基于卷积神经网络的单色布匹瑕疵快速检测算法

吴志洋[1)], 卓　勇[1)*], 李　军[1)], 冯勇建[1)], 韩冰冰[2)], 廖生辉[1)]

1) (厦门大学航空航天学院　厦门　361102)
2) (厦门大学软件学院　厦门　361005)
(zhuoyong@xmu.edu.cn)

摘　要: 针对布匹生产企业存在人工检测布匹瑕疵效率低、误检率、漏检率高的问题, 提出一种基于深度卷积神经网络的单色布匹瑕疵检测算法. 首先由于布匹瑕疵的数据规模远小于大型深度卷积神经网络的数据规模, 如果采用大型卷积神经网络, 计算量大且容易导致过拟合, 因此设计了浅层的卷积神经网络结构; 然后提出双网络并行的模型训练方法, 用一个大网络指导小网络的训练过程, 提高模型的训练效果; 最后为了使得深度卷积神经网络模型脱离 GPU 的限制, 能够在普通电脑、移动设备、嵌入式设备中高速运行, 且保证模型检测精度, 提出结合特征图优化卷积核参数的模型压缩算法. 实验结果表明该算法可实现高准确率、高检测速度, 在 PC 机的 CPU 模式下, 检测速度为 135 m/min, 准确率可达到 96.99%.

关键词: 布匹瑕疵检测; 卷积神经网络; 模型压缩; 双网络并行
中图法分类号: TP391　　**DOI:** 10.3724/SP.J.1089.2018.17173

A Fast Monochromatic Fabric Defect Fast Detection Method Based on Convolutional Neural Network

Wu Zhiyang[1)], Zhuo Yong[1)*], Li Jun[1)], Feng Yongjian[1)], Han Bingbing[2)], and Liao Shenghui[1)]

[1)] (*College of Aerospace Engineering*, *Xiamen University*, *Xiamen*　361102)
[2)] (*College of Software Engineering*, *Xiamen University*, *Xiamen*　361005)

Abstract: Low efficiency, high false detection rate and high loss of manual fabric defect detection are problems lying in fabric manufacturing enterprises. Taking the problems as a starting point, this paper proposes a monochromatic fabric defect detection algorithm based on deep convolutional neural networks. Firstly, as the data scale of fabric defect is far smaller than that of large deep convolutional neural networks, it not only requires a large amount of calculation but is also likely to results in overfitting if large convolutional neural networks are adopted. Thus, we adopt shallow convolutional neural networks. Then, we propose a double network parallel model training method. The training process of using a large network to instruct a small network improves the training effect of the model. Finally, in an effort to make the deep convolutional neural network model release from GPU, to make it operate on computers, mobile and embedded devices at high speed, and to guarantee its detection accuracy, we propose a model compression algorithm combining optimization of convolution kernel parameters by feature maps. The results indicated that this fabric defect detection algorithm achieved high accuracy and high detection speed. In the mode of CPU on personal computers, its detection speed reaches 135 meters per minute and its accuracy reaches 96.99%.

收稿日期: 2018-03-15; 修回日期: 2018-08-08. 基金项目: 国家自然科学基金(51605403); 2016 年工信部智能制造综合标准化与新模式应用项目(2016-213). 吴志洋(1989—), 男, 硕士研究生, 主要研究方向为深度学习、计算机视觉; 卓　勇(1970—), 男, 博士, 教授, 硕士生导师, 论文通讯作者, 主要研究方向为图像处理、人工智能; 李　军(1991—), 男, 硕士研究生, 主要研究方向为数字图像处理; 冯勇建(1958—), 男, 博士, 教授, 博士生导师, 主要研究方向为深度学习、模式识别; 韩冰冰(1991—), 男, 硕士研究生, 主要研究方向为深度学习、人脸识别; 廖生辉(1994—), 男, 硕士研究生, 主要研究方向为计算机视觉.

Key words: fabric defect detection; convolutional neural network; model compression; double network parallel

纺织工业是我国国民经济传统的支柱产业，也是国际竞争优势明显的产业，更是一项重要的民生产业. 织物瑕疵检测是纺织品质量把控的一个重要内容，传统的织物瑕疵检测以人工的方式进行，存在效率低(检测速度仅为 15~20 m/min)，误检率、漏检率高的问题. 因此，自动化的织物瑕疵检测已经成为自动化、纺织、信息技术等领域的研究热点与难点.

织物瑕疵检测的核心在于织物瑕疵的特征提取算法. 目前，对于织物瑕疵的传统检测算法主要分为 3 种: (1) 基于统计学的方法. 如 Ye[1]提出基于图像直方图统计变量的模糊推理，对瑕疵具有旋转平移不变性的优点，但是对图像的噪声特别敏感，且在不规则纹理的织物中瑕疵漏检率高; Thomas 等[2]分别统计行、列的灰度均值，并将其与设定的阈值作比较从而实现瑕疵检测，该方法具有计算简单的优点，并且考虑到了瑕疵检测系统的实时性，但是在光照不均匀等情况下灰度级统计法很难奏效. (2) 基于谱分析的方法. 基于 Gabor 变换的方法[3]和基于滤波器的方法[4]能够基于不同的尺度图像获得较高维度的特征空间，对于边缘瑕疵、孔洞具有很好的检测效果，然而优化滤波器的参数值却极其困难且存在计算量大的缺点. (3) 基于模型的马尔可夫随机场[5]、自动回归[6]等方法. 这类方法通过随机过程建模来区分正常布匹纹理与瑕疵块纹理的不同分布，适用于断线、断针引起的表面缺陷，但对于光照、图片噪声比较敏感.

综上所述，已有的研究成果在检测精度、检测速度、实用性等方面有很好的借鉴作用，但现有的算法难以兼顾检测精度与检测速度，以及无法很好地消除光照不均匀、图片噪声的影响，且绝大多数工作仅仅针对小规模图片进行效果验证，对算法的有效性及可行性缺乏充分验证.

当前，卷积神经网络(convolutional neural network, CNN)在计算机视觉领域展示出了极好的效果，在 ImageNet 比赛(ImageNet large scale visual recognition challenge, ILSVRC)[7]中，1 000 类物体识别的错误率在不断减小，CNN 模型的识别错误率已经低于人类. 近年来涌现出了不少优秀的 CNN 模型: 2012 年的 AlexNet[7]，2014 年的 VGGNet[8]，2015 年的 ResNet[9]，2016 年的 DenseNet[10]. CNN 模拟了人的视觉系统，从低级的边缘特征到高级的语义特征，有着丰富的特征表达能力，且在旋转、平移不变性、复杂的背景、光照不均匀的条件下具备更强的分类与泛化能力.

在布匹瑕疵检测方面，景军锋等[11]结合深度 CNN 与 Meanshift 算法分割布匹缺陷确定缺陷位置，结果以二值图像给出，但未给出瑕疵的类别且没有给出算法在指定硬件条件下的执行时间，而瑕疵的类别与计算实时性对于布匹质量的评估是极为重要的. 本文将 CNN 应用于布匹瑕疵检测，把布匹瑕疵检测任务定义为图片网格切分与分类. 首先采集布匹出现的瑕疵，包括 7 种瑕疵(破洞、纬纱、上胶不均匀、刮纱、竹节、叠纱和脏污)及正常布匹共 8 种样本图像; 然后利用这些样本训练分类模型; 最后将工业相机实际拍摄的图像进行网格切分，将每个切分的小块送进训练好的模型进行识别，除了能够识别出该块区域布匹所属的瑕疵类别，也能够对瑕疵进行粗略的定位，达到缺陷检测的目的.

传统的深度 CNN 模型很难在布匹瑕疵检测中得到应用，这是因为一种优秀的自动化布匹瑕疵检测方法必须同时兼顾检测精度与检测的实时性，而传统的深度学习模型(如 AlexNet, GoogleNet, ResNet)虽然在 1 000 种物体的分类精度已经达到很高的水平，但是由于这些模型的参数数量(卷积核数量与网络层数)过于庞大，在图片识别实时性、内存空间占用这 2 个层面上难应用于普通的 PC 机、移动设备或者嵌入式设备. 一般而言，模型大小取决于训练数据的规模，而布匹瑕疵的类别相对较少，数据规模要远远小于传统网络模型的训练数据规模. 例如，王洋等[12]针对数据集的数据规模设计了一个含有 2 个卷积层、2 个池化层、1 个全连接层的精简网络模型，用于高速铁路侵限异物特征提取的 6 分类网络，并取得了良好的效果.

鉴于此，本文首先针对布匹瑕疵检测的数据规模设计了一个包含 3 个卷积层、3 个池化层、1 个全连接层的 CNN 模型; 然后提出一种双网络并行的模型训练方法，用一个大网络来指导小网络的训练过程，提升模型训练效果; 最后对于每个层的卷积核个数，当个数太少时则网络提取的浅层特征不够细致，导致高层语义特征也很难得到较好的组合而影响识别效果，而当个数过多时又达不到计算的实时性. 如果简单地通过做几组不同卷积核个数的对比实验来确定每个卷积层的卷积

核数量，则很难得到最理想的模型加速效果. 因此，本文针对训练好的网络模型，提出一种结合特征图优化卷积核参数的模型压缩算法，在保证准确率的同时进一步减少计算量，加快检测速度. 在某布匹生产企业中实际采集的大规模数据集上进行实验的结果表明，本文算法可实现高准确率、高检测速度的目的.

1 CNN

经典的 CNN[13]由卷积层 C、池化层 P、全连接层 FC、激活函数层 ReLU，分类器损失函数层 Softmax 组成，如图 1 所示，图中省略了 ReLU.

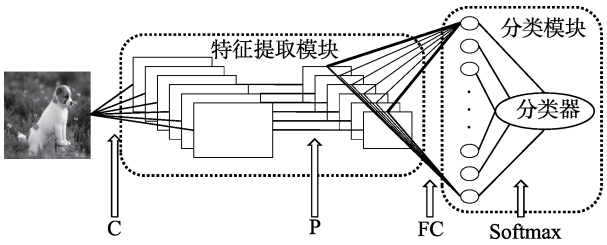


图 1 经典 CNN 结构

C 用来提取特征，其计算公式为

$$y^{j(r)}=f\left(\sum_{i=1}^{I}k^{ij(r)}*x^{i(r)}+b^{j}\right),j=1,2,\cdots,J.$$

其中，输入特征图的通道数为 I；输出特征图的通道数为 J；x^i 表示第 i 个通道输入特征图；y^j 表示第 j 个通道输出特征图；k^{ij} 代表 x^i 与 y^j 之间的卷积核；*表示卷积操作；b^j 代表第 j 个通道输出特征图的偏置项，上标 (r) 代表卷积区域，f 表示激活函数，用来增加模型的非线性.

P 用来降低特征维度，降低计算量，其计算方式为以 $n\times n$ 的滑动窗口在输入特征图上进行不重叠滑窗，在每个滑窗位置取最大值或者平均值.

FC 上的每个神经元都与输入特征图上的每个像素建立连接，用来综合所有提取到的特征，最后将这些特征送入 Softmax 分类器中，输出当前样本属于各个类别的概率值.

2 双网络并行训练结构

本文提出的双网络并行训练结构如图 2 所示，其中省略了 ReLU 层和 LRN 层; AlexNet 模型已被广泛应用于图像识别，由 5 个 C, 3 个 P, 3 个 FC, 1 个 Softmax，以及每个 C 后面接一个 ReLU 层和局部响应归一化(LRN)层组成; OurNet 为本文提出的网络结构.

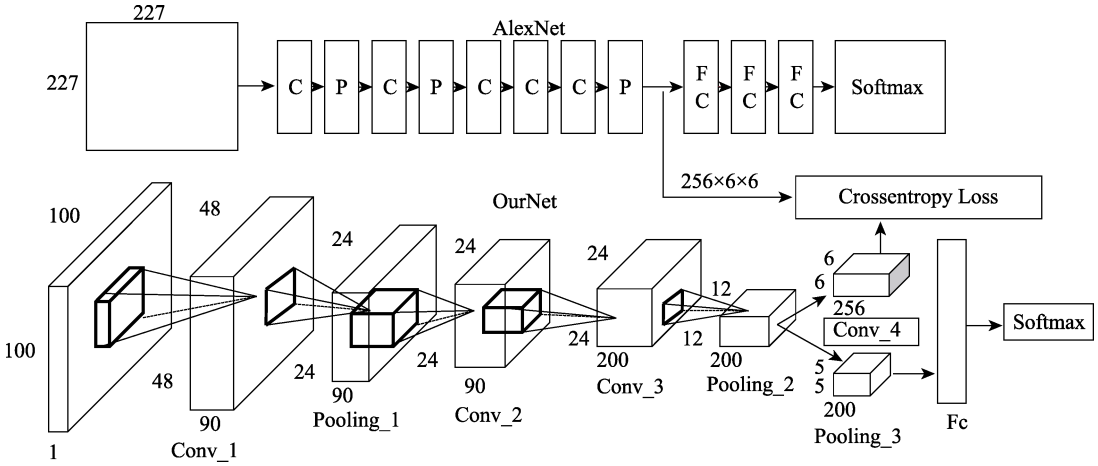


图 2 双网络并行结构

本文模型拓扑结构如表 1 所示，其中，Conv_4 为第 6 层的一个分支，它与 AlexNet 通过交叉熵损失函数连接，在网络训练完毕后，该分支将被移除.

并行网络的损失函数表示为

$$J=\alpha L_{\text{cross}}+(1-\alpha)L_{\text{class}}(\theta) \tag{1}$$

其中，$\alpha=0.23$；

$$L_{\text{class}}(\boldsymbol{\theta})=\frac{-1}{N}\left(\sum_{i=1}^{N}\sum_{j=1}^{k}1\{y^{(i)}=j\}\log\frac{\mathrm{e}^{\boldsymbol{\theta}_j^{\mathrm{T}}x^{(i)}}}{\sum_{l=1}^{k}\mathrm{e}^{\boldsymbol{\theta}_l^{\mathrm{T}}x^{(i)}}}\right) \tag{2}$$

为分类任务损失函数 Softmax；$\boldsymbol{\theta}$ 是模型的训练参数；N 表示每个批次训练样本数量；$y^{(i)}$ 表示类别标签；$x^{(i)}$ 表示输入特征；k 表示分类任务的类别总数.

表 1　本文模型 CNN 设计

网络结构	类型	参数	步长
输入层	数据层		
Conv_1	C	90/5×5	2
ReLU_1	激活层		
Pooling_1	P	2×2	2
Conv_2	C	90/1×1	1
ReLU_2	激活层		
Conv_3	C	200/3×3	1
ReLU_3	激活层		
Pooling_2	P	3×3	2
Pooling_3	P	4×4	2
FC	FC	8	
Conv_4	C	256/7×7	1
Softmax	损失函数层		

L_{cross} 为 OurNet 与 AlexNet 进行联合训练的交叉熵损失函数, 即图 2 中的 Crossentropy Loss. 通过该损失函数约束 2 个网络输出特征图分布的一致性, 进而引导本文提出的小网络进行训练, 其定义为

$$L_{\text{cross}}=\frac{-1}{N}\sum_{i}^{N}[t_i\log s_i+(1-t_i)\log(1-s_i)] \tag{3}$$

其中, N=256×6×6, $\boldsymbol{t}$ 为 AlexNet 在最后一个 P 输出展开的特征向量; $\boldsymbol{s}$ 为本文提出的网络 Conv_4 层输出展开的特征向量.

并行网络训练时, AlexNet 输入数据大小为 227×227×3, 其卷积核参数采用 AlexNet 在 ImageNet 数据集的 1 000 类分类中训练好的参数进行初始化, OurNet 采用高斯分布初始化参数, 其输入数据大小为 100×100×3. 采用反向传播梯度下降算法[13], 权重更新公式为

$$\begin{cases} W_{ij}^{l}=W_{ij}^{l}-\alpha\dfrac{\partial J(W,b)}{\partial W_{ij}^{l}} \\ b_i^{l}\ =b_i^{l}-\alpha\dfrac{\partial J(W,b)}{\partial b_i^{l}} \end{cases} \tag{4}$$

根据人们在深度学习中大量尝试以及本文的分析, 为了得到一个特征提取能力较强的 CNN, 本文模型的设计原则总结如下:

(1) 针对不同数据规模的任务, 需要控制好网络中参数的数量, 一是为了降低模型计算量以及内存占用空间; 二是在一定程度上防止了过拟合.

(2) 在进行模型压缩分析时发现, 浅层的卷积层可以踢除掉更大比例的卷积核而较小地影响到模型的精度, 因此, 对浅层的卷积层 Conv_1, Conv_2 的卷积核通道设置为 90, 而深层的卷积层 Conv_3 的通道设置为 200.

(3) 多任务学习[14]与高级语义信息. 多任务学习能够利用任务之间的相关性相互促进, 共享参数的那部分权值将被约束为更好的值, 使模型具有更好的泛化能力. 本文利用更深的网络 AlexNet 提取到更高级的语义特征, 这些更高级的语义特征更有利于区分图像的类别. 相比于训练标签, 更高级的语义特征能够提供更多的信息, 因此 OurNet 的输出特征向更高级的语义特征逼近, 同时网络进行分类训练任务, 以此构建双网络并行训练.

3　模型压缩算法

模型压缩想的目的如下: 一是减少模型在内存的占用空间; 二是减少模型的计算量, 加快模型计算速度. 这两个目的都致力于让深度 CNN 能够部署在普通 PC 机、移动端设备、嵌入式设备中高速运行, 达到实时性.

3.1　相关的模型压缩算法

网络剪枝(pruning)是模型压缩算法研究中一种较为常用的方法. 剪枝指基于某种规则删除模型中冗余的参数, 减少模型计算量与模型大小, 同时在一定程度上提高模型的泛化能力. Han 等[15]提出将网络中所有低于设定的阈值的参数删除, 得到一个非结构化的稀疏网络, 属于卷积核内权重剪枝; 然后通过微调(fine-tuning, 此处指用剪枝后的模型参数作为网络的初始参数, 再用原有的训练集进行重新训练, 调整网络参数值)尽可能地恢复模型识别精度. 但该方法在前向传播时, 由于其非结构化的稀疏性, 必须开发专用的软件计算库或者是设计出专门的硬件, 导致在实际中, 应用门槛非常高. 为了克服专用稀疏卷积计算库以及专用硬件的缺陷, 在卷积核通道层面上进行剪枝是实际应用中更为合理的选择. Li 等[16]提出 Weight sum 算法, 通过计算每个卷积核参数的绝对值之和, 作为衡量一个卷积核是否重要的依据; Hu 等[17]提出 APoZ 算法, 通过实验发现, 经过激活后的大部分神经元的结果是趋于 0 的, 那些激活后为 0 的神经元是冗余的, 通过统计每个卷积核对应的特征图激活后值为 0 的神经元数量来评估对应的卷积核的重要程度. 然而, 这些方法很明显的缺陷在于

仅仅考虑了在单独每一层中进行卷积核的剪枝, 而没有考虑到当前层的剪枝对下一层带来的影响. 鉴于此, Luo 等[18]提出了 Thinet, 如果能够用第 i+1 层输入的一个子集, 就能够产生和以全集作为输入的结果近似, 那么第 i 层对应该子集之外的卷积核则可以被踢除. 这样的做法考虑到了前后层之间的关系, 在 ImageNet[7]等数据集上的实验效果也优于 Weight sum, APoZ 等算法.

目前, 所有基于剪枝的模型压缩算法均是从卷积核通道选择保留或者踢除的角度进行研究, 重点是如何从已有的卷积核集合中尽可能地挑选出一个理想的子集合, 然后通过逐层微调来恢复模型精度, 即每一层挑选出要保留的参数之后都需要对模型进行一次 fine-tuning, 否则, 模型精度将急剧下降.

3.2 本文模型压缩算法

(1) 模型压缩后, 其拟合能力将随着压缩率的提高而迅速降低. 而基于剪枝的模型压缩算法依靠对压缩后的模型重训练来尽可能地恢复其识别精度, 训练过程依然采用学习样本与标签之间映射关系的方式, 样本标签能提供当前类别与其余几个类别之间的关系, 而对于剩余类别两两之间的联系则无法给出. 这种方式大大提高了小模型拟合数据的难度.

(2) 特征图可以看做样本所有局部特征的集合, 能够反映不同样本之间特征的差异. 然而, 绝大数基于剪枝的模型压缩算法仅仅利用大模型的参数来初始化小模型, 而大模型提取特征的能力(反映在特征图上)无法得到有效利用.

(3) Thinet 方法可以看做小模型间接地学习到大模型层与层之间的关系, 该方法表明了层与层之间的关系有助于提升小模型识别精度.

基于以上 3 点分析, 本文结合 Thinet 的思想, 提出在卷积核筛选完成后, 通过约束初始模型与修剪后模型的第 i+2 层输入特征图分布的一致性来更新保留下来的卷积核参数. 本文算法的创新体现在 2 个方面:

(1) 特征图能够反映不同类别之间的差异, 约束特征图之间的分布, 弥补了直接训练标签的不足, 相当于增加了更强的监督信息;

(2) 大多数基于剪枝的模型压缩算法没有考虑到模型前后层之间的关系, 本文算法通过约束大小模型第 i+2 层输入的一致性来更新小模型第 i 层卷积核参数, 相当于显式地加入了层与层之间的联系约束, 能够给其他基于剪枝的模型压缩算法加入层与层之间的相关性约束, 提升其原有性能.

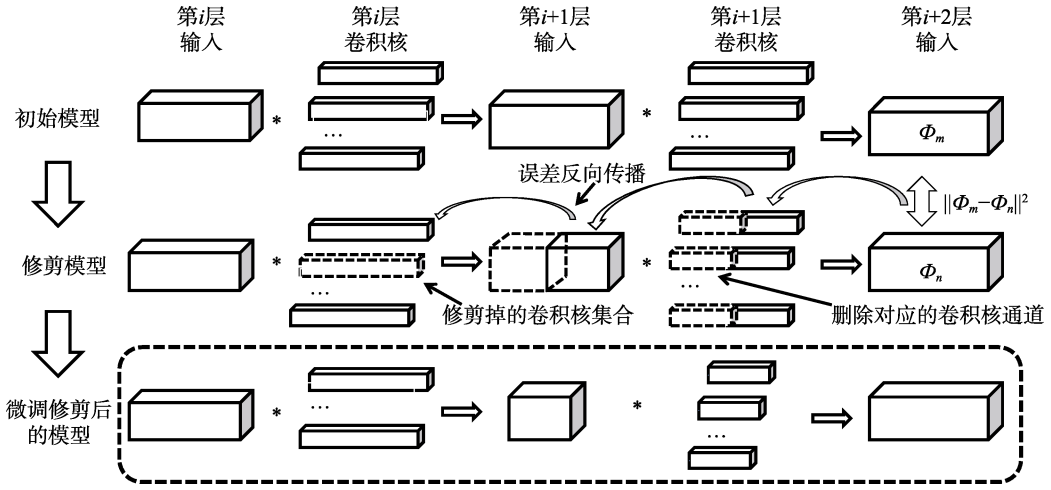


图 3 本文模型压缩算法示意图

本文模型压缩算法示意图如图 3 所示, 假设当前要对网络中第 i 层卷积层进行剪枝, 算法分为两大步骤: 一是根据卷积核的重要程度与压缩率进行卷积核筛选; 二是对保留下来的卷积核进行参数更新. 具体步骤如下:

Step1. 读取一个批次的图片经过未压缩的网络进行前向传播, 获得每层的输出特征图.

Step2. 根据定义的压缩率, 利用文献[16]方法或其他已有的基于剪枝的方法对卷积核进行剔除.

Step3. 以第 i 层保留下来的卷积核、第 i+1 层截短后的卷积核构建一个含有 2 个卷积层、2 个激活函数层、损失函数为欧氏距离的小型网络(记为

SmallNet)，用未压缩模型的输出 Φ_m 作为训练数据的标签，优化目标函数

$$J(w,b;x)=\arg\min\frac{1}{2}\|\Phi_{n,w,b}(x)-\Phi_m(x)\|^2 \qquad (5)$$

根据式(4)，只更新 SmallNet 第 1 层卷积核的参数，即对应图中的第 i 层保留下来的卷积核参数. 其中，x 表示第 i 层的输入数据，Φ_m 为剪枝前的模型输出，w,b 代表第 i 层保留下来的卷积核参数，Φ_n 为剪枝后新构建的模型的输出. 通过式(5)调整第 i 层保留下来的卷积核内参数值，使得所保留下来的卷积核所产生的结果 Φ_n 能够不断向删减之前所产生的结果 Φ_m 逼近，即约束图 3 中 Φ_n 与 Φ_m 的一致性. 其中，SmallNet 为进行 100 个迭代次数的优化.

Step4. 重复执行 Step2~Step3，直到所有卷积层压缩完毕.

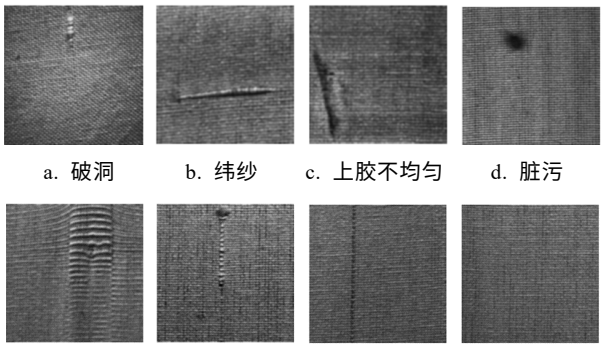
Step5. 加载下一个批次的图片，重复 Step1~Step4，直到所有批次的图片结束.

Step6. 对压缩完成后的模型进行一次 fine-tuning.

4　实验结果与分析

4.1　建立布匹瑕疵样本库

本文采用黑白工业线阵相机 BaslerraL 12288-8gm 与线光源搭建硬件环境，同时采用暗场照明的方式突出布匹的边缘和轮廓信息. 以某布匹生产企业单色布生产线上 9 种单色织物(红色、白色、紫色、暗红色、藏蓝色、浅黄色、暗红色、粉红色和深灰色)出现瑕疵，有破洞、纬纱、上胶不均匀、刮纱、竹节、叠纱和脏污，这 7 种瑕疵图像与正常布匹图像建立布匹瑕疵数据集，如图 4 所示.



a. 破洞　b. 纬纱　c. 上胶不均匀　d. 脏污

e. 刮纱　f. 竹节　g. 叠纱　h. 正常

图 4　样本示例

工业相机采集回来的图像分辨率为 2048×4096，对应实际布匹大小为 90 cm×45 cm，将每幅图分为 3 行 6 列，共 18 幅子图，对这些子图进行人工标记瑕疵类别，且在原始图像上用同样大小的滑动窗口进行样本扩充，每个类别得到 2 000 幅图像，每幅图像缩放到 100×100 像素和 227×227 像素 2 种规格.

4.2　双网络并行训练与结果对比

训练实验建立在 64 位的 Linux 操作系统和 NVIDIA GTX Geforce 1080 GPU 的服务器上，模型实验建立在普通 PC 机上；采用 Windows 10, Inter(R) i5-4570 3.2 GHz CPU, 4 GB 内存；深度学习框为 caffe，下载地址为 https://github.com/BVLC/caffe，相关软件有 Python 2.7 版本、Matlab2014b 版本.

将建立的数据集的各个类别中采用均匀随机抽样的方式抽取 70%, 20%, 10%分别作为训练集、测试集和验证集. 所有训练数据训练 10 次，对本文提出的网络训练参数定义如下：每个批次图像数量为 128、权重衰减系数为 0.000 1、初始学习率为 0.001，每迭代 1 万次学习率缩小 10 倍；用于引导本文模型训练的 AlexNet 模型的训练参数设置与其原有的参数设置保持不变；OurNet 模型参数采用高斯分布初始化，AlexNet 模型则在其原有参数上进行微调.

为了验证大模型对小模型的训练引导作用，本文采用相同的训练策略对提出的网络进行 2 组训练：一组是 OurNet 单独训练，另一组是加入了 AlexNet 的双网络并行训练. 其中，式(1)中包含超参数 α，该参数很难通过训练的方式进行优化，因此本文采用交叉验证方法，将原有的训练集划分为训练集与测试集，根据模型在测试集上的表现确定超参数的值.

图 5 所示为超参数取不同值时模型在测试集上的分类准确率，可以看出，当 α=0.23 时，模型的表现最好；因此，令 α=0.23.

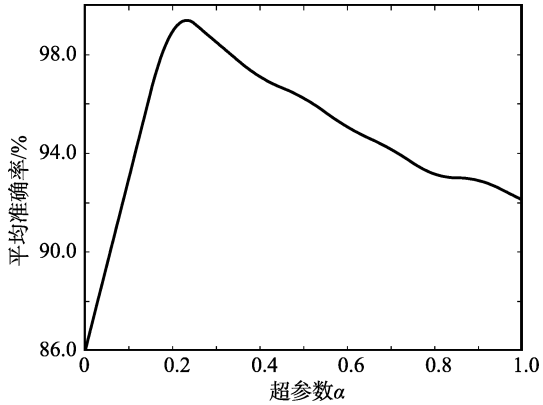


图 5　超参数与准确率关系曲线图

训练情况如图 6 所示, 为了更直观地观察误差的下降情况, 当所有数据训练完一遍后, 统计其平均误差. 可以看出, 双网络联合训练使得模型更快收敛, 且误差收敛到更低的水平, 说明了本文算法的有效性. 为了进一步验证模型的有效性, 本文在测试集上分别对各个颜色的布匹及不同种类的瑕疵统计准确率, 结果如表 2 所示. 其中, 加粗的数值为双网络并行训练的模型(简记为模型 1)得到的结果, 括号内的数值为单网络训练模型(简记为模型 2)的检测结果. 可以看出, 模型 1 比与模型 2 的识别准确率有较大幅度的提高.

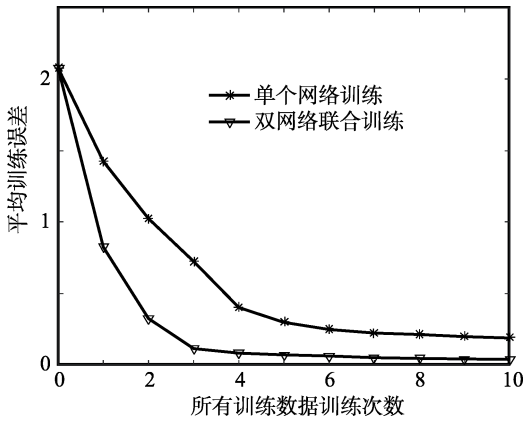


图 6 网络训练时误差收敛情况

表 2 不同训练方式得到的模型识别准确率对比 %

布匹颜色	破洞	纬纱	上胶不均匀	刮纱	竹节	叠纱	脏污
红色	**99.2**(86.1)	**97.2**(86.7)	**95.2**(77.9)	**100.0**(87.7)	**98.3**(81.2)	**99.3**(84.4)	**97.5**(86.2)
白色	**99.3**(86.5)	**98.9**(88.2)	**98.3**(76.2)	**99.2**(76.4)	**96.2**(90.1)	**98.2**(76.5)	**100.0**(88.1)
紫色	**98.5**(72.2)	**98.2**(76.1)	**96.0**(65.9)	**98.7**(83.2)	**99.2**(86.2)	**97.2**(75.2)	**98.2**(76.6)
暗红色	**98.1**(90.2)	**98.2**(66.3)	**98.2**(76.8)	**99.3**(81.9)	**96.8**(88.3)	**98.2**(66.7)	**96.7**(66.9)
藏蓝色	**99.2**(88.4)	**97.5**(86.3)	**98.4**(80.2)	**100.0**(86.7)	**97.6**(66.7)	**98.9**(83.2)	**98.2**(68.7)
浅黄色	**99.1**(86.2)	**99.2**(74.2)	**95.1**(83.5)	**97.9**(87.5)	**100.0**(76.2)	**97.6**(87.2)	**99.5**(76.3)
暗红色	**100.0**(81.6)	**99.1**(78.6)	**96.2**(76.7)	**98.2**(88.6)	**96.2**(70.5)	**99.4**(91.2)	**99.2**(86.5)
粉红色	**98.5**(86.2)	**98.7**(79.4)	**98.2**(76.1)	**98.3**(90.1)	**95.9**(91.2)	**99.2**(86.7)	**98.7**(72.2)
深灰色	**98.6**(88.2)	**99.0**(86.2)	**98.7**(85.2)	**99.1**(83.8)	**96.2**(86.4)	**99.0**(80.2)	**97.2**(81.2)

此外, 将本文基于 CNN 的算法与经典机器学习算法 SVM 进行比较, SVM 算法采用 LibSVM[19], 核函数采用 RBF(radial basis function), 即

$$K(u,v)=\exp(-\gamma\,|u-v|^2).$$

通过交叉验证方法求解行到最优的参数: 惩罚系数为 141.2, 实验结果如表 3 所示; 其中, 本文算法将不再对颜色进行区分, 而是直接统计各瑕疵的识别准确率. 可以看出, 本文算法比 SVM 算法在布匹瑕疵识别上具有更好的识别效果.

表 3 本文算法与 **SVM** 算法识别准确率对比 %

布匹瑕疵	本文算法	SVM 算法
破洞	99.3	73.7
纬纱	99.1	76.8
上胶不均匀	98.8	65.7
刮纱	99.1	81.5
竹节	98.7	67.6
叠纱	99.0	63.8
脏污	99.2	70.1

4.3 模型压缩

本节中将不再具体对布匹的颜色、瑕疵的识别准确率进行细分, 而是对每个测试样本的识别正确与否做一个整体的统计.

首先采用 Weight sum 方法[16], 在压缩率为 10%~60%(剔除各个 C 中 60%的卷积核)的条件下对本文提出的原始模型进行压缩.

在实际的应用中, 对于识别精度和识别速度的权衡是一个非常重要的问题. 本文在 PC 机(参考第 4.2 节)、安卓手机(Android)条件下对 6 种压缩率下的识别准确率和执行速度进行对比, 对工程人员来说这是一项十分重要的参考依据, 其中, 安卓手机的配置为 Android5.0 系统、2.2 GHz CPU.

为进一步验证卷积核筛选环节的必要性, 本文在各个压缩率(10%~60%)条件下定义 6 个 CNN 模型, 其卷积核数量根据压缩率确定; 采用高斯分布初始化参数, 双网络并行训练这 6 个模型. 识别率对比结果如表 4 所示.

可以看出, 本文模型在占用内存空间上小于 1 MB, 可以很容易的加载到 PC 机、移动端与嵌入式设备; 在执行速度上, 不管是普通 PC 机还是移动端设备, 均有着十分良好的表现, 且产生更低的

能源消耗. 瑕疵识别精度在压缩率 60%(剔除每个 C 中 60%的参数)时有了明显下降，因此，在实际应用时本文选取的压缩率为 50%. 注意：在统计总浮点运算时，由于 CNN 的绝大多数计算量都集中在卷积层，因此仅统计所有 C 的浮点运算量的总和，在识别速度上，未加入从摄像头获取图片的时间. 另外，表 4 有 2 个识别准确率：识别率 1 是由模型压缩之后得到；识别率 2 是在相应压缩率条件下，将网络的卷积核数目直接设置为压缩后的网络的卷积核数目，直接训练得到的模型识别准确率. 可以看出，在各个压缩率条件下，未采用压缩而直接训练得到的模型，其识别率比采用压缩算法得到的模型识别率低，且 2 个识别率之间的差距随着压缩率的加大而加大，在压缩率为 10%时，两者识别准确率之差为 10.73%；压缩率为 60%时，两者之差为 26.30%.

表 4　不同压缩比条件下模型识别率与单张子图识别时间对比

压缩比例	$10^{-6}\times$总浮点运算	模型总大小/KB	检测时间/ms		识别准确率/%	
			CPU	Android	1	2
10	169.71	706	20.95	28.30	99.28	85.55
20	150.84	574	19.52	26.01	98.53	83.37
30	132.00	456	17.02	23.59	97.33	79.13
40	113.13	352	14.68	21.23	96.98	73.69
50	94.27	260	11.24	18.69	96.22	71.09
60	75.43	182	9.83	16.82	85.99	59.69

为进一步验证本文算法的有效性，与 APoZ[17]，Thinet[18]等压缩策略及其组合策略在多种压缩率的条件下对模型识别精度进行比较，结果如表 5 所示. 其中，Weight sum+本文算法的含义为：采用 Weight sum 方法进行卷积核筛选，然后用本文算法进行卷积核参数优化；其他组合的含义以此类推. 可以看出，本文算法(Weight sum+本文算法、APoZ+本文算法和 Thinet+本文算法)能够提高已有的模型压缩算法在各个压缩率下的性能.

表 5　多种模型压缩算法识别率对比　　%

压缩比例/%	Weight sum	APoZ	Thinet	本文算法+		
				Weight sum	APoZ	Thinet
10	98.33	98.14	99.13	99.28	98.88	99.26
20	97.25	97.02	97.75	98.53	98.42	98.62
30	96.57	95.24	97.42	97.33	97.15	97.95
40	94.65	94.11	97.20	96.98	96.32	97.43
50	93.32	93.11	95.53	96.22	96.15	96.99
60	72.56	70.37	83.25	85.99	83.36	88.21

Thinet+本文算法能够得到更理想的效果，但其筛选方式过于烦琐，大大增加了算法的研发周期，其精确度比 Weight sum+本文算法在压缩率为 50%时，识别精确度仅有 0.77%差距. 在实际研发时，采用 Weight sum+本文算法可加快研发周期.

4.4　模型压缩算法结果分析

(1) 从表 4 可以看出：在同样参数个数的条件下，采用高斯初始化直接训练得到的模型的识别率明显低于从大模型压缩然后训练得到的模型识别率，这 2 种模型之间最大的差异就是模型初始化参数的不同，经过压缩算法得到的模型能够得到更好的初始化. 从更极端的层面考虑，例如，一个不理想的参数初始化方案可能导致在前向传播时出现输出值全部为负值，则经过 ReLU 激活函数后，在特征图上的值全部变为 0，导致该层所有参数的梯度将变为 0，无法调整权值. 因此，模型的参数初始化对模型的训练至关重要.

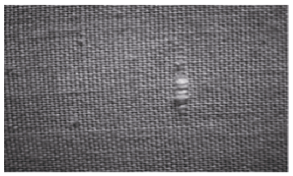
(2) 从表 5 可以看出，对比 Weight sum, ApoZ 和 Thinet 这 3 种算法，前 2 种算法在每一层中独立对卷积核的重要性进行评估；而 Thinet 算法则是考虑到了前后层之间的影响，从而保留了更重要的卷积核，可以认为，Thinet 算法寻找到了一种更有效的模型参数初始化方案.

(3) 从本文算法的效果可以看出，特征图提供的信息对于得到一个更好的模型参数初始化起到了重要作用，同时也能够将层与层之间的相关性引入到其他基于剪枝的模型压缩算法，提升其原有性能.

4.5　布匹检测速度测试

工业相机采集回来的图片分辨率为 4096×2048，对应实际图像大小为 0.9 m×0.45 m，在瑕疵最小长度单位为 15 cm 时，将均等划分为 3 行 6 列共 18 张子图，每张子图代表布匹实际大小为 15 cm×15 cm，如图 7 所示.

取压缩率为 50%，模型处理单幅子图的时间



a. 破洞瑕疵

0	0	0	0	0	0
0	0	0	1	0	0
0	0	0	0	0	0

b. 图像分块识别结果

图 7　瑕疵检测示意图

为 11.24 ms, 则 18 幅子图处理的总时间为 11.24 ms×18 = 202 ms, 对应的布匹长度为 0.45 m. 理论上, 1 min 可以处理(60/0.202)×0.45 m = 135 m. 实际测试时, 在能够及时处理摄像机采集的图像的条件下, 处理的布匹速度为 130~133 m/min 范围内, 原因是在模型测试时并未统计摄像机采集图像与图像传输的时间. 该实时处理速度是人工检测的 6~7 倍, 可代替人工进行布匹检验.

5 结 语

针对传统数字图像技术在布匹瑕疵识别领域表现欠佳的问题, 本文提出将 CNN 应用于该领域, 使得识别性能显著提升. 实验结果表明:

(1) 相对于传统的布匹瑕疵检测算法, 本文算法大大降低了对图像光照不均匀的敏感度, 且鲁棒性更强、精确性更高;

(2) 相对于经典的深度卷积神经网络, 本文针对布匹瑕疵数据规模的 CNN 网络结构与检测方案, 创新性地提出一种双模型并行的模型训练方法, 加快模型的收敛速度与提升了模型性能;

(3) 基于剪枝策略的模型压缩算法往往只从原始模型中提取参数初始化小模型, 忽略了原始模型中隐藏的其他重要信息, 本文提出优化的模型压缩算法将该隐藏信息有效利用起来, 通过对比实验验证了该算法的有效性.

收集布匹瑕疵样本需要耗费大量的人力物力, 基于小样本的模型学习将是下一步的研究重点.

参考文献(References):

[1] Ye Y. Fabric defect detection using fuzzy inductive reasoning based on image histogram statistic variables[C] //Proceedings of the 6th International Conference on Fuzzy Systems and Knowledge Discovery. Los Alamitos: IEEE Computer Society Press, 2009: 191-194

[2] Thomas T, Cattoen M. Automatic inspection of simply patterned material in the textile industry[C] //Proceedings of SPIE. Bellingham: Society of Photo-Optical Instrumentation Engineers, 1994: 2-12

[3] Shu Y, Tan Z. Fabric defects automatic detection using Gabor filters[C] //Proceedings of the 5th World Congress on Intelligent Control and Automation. Los Alamitos: IEEE Computer Society Press, 2004, 4: 3378-3380

[4] Mak K L, Peng. P, Yiu K F C. Fabric defect detection using morphological filters[J]. Image and Vision Computing, 2009: 27(10): 1585-1592

[5] Ozdemir S, Ercil A. Markov random fields and Karhunen-Love transforms for defect inspection of textile products[C] //Proceedings of the IEEE Conference on Emerging Technologies and Factory Atomation. Los Alamitos: IEEE Computer Society Press, 1996, 2: 697-703

[6] Hajimowlana S H, Muscedere R, Jullien G A, *et al*. 1D autoregressive modeling for defect detection in web inspection systems[C] //Proceedings of Midwest Symposium on Circuits and Systems. Los Alamitos: IEEE Computer Society Press, 1998: 318-321

[7] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks[C] //Proceedings of the 25th International Conference on Neural Information Processing Systems. Cambridge: MIT Press, 2012, 1: 1097-1105

[8] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[OL]. [2018-03-15]. http://arxiv.org/abs/1409.1556v6

[9] He K M, Zhang X Y, Ren S Q, *et al*. Deep residual lear-ning for image recognition[OL]. [2018-03-15]. http://arxiv.org/abs/1512.03385v1

[10] Huang G, Liu Z, van der Maaten L, *et al*. Densely connected convolutional networks[C] //Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition. Hawaii: IEEE Computer Society Press, 2016: 2261-2269

[11] Jing Junfeng, Fan Xiaoting, Li Pengfei, *et al*. Yarn-dyed fabric defect detection based on deep-convolutional neural network[J]. Journal of Textile Research, 2017, 38(2): 68-74(in Chinese)
(景军锋, 范晓婷, 李鹏飞, 等. 应用深度卷积神经网络的色织物缺陷检测[J]. 纺织学报, 2017, 38(2): 68-74)

[12] Wang Yang, Yu Zujun, Zhu Liqiang, *et al*. Fast feature extraction algorithm for high-speed railway clearance intruding objects based on CNN[J]. Chinese Journal of Scientific Instrument, 2017, 38(5): 1267-1275(in Chinese)
(王 洋, 余祖俊, 朱力强, 等. 基于 CNN 的高速铁路侵限异物特征快速提取算法[J]. 仪器仪表学报, 2017, 38(5): 1267-1275)

[13] LeCun Y, Boser B, Denker J S, *et al*. Backpropagation applied to handwritten zip code recognition[J]. Neural Computation, 1989, 1(4): 541-551

[14] Caruana R. Multitask learning[J]. Machine Learning, 1997, 28(1): 41-75

[15] Han S, Mao H Z, Dally W J. Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman Coding[OL]. [2018-03-15]. https://arxiv.org/abs/1510.0-0149

[16] Li H, Kadav A, Durdanovic I, *et al*. Pruning filters for efficient ConvNets[OL]. [2018-03-15]. https://arxiv.org/abs/1608.0-8710

[17] Hu H Y, Peng R, Tai Y W, *et al*. Network trimming: a data-driven neuron pruning approach towards efficient deep architectures[OL]. [2018-03-15]. https://arxiv.org/abs/1607.03250v1

[18] Luo J H, Wu J X, Lin W Y. ThiNet: a filter level pruning method for deep neural network compression[C] //Proceedings of International Conference on Computer Vision. Los Alamitos: IEEE Computer Society Press, 2017: 5068-5076

[19] Chang C C, Lin C J. LIBSVM: a library for support vector machines[J]. ACM Transactions on Intelligent Systems and Technology, 2011, 2(3): Article No.27