实验七 matlab中深度学习操作

1. 实验目的
2. 熟悉深度学习、卷积和分类回归任务等数学原理
3. 了解matlab实现深度学习模型搭建、训练及数据集验证等操作方法
4. 编写并尝试matlab中的图像分类、图像回归等算法
5. 实验平台

MATLAB 2022b

Image Processing Toolbox、Deep Learning Toolbox、Statistics and Machine Learning Toolbox库需自行安装

三、实验步骤

1. 创建简单的深度学习神经网络以用于分类

此示例说明如何创建和训练简单的卷积神经网络来进行深度学习分类。卷积神经网络是深度学习的基本工具，尤其适用于图像识别。

该示例演示如何：

加载和浏览图像数据。

定义神经网络架构。

指定训练选项。

训练神经网络。

预测新数据的标签并计算分类准确度。

1. 加载和浏览图像数据

加载数字样本数据作为图像数据存储。imageDatastore 根据文件夹名称自动标注图像，并将数据存储为 ImageDatastore 对象。通过图像数据存储可以存储大图像数据，包括无法放入内存的数据，并在卷积神经网络的训练过程中高效分批读取图像。

digitDatasetPath = fullfile(matlabroot,'toolbox','nnet','nndemos', ...

'nndatasets','DigitDataset');

imds = imageDatastore(digitDatasetPath, ...

'IncludeSubfolders',true,'LabelSource','foldernames');

显示数据存储中的部分图像。

figure;

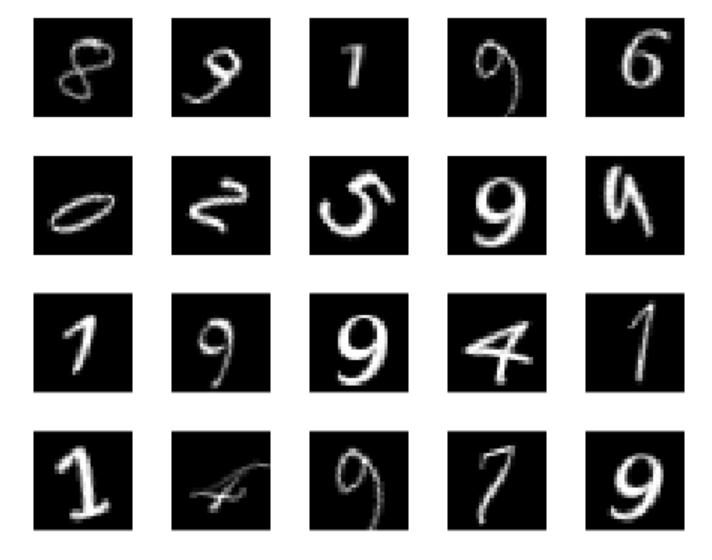
perm = randperm(10000,20);

for i = 1:20

subplot(4,5,i);

imshow(imds.Files{perm(i)});

end



计算每个类别中的图像数量。labelCount 是一个表，其中列出了标签，以及每个标签对应的图像数量。数据存储包含数字 0-9 的总共 10000 个图像，每个数字对应 1000 个图像。您可以在神经网络的最后一个全连接层中指定类数作为 OutputSize 参数。

labelCount = countEachLabel(imds)

必须在神经网络的输入层中指定图像的大小。检查 digitData 中第一个图像的大小。每个图像的大小均为 28×28×1 像素。

img = readimage(imds,1);

size(img)

1. 指定训练集和验证集

将数据划分为训练数据集和验证数据集，以使训练集中的每个类别包含 750 个图像，并且验证集包含对应每个标签的其余图像。splitEachLabel 将数据存储 digitData 拆分为两个新的数据存储 trainDigitData 和 valDigitData。

numTrainFiles = 750;

[imdsTrain,imdsValidation] = splitEachLabel(imds,numTrainFiles,'randomize');

1. 定义神经网络架构

layers = [

imageInputLayer([28 28 1])

convolution2dLayer(3,8,'Padding','same')

batchNormalizationLayer

reluLayer

maxPooling2dLayer(2,'Stride',2)

convolution2dLayer(3,16,'Padding','same')

batchNormalizationLayer

reluLayer

maxPooling2dLayer(2,'Stride',2)

convolution2dLayer(3,32,'Padding','same')

batchNormalizationLayer

reluLayer

fullyConnectedLayer(10)

softmaxLayer

classificationLayer];

图像输入层 imageInputLayer 用于指定图像大小，在本例中为 28×28×1。这些数字对应于高度、宽度和通道大小。数字数据由灰度图像组成，因此通道大小（颜色通道）为 1。对于彩色图像，通道大小为 3，对应于 RGB 值。您不需要打乱数据，因为 trainNetwork 默认会在训练开始时打乱数据。trainNetwork 还可以在训练过程中的每轮训练开始时自动打乱数据。

卷积层在卷积层中，第一个参数是 filterSize，它是训练函数在沿图像扫描时使用的滤波器的高度和宽度。在此示例中，数字 3 表示滤波器大小为 3×3。您可以为滤波器的高度和宽度指定不同大小。第二个参数是滤波器数量 numFilters，它是连接到同一输入区域的神经元数量。此参数决定了特征图的数量。使用 'Padding' 名称-值对组对输入特征图进行填充。对于默认步幅为 1 的卷积层，'same' 填充可确保空间输出大小与输入大小相同。您也可以使用 convolution2dLayer 名称-值对组参数定义该层的步幅和学习率。

批量归一化层批量归一化层对神经网络中的激活值和梯度传播进行归一化，使神经网络训练成为更简单的优化问题。在卷积层和非线性部分（例如 ReLU 层）之间使用批量归一化层，来加速神经网络训练并降低对神经网络初始化的敏感度。使用 batchNormalizationLayer 创建批量归一化层。

ReLU 层批量归一化层后接一个非线性激活函数。最常见的激活函数是修正线性单元 (ReLU)。使用 reluLayer 创建 ReLU 层。

最大池化层卷积层（带激活函数）有时会后跟下采样操作，以减小特征图的空间大小并删除冗余空间信息。通过下采样可以增加更深卷积层中的滤波器数量，而不会增加每层所需的计算量。下采样的一种方法是使用最大池化，可使用 maxPooling2dLayer 创建。最大池化层返回由第一个参数 poolSize 指定的矩形输入区域的最大值。在此示例中，该矩形区域的大小是 [2,2]。'Stride' 名称-值对组参数指定训练函数在沿输入扫描时所采用的步长。

全连接层卷积层和下采样层后跟一个或多个全连接层。顾名思义，全连接层中的神经元将连接到前一层中的所有神经元。该层将先前层在图像中学习的所有特征组合在一起，以识别较大的模式。最后一个全连接层将特征组合在一起来对图像进行分类。因此，最后一个全连接层中的 OutputSize 参数等于目标数据中的类数。在此示例中，输出大小为 10，对应于 10 个类。使用 fullyConnectedLayer 创建全连接层。

softmax 层 softmax 激活函数对全连接层的输出进行归一化。softmax 层的输出由总和为 1 的多个正数组成，这些数字随后可被分类层用作分类概率。使用 softmaxLayer 函数在最后一个全连接层后创建一个 softmax 层。

分类层最终层是分类层。该层使用 softmax 激活函数针对每个输入返回的概率，将输入分配到其中一个互斥类并计算损失。要创建分类层，请使用 classificationLayer。

1. 指定训练选项

定义神经网络结构后，指定训练选项。使用具有动量的随机梯度下降 (SGDM) 训练神经网络，初始学习率为 0.01。将最大训练轮数设置为 4。一轮训练是对整个训练数据集的一个完整训练周期。通过指定验证数据和验证频率，监控训练过程中的神经网络准确度。每轮训练都会打乱数据。软件基于训练数据训练神经网络，并在训练过程中按固定时间间隔计算基于验证数据的准确度。验证数据不用于更新神经网络权重。打开训练进度图，关闭命令行窗口输出。

options = trainingOptions('sgdm', ...

'InitialLearnRate',0.01, ...

'MaxEpochs',4, ...

'Shuffle','every-epoch', ...

'ValidationData',imdsValidation, ...

'ValidationFrequency',30, ...

'Verbose',false, ...

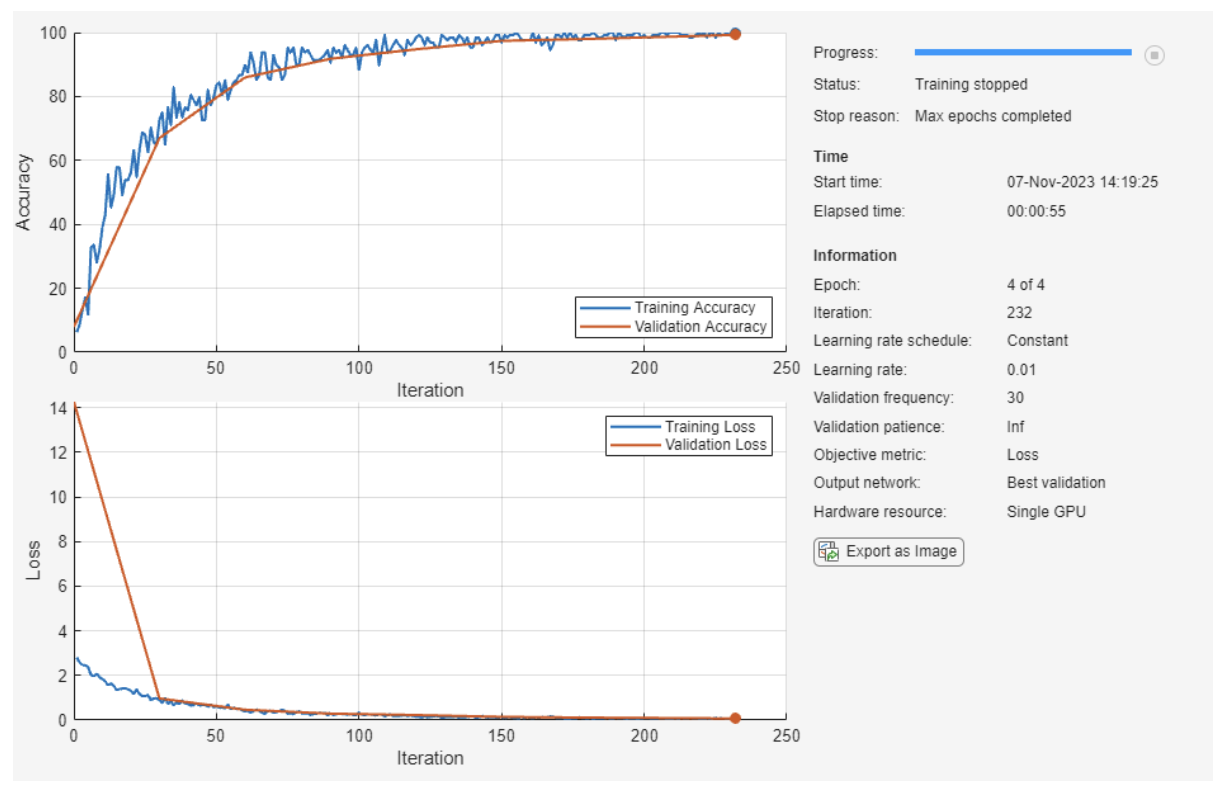
'Plots','training-progress');

1. 使用训练数据训练神经网络

使用 layers 定义的架构、训练数据和训练选项训练神经网络。默认情况下，trainNetwork 使用 GPU（如果有），否则使用 CPU。在 GPU 上训练需要 Parallel Computing Toolbox™ 和支持的 GPU 设备。有关受支持设备的信息，请参阅GPU Computing Requirements (Parallel Computing Toolbox)。您还可以使用 trainingOptions 的 'ExecutionEnvironment' 名称-值对组参数指定执行环境。

训练进度图显示了小批量损失和准确度以及验证损失和准确度。损失是交叉熵损失。准确度是神经网络分类正确的图像的百分比。

net = trainNetwork(imdsTrain,layers,options);



1. 对验证图像进行分类并计算准确度

使用经过训练的神经网络预测验证数据的标签，并计算最终验证准确度。准确度是神经网络预测正确的标签的比例。在本例中，超过 99% 的预测标签与验证集的真实标签相匹配。

YPred = classify(net,imdsValidation);

YValidation = imdsValidation.Labels;

accuracy = sum(YPred == YValidation)/numel(YValidation)

1. 针对回归训练卷积神经网络

此示例说明如何使用卷积神经网络拟合回归模型来预测手写数字的旋转角度。

卷积神经网络（CNN 或 ConvNet）是深度学习的基本工具，尤其适用于分析图像数据。例如，您可以使用 CNN 对图像进行分类。要预测连续数据（例如角度和距离），可以在网络末尾包含回归层。

该示例构造一个卷积神经网络架构，训练网络，并使用经过训练的网络预测手写数字的旋转角度。这些预测对于光学字符识别很有用。

此外，您可以选择使用 imrotate (Image Processing Toolbox™) 旋转图像，并可选择使用 boxplot (Statistics and Machine Learning Toolbox™) 创建残差箱线图

1. 加载数据

数据集包含手写数字的合成图像以及每个图像的旋转角度（以度为单位）。

分别从 MAT 文件 DigitsDataTrain.mat 和 DigitsDataTest.mat 中加载训练数据和测试数据。变量 anglesTrain 和 anglesTest 是以度为单位的旋转角度。训练数据集和测试数据集各包含 5000 个图像。

load DigitsDataTrain.mat

load DigitsDataTest.mat

使用 imshow 显示 20 个随机训练图像。

numTrainImages = numel(anglesTrain);

figure

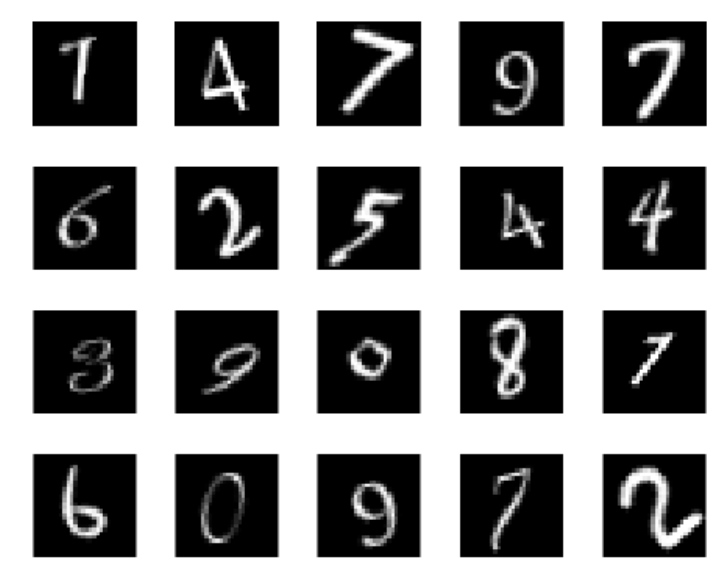
idx = randperm(numTrainImages,20);

for i = 1:numel(idx)

subplot(4,5,i)

imshow(XTrain(:,:,:,idx(i)))

end



1. 检查数据归一化

在训练神经网络时，最好确保数据在网络的所有阶段均归一化。对于使用梯度下降的网络训练，归一化有助于训练的稳定和加速。如果数据比例不佳，则损失可能会变为 NaN，并且网络参数在训练过程中可能发生偏离。归一化数据的常用方法包括重新缩放数据，使其范围变为 [0,1]，或使其均值为 0 且标准差为 1。可以归一化以下数据：

输入数据。在将预测变量输入到网络之前对其进行归一化。在此示例中，输入图像已归一化到范围 [0,1]。

层输出。可以使用批量归一化层来归一化每个卷积层和全连接层的输出。

响应。如果使用批量归一化层来归一化网络末尾的层输出，则网络的预测值在训练开始时就被归一化。如果响应的比例与这些预测值完全不同，则网络训练可能无法收敛。如果您的响应比例不佳，则尝试对其进行归一化，并查看网络训练是否有所改善。如果在训练之前将响应归一化，则必须变换经过训练网络的预测值，以获得原始响应的预测值。

绘制响应的分布。响应（以度为单位的旋转角度）大致均匀地分布在 -45 和 45 之间，效果很好，无需归一化。在分类问题中，输出是类概率，始终需要归一化。

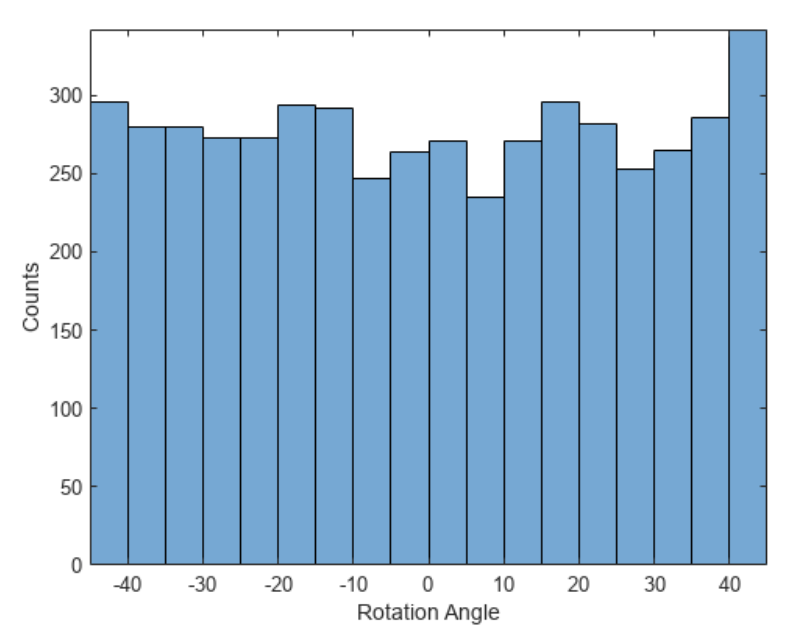
figure

histogram(anglesTrain)

axis tight

ylabel('Counts')

xlabel('Rotation Angle')



通常，数据不必完全归一化。但是，如果在此示例中训练网络来预测 100\*anglesTrain 或 anglesTrain+500 而不是 anglesTrain，则损失将变为 NaN，并且网络参数在训练开始时会发生偏离。即使预测 aY + b 的网络与预测 Y 的网络之间的唯一差异是对最终全连接层的权重和偏置的简单重新缩放，也会出现这些结果。

如果输入或响应的分布非常不均匀或偏斜，您还可以在训练网络之前对数据执行非线性变换（例如，取其对数）。

1. 创建网络层

要求解回归问题，请创建网络层并在网络末尾包含一个回归层。

第一层定义输入数据的大小和类型。输入图像的大小为 28×28×1。创建与训练图像大小相同的图像输入层。

网络的中间层定义网络的核心架构，大多数计算和学习都在此处进行。

最终层定义输出数据的大小和类型。对于回归问题，全连接层必须位于网络末尾的回归层之前。创建一个大小为 1 的全连接输出层以及一个回归层。

在 Layer 数组中将所有层组合在一起。

layers = [

imageInputLayer([28 28 1])

convolution2dLayer(3,8,'Padding','same')

batchNormalizationLayer

reluLayer

averagePooling2dLayer(2,'Stride',2)

convolution2dLayer(3,16,'Padding','same')

batchNormalizationLayer

reluLayer

averagePooling2dLayer(2,'Stride',2)

convolution2dLayer(3,32,'Padding','same')

batchNormalizationLayer

reluLayer

convolution2dLayer(3,32,'Padding','same')

batchNormalizationLayer

reluLayer

dropoutLayer(0.2)

fullyConnectedLayer(1)

regressionLayer];

1. 训练网络

创建网络训练选项。进行 30 轮训练。将初始学习率设置为 0.001，并在 20 轮训练后降低学习率。通过指定验证数据和验证频率，监控训练过程中的网络准确度。软件基于训练数据训练网络，并在训练过程中按固定时间间隔计算基于验证数据的准确度。验证数据不用于更新网络权重。打开训练进度图，关闭命令行窗口输出。

miniBatchSize = 128;

validationFrequency = floor(numel(anglesTrain)/miniBatchSize);

options = trainingOptions('sgdm', ...

'MiniBatchSize',miniBatchSize, ...

'MaxEpochs',30, ...

'InitialLearnRate',1e-3, ...

'LearnRateSchedule','piecewise', ...

'LearnRateDropFactor',0.1, ...

'LearnRateDropPeriod',20, ...

'Shuffle','every-epoch', ...

'ValidationData',{XTest,anglesTest}, ...

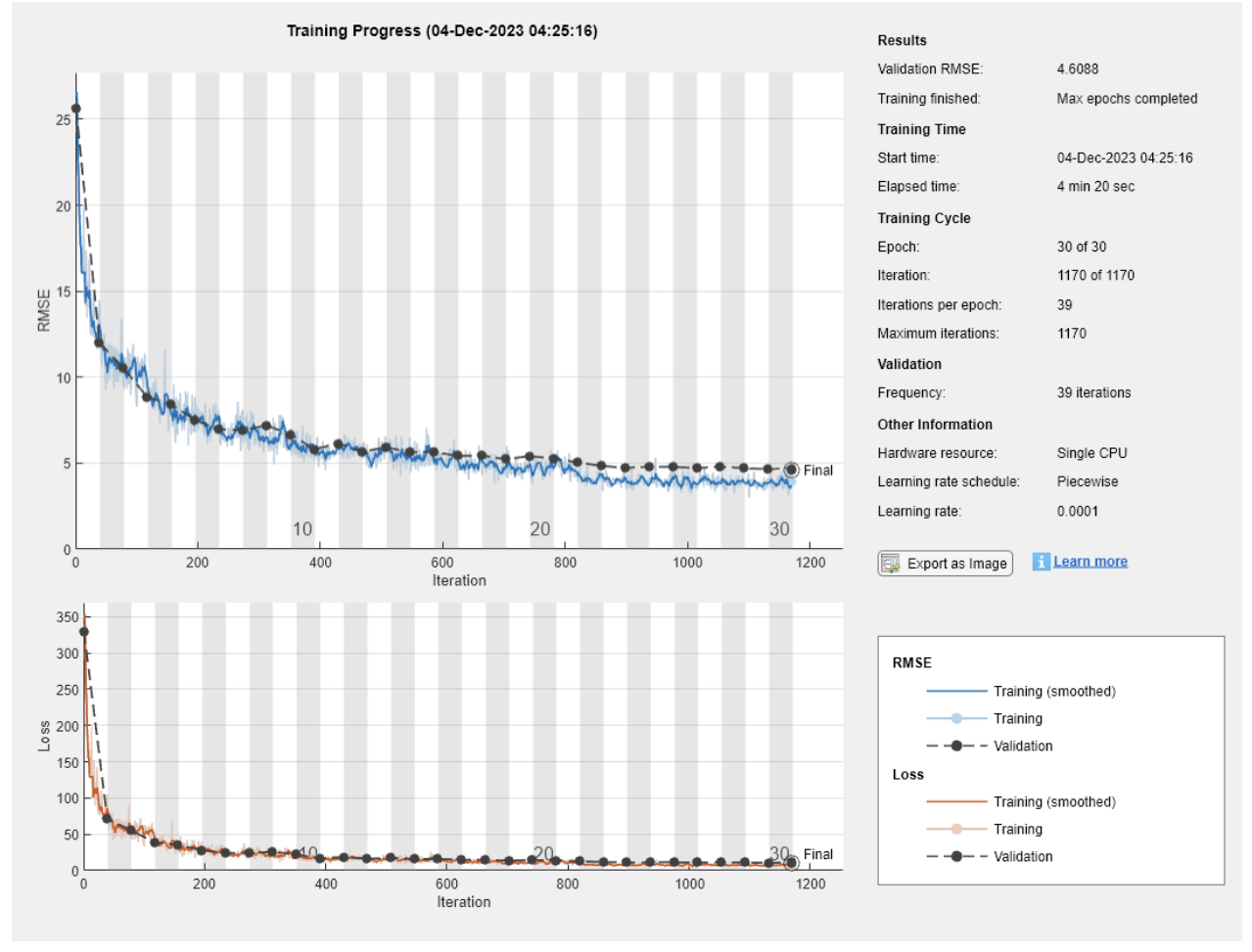
'ValidationFrequency',validationFrequency, ...

'Plots','training-progress', ...

'Verbose',false);

使用 trainNetwork 创建网络。如果存在兼容的 GPU，此命令会使用 GPU。使用 GPU 需要 Parallel Computing Toolbox™ 和支持的 GPU 设备。有关受支持设备的信息，请参阅GPU Computing Requirements (Parallel Computing Toolbox)。否则，trainNetwork 将使用 CPU。

net = trainNetwork(XTrain,anglesTrain,layers,options);



检查 net 的 Layers 属性中包含的网络架构的详细信息。

net.Layers

1. 测试网络

基于测试数据评估准确度来测试网络性能。

使用 predict 预测测试图像的旋转角度。

YTest = predict(net,XTest);

通过计算以下值来评估模型性能：

在可接受误差界限内的预测值的百分比

预测旋转角度和实际旋转角度的均方根误差 (RMSE)

计算预测旋转角度和实际旋转角度之间的预测误差。

predictionError = anglesTest - YTest;

计算在实际角度的可接受误差界限内的预测值的数量。将阈值设置为 10 度。计算此阈值范围内的预测值的百分比。

thr = 10;

numCorrect = sum(abs(predictionError) < thr);

numTestImages = numel(anglesTest);

accuracy = numCorrect/numTestImages

使用均方根误差 (RMSE) 来衡量预测旋转角度和实际旋转角度之间的差异。

squares = predictionError.^2;

rmse = sqrt(mean(squares))

在散点图中可视化预测。绘制预测值对真实值的图。

figure

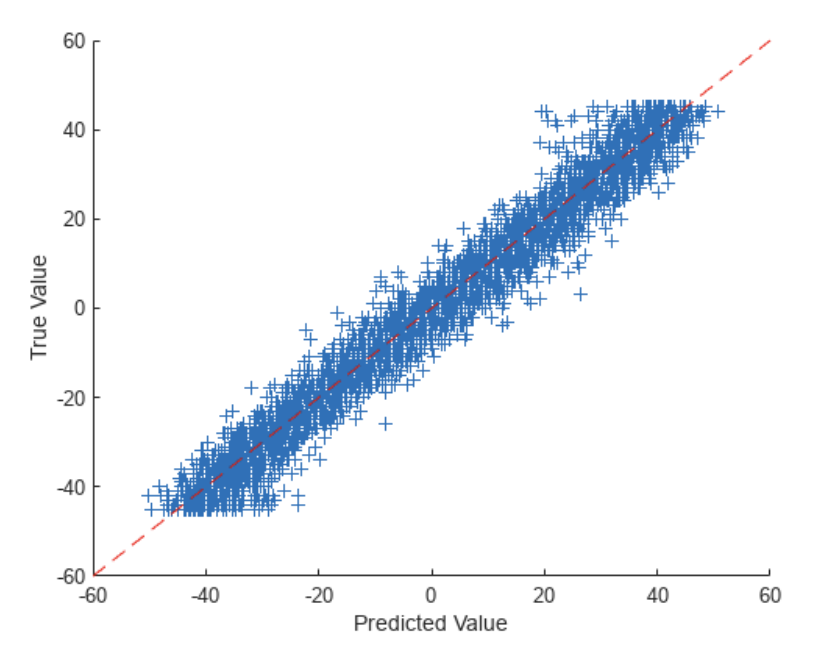
scatter(YTest,anglesTest,'+')

xlabel("Predicted Value")

ylabel("True Value")

hold on

plot([-60 60], [-60 60],'r--')



1. 校正数字旋转

您可以使用 Image Processing Toolbox 中的函数来摆正数字并将它们显示在一起。使用 imrotate (Image Processing Toolbox) 根据预测的旋转角度旋转 49 个样本数字。

idx = randperm(numTestImages,49);

for i = 1:numel(idx)

image = XTest(:,:,:,idx(i));

predictedAngle = YTest(idx(i));

imagesRotated(:,:,:,i) = imrotate(image,predictedAngle,'bicubic','crop');

end

显示原始数字以及校正旋转后的数字。您可以使用 montage (Image Processing Toolbox) 将数字显示在同一个图像上。

figure

subplot(1,2,1)

montage(XTest(:,:,:,idx))

title('Original')

subplot(1,2,2)

montage(imagesRotated)

title('Corrected')

