CV

1. **MNIST classification**

**a)**

learning rate: Determines the speed of weight update; Setting too large will make the result exceed the optimal value, too small will make the falling speed too slow.

weight decay: omit overfitting; To adjust the influence of model complexity on the loss function, large weight decay means the value of the complex model loss function is large.

**b)**

Correct the mistakes in CNN2 and train it on MNIST train set.

self.conv2 = nn.Conv2d(in\_channels=32, out\_channels=64, kernel\_size=5, stride=1, padding=2)# in\_channels should be the same number of the last layer’s out\_channels and kernel\_size means 5x5;

self.fc1 = nn.Linear(in\_features=3136, out\_features=256)#in\_features=64\*7\*7

**c)**

﻿train\_data = datasets.FashionMNIST('./fashionmnist\_data/', train=True, download=True,

transform=transforms.Compose([transforms.ToTensor(),transforms.Normalize((0.1307,), (0.3081,))]))

**d)**

Design CNN3 with additional regularization

add the dropout layer in the end of ﻿fc layers: self.drop = nn.Dropout(p=0.5)

the accuracy is 87% ( before CNN was 84%)

**e)**

compute valid convolution of w and X

with stride=1 and bias=1, the result is attached by [zyuan@tcd.ie\_out](mailto:zyuan@tcd.ie_out).csv

**f)**

Calculate the number of parameters and the number of FLOPS of this network

**2)**

**a)**

particular network depth (DnCNN-3: 20) to capture enough spatial information for denoising; a single DnCNN-3 model is trained for three general image denoising tasks, including blind Gaussian denoising, SISR and JPEG image deblocking.

**b)**

test.py (σ=25, 51) with the same model\_000

PSNR and SSIM: 29.26dB, 0.9022(σ=25); 16.67dB, 0.3328(σ=51)

**c)**

train.py(epoch=1, lr=0.0001, σ=51)

test(σ=51,model\_001): PSRN/SSIM: 26.19dB, 0.8248

**d)**

test.py-xxx.png(model\_000,model\_001) submit image and PSRN/SSIM:

15.72dB, 0.6943(model\_000); 17.69dB,0.6873(model\_001)

**e)**

y(noisy image)=x(clean image)+v(noise)

std=43.189632 mean=126.959图片包含 物体

描述已自动生成

**3）Semantic segmentation**

**a)**

Convnets are built on translation invariance. Their basic components (convolution, pooling, and activation functions) operate on local input regions, and depend only on relative spatial coordinates.

By discarding the final average pooling layer, discarding the final classifier layer, and convert all fully connected layers to convolutions.

**b)**

zyuan@tcd.ie\_predicted.png

**c)**

**﻿** [zyuan@tcd.ie\_iou.csv](mailto:zyuan@tcd.ie_iou.csv) IOU: 0.6843470483005367