data normalization & Data augmentation.

Normalize : train&validation&test
RandomCrop: train
Gaussian Blur: train
Random Rotation: train

```
train_transform = transforms.Compose(
   [transforms.RandomCrop(224),
    transforms.RandomHorizontalFlip(),
                transforms.RandomRotation(30,    resample=Image.BICUBIC,
expand=False),
    transforms.GaussianBlur(7,3),
    transforms.ToTensor(),
         transforms.Normalize((0.485,  0.456,  0.406),  (0.229,  0.224,
0.225))])
val_transform = transforms.Compose(
   [transforms.ToTensor(),
    transforms.CenterCrop(224),
         transforms.Normalize((0.485,  0.456,  0.406),  (0.229,  0.224,
0.225))])
test_transform = transforms.Compose(
   [transforms.ToTensor(),
    transforms.CenterCrop(224),
         transforms.Normalize((0.485,  0.456,  0.406),  (0.229,  0.224,
0.225))])
```

Deeper network &.Normalization layers

Add several layers

```
class Net(nn.Module):
   def __init__(self):
       super(Net, self).__init__()
           # 3 input image channel, 16 output channels, 3x3 square
convolution kernel
       self.conv1 = nn.Conv2d(3,16,kernel_size=3,stride=2,padding=1)
       self.conv2 = nn.Conv2d(16, 32,kernel_size=3,stride=2, padding=1)
       self.conv3 = nn.Conv2d(32, 64,kernel_size=3,stride=2, padding=1)
```

```
            self.conv4  =  nn.Conv2d(64,  128,kernel_size=3,stride=2,
padding=1)
            self.conv5  =  nn.Conv2d(128,  256,kernel_size=3,stride=2,
padding=1)
            self.conv6  =  nn.Conv2d(256,  512,kernel_size=3,stride=2,
padding=1)
            self.conv7  =  nn.Conv2d(512,  512,kernel_size=3,stride=2,
padding=1)
        self.pool = nn.MaxPool2d(2, 2)
        self.dropout = nn.Dropout2d(0.4)
        self.batchnorm1 = nn.BatchNorm2d(16)
        self.batchnorm2 = nn.BatchNorm2d(32)
        self.batchnorm3 = nn.BatchNorm2d(64)
        self.batchnorm4 = nn.BatchNorm2d(128)
        self.batchnorm5 = nn.BatchNorm2d(256)
        self.batchnorm6 = nn.BatchNorm2d(512)
        self.fc1 = nn.Linear(64*8,512 )
        self.fc2 = nn.Linear(512, 256)
        self.fc3 = nn.Linear(256, 128)
        self.fc4 = nn.Linear(128, 64)
        self.fc5 = nn.Linear(64, 32)
        self.fc6 = nn.Linear(32, 13)

    def forward(self, x):
        x = self.batchnorm1(F.relu(self.conv1(x)))
        x = self.batchnorm2(F.relu(self.conv2(x)))
        x = self.batchnorm3(self.pool(F.relu(self.conv3(x))))
        x = self.dropout(self.batchnorm3(self.pool(x)))
        x = self.batchnorm4(F.relu(self.conv4(x)))
        x = self.batchnorm5(F.relu(self.conv5(x)))
        x = self.batchnorm6(F.relu(self.conv6(x)))
        x = self.dropout(self.conv7(x))
        x = x.view(-1, 64*8) # Flatten layer
        x = self.dropout(self.fc1(x))
        x = self.dropout(self.fc2(x))
        x = self.dropout(self.fc3(x))
        x = self.dropout(self.fc4(x))
        x = self.dropout(self.fc5(x))
        x = F.log_softmax(self.fc6(x),dim = 1)
        return x
```

Early stopping.

set the epoch to 20 as the validation accuracy starts to decrease.