附录

网络结构

图像增强V1

```
class ConvNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(6, 9, 5, padding=2) # 28x28
        self.pool1 = nn.MaxPool2d(2, 2) # 14x14
        self.conv2 = nn.Conv2d(12, 21, 5) # 10x10
        self.pool2 = nn.MaxPool2d(2, 2) # 5x5
        self.conv3 = nn.Conv2d(24, 120, 5)
        self.fc1 = nn.Linear(864, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        in_size = x.size(0)
        old = x
        out = torch.cat([x, old], dim=1)

        out = self.conv1(out) # 24
        out = F.relu(out)
        out = self.pool1(out)  # 12

        transform = transforms.Resize((out.shape[2], out.shape[2]))
        old = transform(x)
        out = torch.cat([out, old], dim=1)

        out = self.conv2(out) # 10
        out = F.relu(out)
        out = self.pool2(out)

        transform = transforms.Resize((out.shape[2], out.shape[2]))
        old = transform(x)
        out = torch.cat([out, old], dim=1)
        residual3 = self.conv3(out)

        out = out.view(in_size, -1)
        out = self.fc1(out)
        out = F.relu(out)
        out = self.fc2(out)
        out = F.log_softmax(out, dim=1)
        return out
```

图像增强V2

```
class ConvNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(6, 9, 5, padding=2) # 28x28
        self.pool1 = nn.MaxPool2d(2, 2) # 14x14
        self.conv2 = nn.Conv2d(12, 21, 5) # 10x10
        self.pool2 = nn.MaxPool2d(2, 2) # 5x5
        self.conv3 = nn.Conv2d(24, 120, 5)
        self.conv11 = nn.Conv2d(6, 24, 5)
        self.conv12 = nn.Conv2d(6, 120, 5)
        self.conv21 = nn.Conv2d(12, 120, 5)
        self.fc1 = nn.Linear(9504, 1024)
        self.fc2 = nn.Linear(1024, 10)

    def forward(self, x):
        in_size = x.size(0)
        old = x
        out = torch.cat([x, old], dim=1)
        out12 = self.conv12(out)
        out = self.conv1(out) # 24
        out = F.relu(out)
        out = self.pool1(out)  # 12

        transform = transforms.Resize((out.shape[2], out.shape[2]))
        old = transform(x)
        out = torch.cat([out, old], dim=1)
        out21 = self.conv21(out)

        out = self.conv2(out) # 10
        out = F.relu(out)
        out = self.pool2(out)

        transform = transforms.Resize((out.shape[2], out.shape[2]))
        old = transform(x)
        out12 = transform(out12)
        out21 = transform(out21)
        out = torch.cat([out, old], dim=1)
        residual3 = self.conv3(out)

        out = torch.cat([out, out12, out21], dim=1)
        out = out.view(in_size, -1)
        out = self.fc1(out)
        out = F.relu(out)
        out = self.fc2(out)
        out = F.log_softmax(out, dim=1)
        return out
```

ViT+lenet-5

```
# 定义Vision Transformer模型
class VisionTransformer(nn.Module):
    def __init__(self, num_classes=10, image_size=32, patch_size=4,
hidden_dim=128, num_heads=8, num_layers=1):
        super(VisionTransformer, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5, padding=2) # 28x28
        self.pool1 = nn.MaxPool2d(2, 2) # 14x14
        self.conv2 = nn.Conv2d(6, 16, 5, padding=2) # 10x10
        self.pool2 = nn.MaxPool2d(2, 2) # 5x5
        self.conv3 = nn.Conv2d(16, 128, 5)

        # self.embedding = nn.Conv2d(3, hidden_dim, kernel_size=patch_size,
stride=patch_size)
        self.transformer =
nn.TransformerEncoder(nn.TransformerEncoderLayer(d_model=hidden_dim,
nhead=num_heads), num_layers)
        self.fc = nn.Linear(2048, 1024)
        self.fc1 = nn.Linear(1024,128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        out = self.conv1(x) #24
        out = F.relu(out)
        out = self.pool1(out)  #12
        out = self.conv2(out) #10
        out = F.relu(out)
        out = self.pool2(out)
        out = self.conv3(out)

        # out = F.interpolate(out, size=(32, 32), mode='bilinear',
align_corners=False)
        # out = self.embedding(x)
        out = out.flatten(2).permute(2, 0, 1)
        out = self.transformer(out)
        out = out.permute(1, 0, 2).flatten(1)

        out = self.fc(out)
        out = self.fc1(out)
        out = self.fc2(out)
        return out
```

ViT+resnet18

具体见代码。

```
class ViT_ResNet18(nn.Module):
    def __init__(self, num_classes=10):
        super(ViT_ResNet18, self).__init__()

        # 创建ViT模型
        self.vit = VisionTransformer(num_classes=num_classes)
```

```
        # 创建resnet模型
        self.resnet = ResNet18()

    def forward(self, x):
        # 使用resnet模型处理ViT的输出
        x = self.resnet(x)
        # x = F.interpolate(x, size=(32, 32), mode='bilinear',
align_corners=False)
        # 使用ViT模型处理输入
        x = self.vit(x)
        return x
```

ViT+lenet-5大核

```
# 定义Vision Transformer模型
class VisionTransformer(nn.Module):
    def __init__(self, num_classes=10, image_size=32, patch_size=4,
hidden_dim=128, num_heads=8, num_layers=1):
        super(VisionTransformer, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 11, padding=5) # 28x28
        self.pool1 = nn.MaxPool2d(2, 2) # 14x14
        self.conv2 = nn.Conv2d(6, 16, 7, padding=3) # 10x10
        self.pool2 = nn.MaxPool2d(2, 2) # 5x5
        self.conv3 = nn.Conv2d(16, 128, 7,padding=1)

        # self.embedding = nn.Conv2d(3, hidden_dim, kernel_size=patch_size,
stride=patch_size)
        self.transformer =
nn.TransformerEncoder(nn.TransformerEncoderLayer(d_model=hidden_dim,
nhead=num_heads), num_layers)
        self.fc = nn.Linear(2048, 1024)
        self.fc1 = nn.Linear(1024,128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        out = self.conv1(x) #24
        out = F.relu(out)
        out = self.pool1(out)  #12
        out = self.conv2(out) #10
        out = F.relu(out)
        out = self.pool2(out)
        out = self.conv3(out)

        # out = F.interpolate(out, size=(32, 32), mode='bilinear',
align_corners=False)
        # out = self.embedding(x)
        out = out.flatten(2).permute(2, 0, 1)
        out = self.transformer(out)
        out = out.permute(1, 0, 2).flatten(1)

        out = self.fc(out)
        out = self.fc1(out)
        out = self.fc2(out)
        return out
```

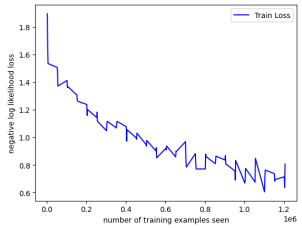
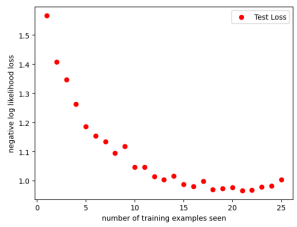
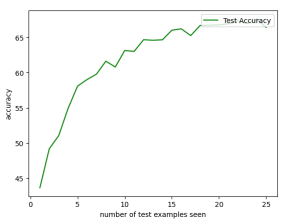
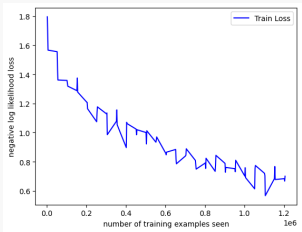
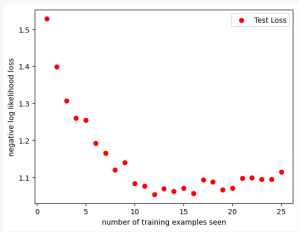
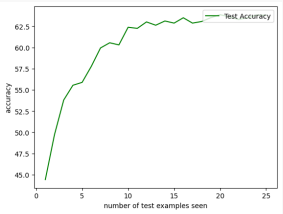
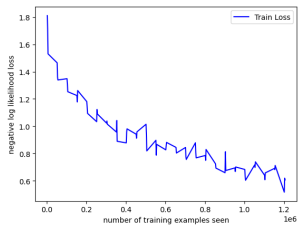
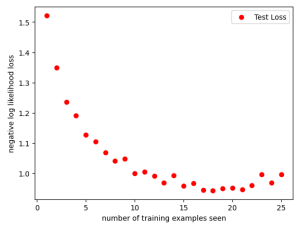
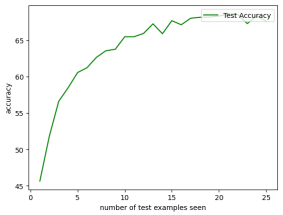
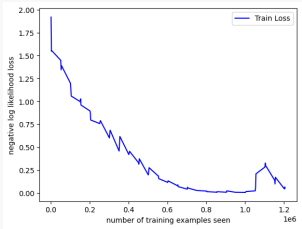
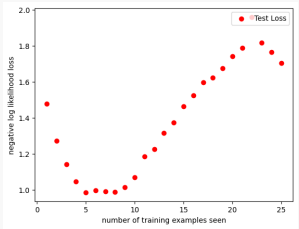
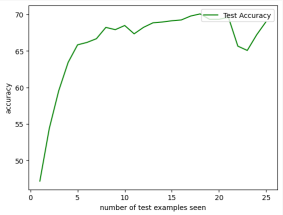
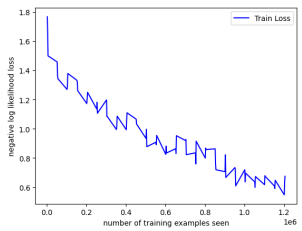
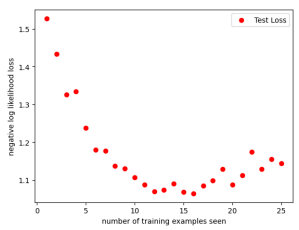
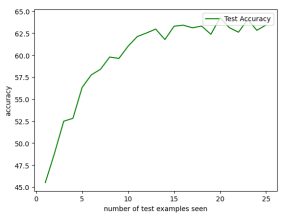
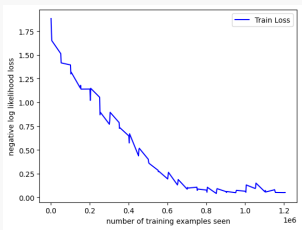
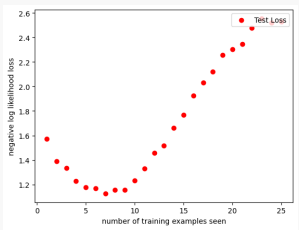
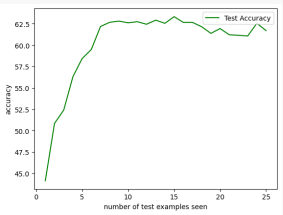
重要网络结构见上，其他具体网络结构见代码即可

代码链接：

```
https://github.com/ylwango613/Network_DLreport
```

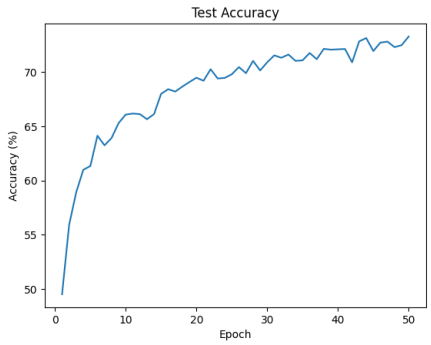
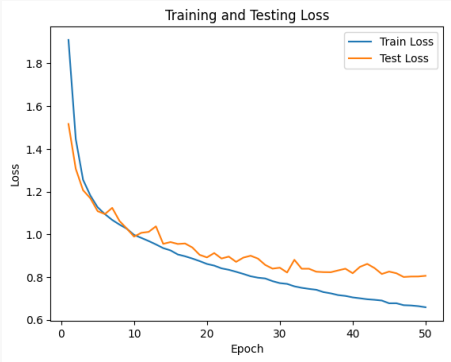
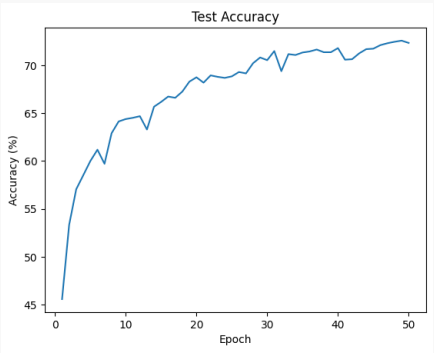
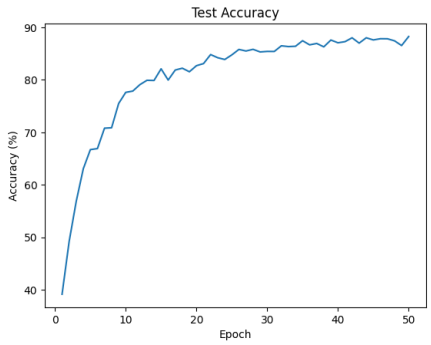
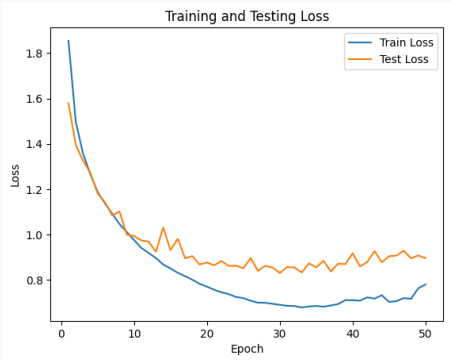
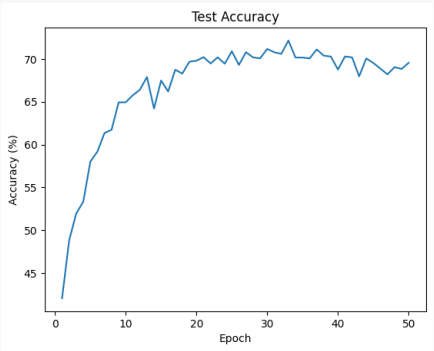
实验结果图

原图RGB增强

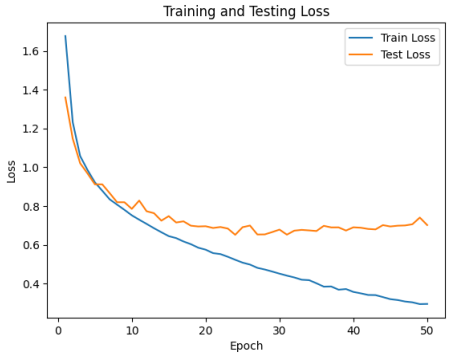
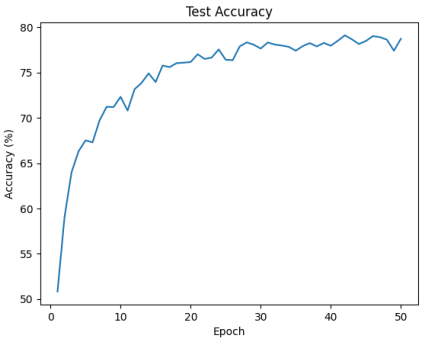
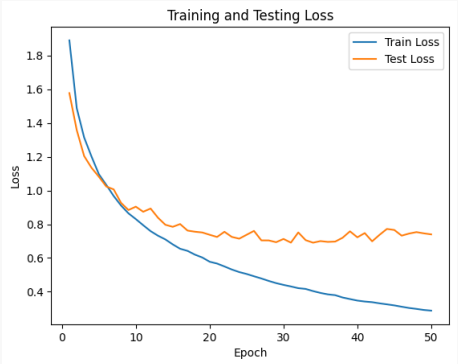
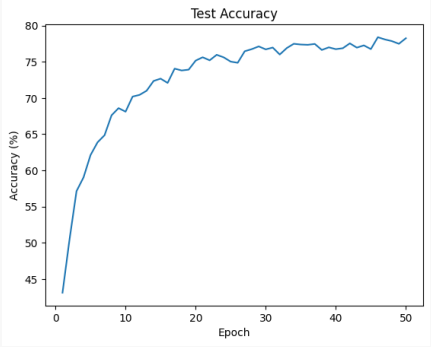
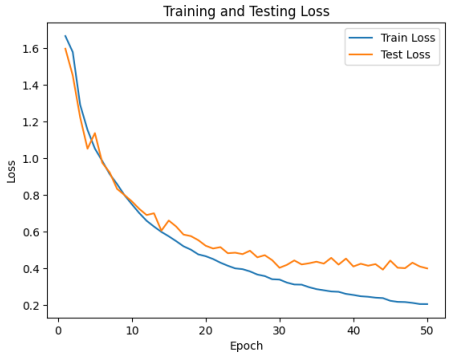
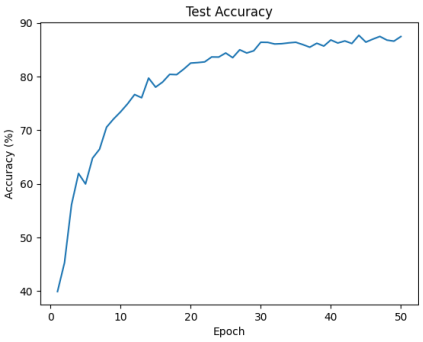
模型	训练集loss	loss变化	准确度变化
LeNet			
大核 LeNet			
V1增强 LeNet			
V2增强 LeNet			
V1增强 大核 LeNet			
V2增强 大核 LeNet			

卷积特征提取

一层transformer

模型	loss变化	准确度变化
无卷积		
LeNet5辅助卷积		
ResNet-18辅助卷积		
LeNet5大核辅助卷积		

两层transformer

模型	loss变化	准确度变化
无卷积		
LeNet5-5辅助卷积		
ResNet-18辅助卷积		
LeNet5大核辅助卷积