基于离散傅里叶分析的视觉识别卷积神经网络

李艺海

专业：数学与应用数学　　指导教师：张亚静

摘要　随着方向梯度直方图算法（HOG）在神经网路中得到广泛应用，本文提出并探索一个新的问题：提取图片中物体的边缘特征信息，有没有比 HOG 更效率的算法？本文基于离散傅里叶变换提出了一种新型的提取图片中边缘特征的图片处理的新算法(DFT)。解决了 HOG 的计算时间长，计算精度不够，不兼容不同色彩空间，不能高效处理超高分辨率图像的问题。该方法结合了离散傅里叶变换的优点，将彩色图片从色彩空间域转换到频率域，在频率域进行处理后逆变换回到色彩空间域。针对此算法 Pytorch 下设计了一个小型视觉几何群神经网络（Mini-VGG）。使用不同参数下的理想高通滤波器（IHPF），高斯高通滤波器（GHPF）和布特沃斯高通滤波器（BHPF）与典型的矩形方向梯度直方图算法（R-HOG）处理的数据集 Fashion-MINIST 和 CIFAR10，在 Mini-VGG 上进行图片分类任务对比训练。得出一个较好的图像特征提取算法，验证了此方法的可行性、正确性和一定的优越性。并且在文章的最后，对此算法进行综合性的讨论，阐述了此算法在图像特征提取尤其是在物体边缘提取应用上的未来发展与展望。

关键词：离散傅里叶分析；方向梯度直方图；小型视觉几何群神经网络；高通滤波器；CIFAR10；Fashion-MINIST

1 引言

1.1 介绍

计算机领域的学者们从上世纪中叶开始就在尝试通过计算机来识别图片中的内容，理解图片中包含的信息，甚至理解图片中的信息整理后主动表达。受到当时计算机计算能力的限制，直到二十一世纪初期才出现了可以识别图片中人脸信息的算法[27]。到了 2009 年，随着数据量的越来越大，李飞飞教授带领的研究团队整理出了首个神经网络图像识别分类任务数据集 ImageNet，该数据集共有 1000 个类别、1431167 张 32*32 像素的图片。

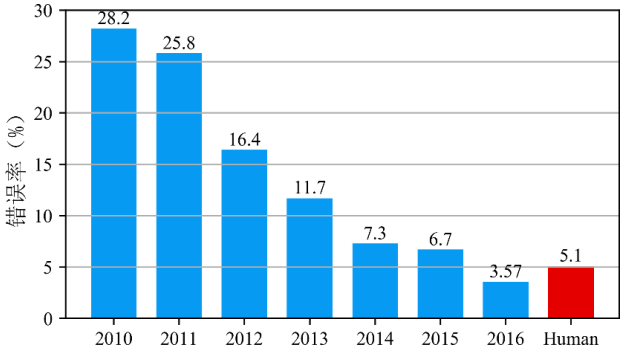


图 1：大规模视觉识别竞赛（LSVRC）结果

为了推动视觉识别算法的开发，团队基于 ImageNet 数据集，又主办了大规模视觉识别竞赛。该竞赛在后续几年激发了大量学者设计可以进行训练学习的图片识别算法，这种特殊的，可以具有学习特性的算法被叫做神经网络。在竞赛开展的几年里，有一个不可忽略的里程碑。在 2012 年竞赛中

首次出现了卷积神经网络 AlexNet[28]，该网络算法极大推进了视觉识别的发展。将 ImageNet 数据集分类任务的错误率由往前的 25%以上，降到了 15%左右（图 1）。

随着往后在神经网络在图片识别领域的继续发展，在后续的几年竞赛中，分别出现了像 VGG[22]、GoogleNet[29]、ResNet[24]、DenseNet[30]这样错误率均在 5%左右的神经网络。这个错误率已经可以与人相比拟，其中甚至有的表现优于常人。从 2017 年竞赛停办至今，在视觉识别领域卷积神经网络得到长足的发展。现在的神经网络不仅仅应用于图片识别任务，还在应用自然语言处理，深度机器学习等领域。但是目前主流的应用于图像识别的神经网络主流依然是各种基于卷积核的神经网络，新型网络大部分创新处体现在网络的架构设计上面。比如华为与清华大学在 2020 年国际计算机视觉与模式识别会议（CVPR）上联合发布了一种只有加法构成的神经网络 AdderNet[32]。在 ImageNet 等大型分类任务上同样取得了非常好的效果。

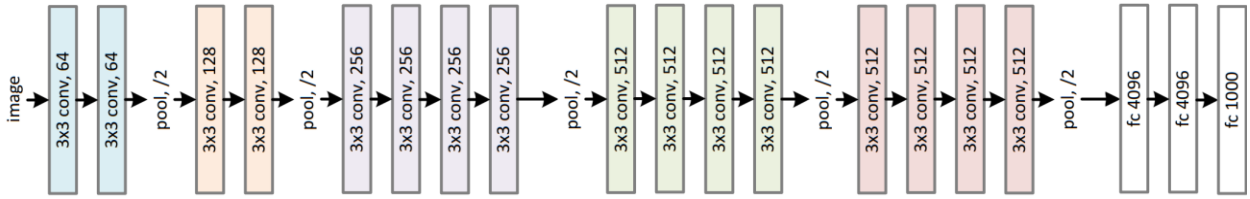
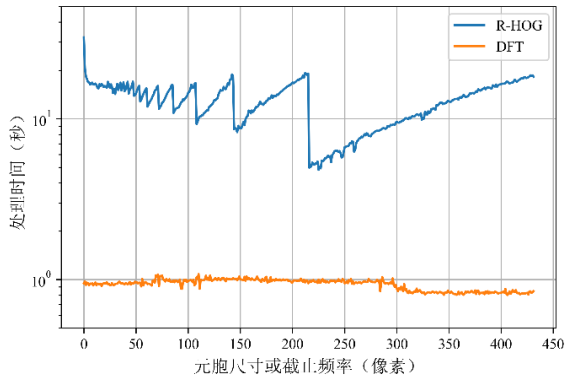


图 2：视觉几何群网络（VGG）架构[22]

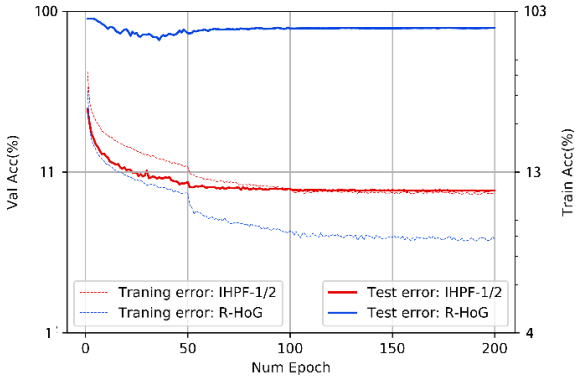
在深度神经网络中，以视觉几何群神经网络(VGG)网络为例，其结构如图 2。神经网络的主要结构可以分为特征提取部分（Features）和分类器（Classifier）部分。VGG 中的分类器部分由 3 个全连接层构成，特征提取部分由众多卷积层和池化层构成。这些卷积层的卷积核包含了特征提取过程中得到的关键信息，可能包含了图像的边缘信息，旋转翻转信息等等。在网络的输入数据方面，会对训练数据进行预处理，并且已经有很多常用的方法，像标准化（Normalization），随机裁剪（Random Resize），随机翻转（Random Reflip），白化（Whitening）等。这些都已经在现今的神经网络任务训练中得到了广泛的应用。

1.2 相关工作

在图像识别经典著作[20]中指出重现现实的视觉三维世界必须经过三个步骤，第一个过程就是获得目标主要的边缘信息。现在获取图片边缘特征信息的主要方法是频率梯度直方图（HOG）[12]，此方法已经在各个方面有了应用。像实时面部识别[14]和基于 HOG 的车辆检测[25]和基于 HOG-CNNs 的快速目标检测[25]。另一方面，李飞飞教授尝试获得图片语言的基本元素，构建了与频率梯度直方图相类似的颜色频率直方图[13]，以此来从图片中获得更多的信息。



(a) HOG 和 DST 处理单张图片时间对比



(b) HOG 和 DFT 训练过程对比

图 3：方向梯度直方图（HOG）与离散傅里叶变换（DFT）对比。左 3(a)采用 2160*2160 灰度图，右 3(b) DFT 采用理想滤波器滤波器（IHPF）

随着 HOG 在神经网路中得到广泛应用，本文尝试提出一个新的问题：提取图片中物体的边缘特征信息，有没有比 HOG 更效率的算法？

频率梯度直方图 传统的频率梯度直方图是选取方块（R-HOG）或者圆块（C-HOG）作为求梯度的单元[12]。将二维离散函数 $f(x,y)$ 通过 Gaussian 模糊之后，逐个像素基于模块单元中心方向计算梯度。梯度方向可以覆盖 0°-180°（无方向）或者 0°-360°（有方向）。HOG 的一个优点就是逐个像素计算了梯度，如果单元很小，那么很大限度上地保留了原图片中的梯度信息，但同时也消耗了大量地算力。如果单元很大，有失去了许多梯度信息。可能导致浪费算力或阻碍网络收敛。随着现在摄像设备的进步，1980*1080 的 FHD 全高清图像和 3840*2160 的 4K 超高清图像已经非常常见。以一张 2160*2160 的灰度图为例（图 3(a)），以 5 整数倍进行不同单元宽度和不同截止频率的 HOG 变换和本文即将使用的离散傅里叶变换 DFT。单元宽度为 5 个像素时求出全部梯度信息需要近 30 秒，这样的时间长度使得百万张图片的分类训练任务几乎不可能完成。虽然更大的单元宽度能将训练时间缩短到 10 秒左右，但越大的单元宽度会遗漏图片中越多的梯度信息，使得分类任务最终的效果大打折扣。

频率域学习 基于频率域的图像识别的主要思路是，图像可以看成特殊的二维波信号，每一点的灰度级就是这一点上的信号“强度”。根据信号的概念，频率是描述信号变化的快慢的物理量。这里就指图像空间里灰度变化的快慢程度，即图像的梯度。频率较大的对应图像中的“噪点”或者“目标边缘”。举例来讲，如果一张图片整体变化程度不大（比如单色图），则它在频率域下低频成分很多，而高频部分则很少。频域压缩表示包含丰富的图像理解任务模式，联合训练专用的基于自动编码器的网络进行压缩和推理任务[8][9][10]。从频域提取特征对图像进行分类[7]。提出了一种模型转换算法，将空间域神经网络模型转换为频域[11]。本文提出的方法不同的是基于离散傅里叶变换（DFT）。同时是对整张图片进行变换，这样避免了从空间到频域的复杂模型转换过程。因此，此方法具有更广阔的应用范围。

申明：在本文中的所有实验均是在 intel(R) Core(TM) i7-6700HQ CPU @2.6GHz，NVIDIA 960M CUDA GPU 4GB 上进行。所有图表均为实验结果，若引用其他出处的图表均在说明中标注引用文献。

2 算法

2.1 频率梯度直方图

在具体研究频率域图像处理之前，我们还将总体的将本文设计出的边缘特征提取算法（DFT）和已有的方法进行比较。就已有的方向梯度直方图（HOG）和广义 Haar 小波（Generalized Haar Warelet）、基于主成分分析的尺度不变特征变换（PCA-SIFT）、形状上下文算法（Shape Context）而言。在[12]中，Navneet Dalal 和 Bill Triggs 等人将圆形 HOG（C-HOG）和矩形 HOG（R-HOG）、圆形边缘 HOG（EC-HOG）与小波（Warelet），PCA-SIFT，梯度形状匹配（G-ShapeC），边缘形状匹配（E-ShapeC）进行在 MIT 和 INRIA 数据集分类任务上进行了对比。

MIT 数据集含有 924 张行人图片（64*128 像素），人物从肩到脚的距离约 80 象素，该数据集只含人物背面和正面两个不同的视角，仅有一个人物的图片集合。INRIA 数据集有训练集和测试集，训练集图片参数与 MIT 相同，但是测试集图片分辨率不确定。在各种实验情形下，得到结论基于 HOG 的方法正确率更高。所以在本文中，将结合不同参数下的三种高通滤波器下 DFT 边缘特征提取算法和已有的 HOG 算法进行对比。

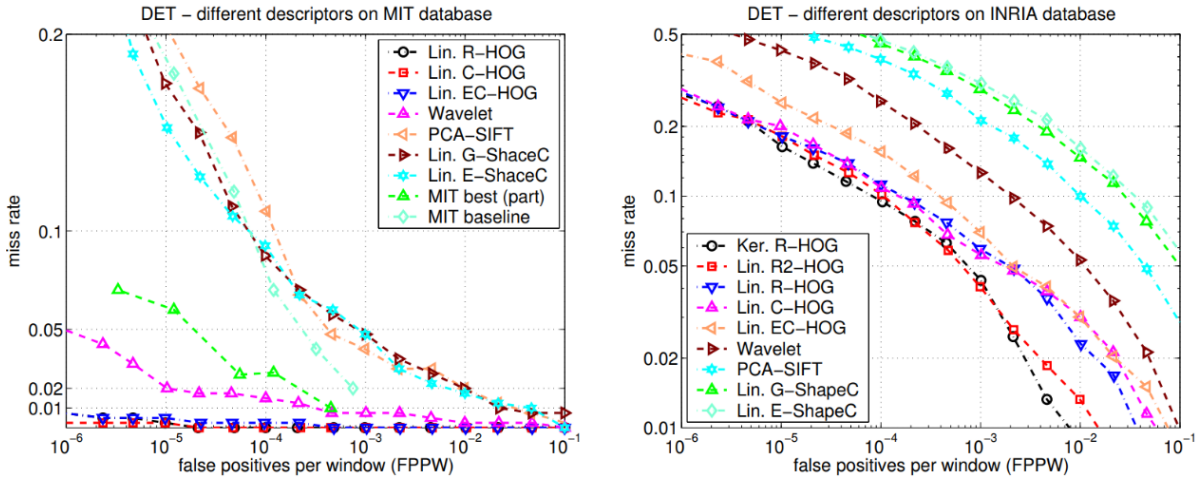


图 4[12]：各种特征描述器在 MIT 数据集 INRIA 数据集分类任务上的效果，False Positive Per Window（FPPW）是目标检测任务中常用的衡量指标。

矩形方向梯度直方图　以矩形方向梯度直方图（R-HOG）为例，图像局部目标的表象和形状能够被梯度或边缘的方向密度分布很好地描述。对图片在每个像素的各个方向上求出梯度的幅值，并且求出幅值最大时的角度。

(a)原图矩形网格单元　(b)示例单元，单元为宽度 20 像素的正方形　(c)最终效果图

图 5：矩形方向梯度直方图示例图-青蛙

然后将图片划分为方格状（见图 5(a)），每个网格叫做单元（Cell）。这里的单元理论上可以是任何形状，在[12]中对矩形和圆形进行了实验，最终圆形与矩形得到的结果差异并不大。对单元内的梯度幅值和角度进行统计。以其中一个单元为例（图 5(b)），幅值和角度数据见表 1 和表 2。

表 1：梯度幅值

0.0794	-0.1859	…	0.4115
-0.3059	-0.31	…	0.2309
…	…	…	…
0.0562	0.2127	…	0

表 2：梯度角度

129.2	149.6	…	90
176.7	159.8	…	90
…	…	…	…
0	0	…	0

最后根据每个单元中的梯度直方图求出每个单元内的梯度向量（见图(6)），在这之前可以在不

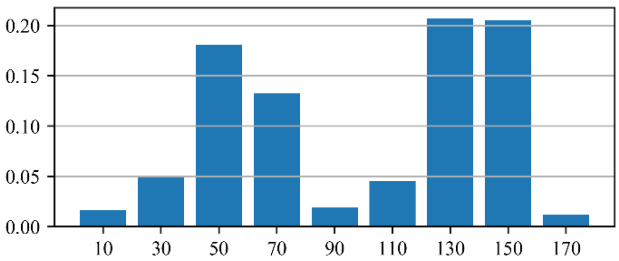


图 6：HOG 频率梯直方图，角度类别（Bin size）为 9

同尺寸内进行归一化，示例中使用的是在整张图片范围内进行归一化。具体的效果图见示意图 5(c)。

结果 本次使用的是表格 3 中 Mini-VGG 网络的架构，进行 Fashion-MINIST 数据集分类任务训练，并且与本文中设计的 DFT 算法进行对比。在数据预处理上除了 R-HOG 和使用截止频率为图片尺寸一半的理想滤波器的 DFT 算法之外，和在训练进行相同的标准化处理。从图 3(b)中的训练结果可以看出，即使 R-HOG 在训练集上错误率很低，也无在测试集上的取得较低的错误率。最低的错误率也达到了 66.16%。相比之下，对于 DFT 算法在更早的时候收敛到了较低的错误率，而且在训练集和测试集上同时表现很好。另外，DFT 算法训练的时间仅有 4.98 小时，而 R-HOG 更是有近乎三倍的 14.28 小时。

2.2 频率域图像处理

Uber 人工智能实验室发表的[7]中，提出了采用离散余弦变换（DCT）系数作为输入的方法，从频域提取特征对图像进行分类。基于静态信息通道选择的频率域学习方法在图像分类任务上能获得更高的精度，并且进一步缩小了输入数据的大小。并且，设计了一种针对图像格式为 JPEG，色彩空间为 YCbCr 的图像处理算法‘`libjpeg`’。结合修改后的 ResNet-50 网络，最终在 ImageNet 数据集分类任务上取得了低于通常 7.3%的错误率达到了 6.98%的错误率和 220 秒左右的单张图片训练时间。在连续信号的情形下，连续傅里叶正弦变换 Ω 和逆变换 Ω^{-1} 的定义如下：令 $f(x_1,x_2)$，其中 $x_1,x_2\in\mathbb{R}^1$，f 是连续函数。则有二维傅里叶变换表达式 1：

$$\begin{aligned}\Omega(f(x_1,x_2))&=F(y_1,y_2)=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}f(x_1,x_2)\exp[-2\pi i(y_1x_1+y_2x_2)]\mathrm{d}x_1\mathrm{d}x_2\\ \Omega^{-1}(F(y_1,y_2))&=f(x_1,x_2)=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}F(y_1,y_2)\exp[2\pi i(x_1y_1+x_2y_2)]\mathrm{d}y_1\mathrm{d}y_2\end{aligned}\tag{1}$$

傅里叶余弦变换算（DCT）法与傅里叶变换（DFT）不同的是，其前者是将后者对小波进行出原点意外的偶延拓，再对其进行傅里叶变换。在 $(x,y),(u,v)\neq(0,0)$ 相应的离散余弦变换（DCT）公式为下式：

$$\begin{cases}F(u,v)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)\cos\left[\frac{\pi}{M}u\left(x+\frac{1}{2}\right)\right]\cos\left[\frac{\pi}{N}v\left(y+\frac{1}{2}\right)\right] & (2)\\ f(x,y)=\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}F(u,v)\cos\left[\frac{\pi}{M}x\left(u+\frac{1}{2}\right)\right]\cos\left[\frac{\pi}{N}y\left(v+\frac{1}{2}\right)\right] & (3)\end{cases}$$

当 $(x,y),(u,v)=(0,0)$ 时

$$\begin{cases}F(u,v)=\frac{1}{2}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)\cos\left[\frac{\pi}{M}u\left(x+\frac{1}{2}\right)\right]\cos\left[\frac{\pi}{N}v\left(y+\frac{1}{2}\right)\right] & (4)\\ f(x,y)=\frac{1}{2MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}F(u,v)\cos\left[\frac{\pi}{M}x\left(u+\frac{1}{2}\right)\right]\cos\left[\frac{\pi}{N}y\left(v+\frac{1}{2}\right)\right] & (5)\end{cases}$$

为了保留原图中的更多信息，本文选择采用离散傅里叶变换（DFT）作为原图域频率域之间的转化公式。与连续情形下相同，同样有卷积定理 9 等性质。

以通常的 RGB 通道数据为研究对象，记 RGB 通道中的任意一通道高度为 N，宽度为 M 的信息 $f(x,y)$，x 和 y 是代表像素点，是离散实变量，u 和 v 为离散频率域变量。离散傅里叶变换和逆变换见式子 6、7:

$$\begin{cases} F(u,v)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\exp\left[-2\pi i\left(\frac{xu}{M}+\frac{yv}{N}\right)\right] & (6) \\ f(x,y)=\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} F(u,v)\exp\left[2\pi i\left(\frac{ux}{M}+\frac{vy}{N}\right)\right] & (7) \end{cases}$$

二维卷积表达式：

$$f(x,y)*h(x,y)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(s,t)h(x-s,y-t) \tag{8}$$

其中， $x=1,2,\cdots,M-1, y=1,2,\cdots,N-1$。二维卷积定理[6]如下式 9：

$$\begin{aligned} f(x,y)*h(x,y) &\Leftrightarrow F(u,v)H(u,v) \\ f(x,y)h(x,y) &\Leftrightarrow F(u,v)*H(u,v) \end{aligned} \tag{9}$$

与离散余弦变换不同的是，正弦变换得到的是复数。因为离散傅里叶变换后得函数具有良好的对称性质，那么用极坐标表示频率域函数 $F(u,v)\in\mathbb{C}$ 可以让数据处理起来更加方便。

$$F(u,v)=\left|F(u,v)\right|\exp\left[(\Phi(u,v)i)\right] \tag{10}$$

其中：

$$\left|F(u,v)\right|=\sqrt{\left[R^2(u,v)+I^2(u,v)\right]}$$
$$\Phi(u,v)=\arctan\left[\frac{I(u,v)}{R(u,v)}\right]$$

幅度 $\left|F(u,v)\right|$ 称为频谱 ρ， $\Phi(u,v)$ 为相角 ϕ。
由式子 6 和 7 可得在原点处有：

$$F(0,0)=MN\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)=MN\,\overline{f(x,y)} \tag{11}$$

就有 $F(0,0)=MN\left|\overline{f(x,y)}\right|$，$\overline{f(x,y)}$ 为图像的平均灰度。一张图像经过傅里叶变换后的频率域信息中，相位角 ϕ 支配着图像的形状，频谱信息 ρ 主导着图像的灰度信息[5]。

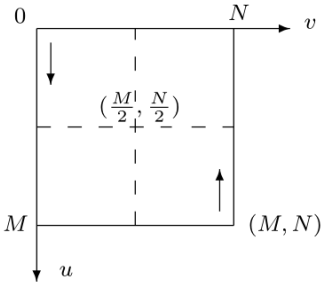
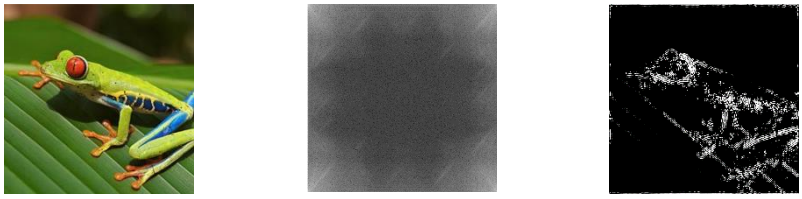


图 7：对称示意图

在单张图片所在的区域 $[0,N]\times[0,M]$ 上，因为频谱 ρ 具有周期性 $\rho(u,v)=\rho(u+sM,v+tN),(s,t\in\mathbb{Z})$ 和共轭对称性 $\rho(u,v)=\rho(-u,-v)$，从而 $\rho(u,v)=\rho(M-u,N-v)$，即 ρ 在单张图片内是关于图片中心 $(M/2,N/2)$ 对称的。



(a)示例图 620*620　　(b)离散傅里叶变换后　　(c)最终效果

图 8：示例图，最终效果图总处理时间为 0.1028 秒

在[6]中指出，$F(0,0)$与图片的尺寸对应着原图像的平均灰度$\overline{f(x,y)}$。得到的频率域图像中低频部分对应着原图像的缓慢变化的分量，也就是图像中比较平坦的区域，如图 8(b)所示。高频部分对应的就是原图像中变化较快的部分，也就是原图中的噪点和边缘特征信息。频率域图像处理就是指对原图的频率域图像进行修改，再逆变换回去的过程，通常表达为式子 12 和 13。

$$G(u,v)=F(u,v)H(u,v) \tag{12}$$

$$g(u,v)=F^{-1}(G(u,v)) \tag{13}$$

$H(u,v)$为转移函数或者是滤波函数，$G(u,v)$为对$F(u,v)$进行滤波后的输出结果，H和F的相乘定义为对应元素相乘，F^{-1}为傅里叶逆变换，$g(x,y)$为最终输出图像。以上过程简要描述为如图 10。图 8(c)为经过离散正弦变换后的效果图，滤波器$H(u,v)$采用理想高通滤波器（IHPF）具体见式子 15。

相比之下频率梯度直方图（HOG）[12]的思路是通过对图片分块，在不同方向求像素之间的梯度，得到在每个块内的梯度在 360 度（180 度）范围内的分布和强度信息，获得整张图片在不同角度的梯度信息。本文提出的方法目的通过离散傅里叶变换，对图片在频率层面进行处理。利用二位离散傅里叶变换的性质，可以处理在空间域上不便于处理的问题。此外，基于离散傅里叶变换的方式相较于 HOG 通常运算速度更快，使模型更容易训练，提高训练效率。另一方面，除了离散傅里叶变换有相应的快速算法之外，根据卷积定理式子 9 也可以将 DFT 处理特征提取视为不同的卷积层，也可以在频率域层面增加更多的处理，为以后的拓展打下了基础。

根据示意图 10，图像频率域滤波处理是通过修改图像的傅里叶变换后的数据再计算逆变换得到处理后的图像。对于一张单通道，大小为$M\times N$的数字图$f(x,y)$，基本的滤波可以表达为式 14。

$$g(x,y)=\Omega^{-1}(H(u,v)F(u,v)) \tag{14}$$

其中$F(x,y)=\Omega(f(x,y))$，$H(u,v)$是滤波函数，$g(x,y)$是输出图像，大小与原图像相同。

在提取物体边缘特征信息时候，要从频率信息中过滤出原图中的高频信息，故使用高通滤波器[6]。本文中主要考虑理想滤波器（Ideal Filter）图 9(a)，布特沃斯滤波器（Butterworth Filter）图 9(b)和高斯滤波器（Gauss Filter）图 9(c)，这三种滤波器的应用[3][4]。以下是 3 种滤波器高频情形下的的表达式：

1. 理想高通滤波器 IHPF

$$H(u,v)=1-\lambda_{D_0}(u,v) \tag{15}$$

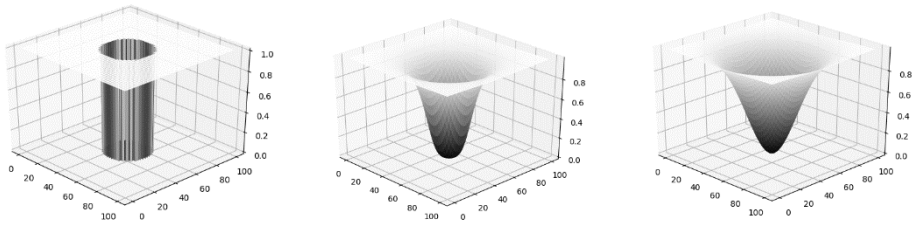
2. 布特沃斯高通滤波器 BHPF

$$H(u,v)=\frac{1}{1+\left[\frac{D(u,v)}{D_0}\right]^{2n}} \tag{16}$$

3. 高斯高通滤波器 GHPF

$$H(u,v)=1-\exp\left\{-\frac{D^2(u,v)}{2D_0^2}\right\} \tag{17}$$

$\lambda_{D_0}(u,v)$ 是区域 $\{(u,v)\in\mathbb{C}^2\,|D(u,v)\le D_0\}$ 的特征函数，D_0 为截止频率，$D(u,v)$ 为频率域的中心点 (u,v) 到频率域原点的距离。



(a) IHPF D_0=180　　(b) BHPF D_0=180 , $n=2$　　(c) GHPF D_0=180

图 9：不同的滤波器

在本文中，给出 DFT 的一些简单应用，特征提取上选择典型的三种滤波器 IHPF，BHPF，GHPF。系统地讨论各个参数下的离散傅里叶变换在经典卷积神经网络上的表现。在不包含前处理和后处理的情形下，图像的灰度层进行全尺寸的离散傅里叶变换，后进行在频率域的滤波处理。最后进行逆变换。以 Fashion-MINIST 数据集中的灰度图为例，原图和在不同的高通滤波器下的效果图见图 11。

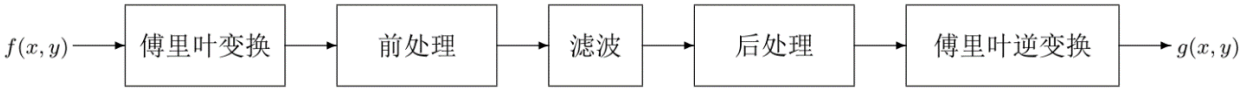


图 10：提取边缘信息步骤示意图

通过上文中对方向梯度直方图和频率域图像处理算法实现过程发现，在三种已有的滤波器上可选择的参数有截止频率 D_0 和幂指数 n（布特沃斯滤波器的阶数）。更多的，对于色彩空间为 RGB 三通道的彩色图片，初步的设计就是分别对 R 通道、G 通道、B 通道应用相同算法，最终依次叠加构成三通道图片，效果见图 11。选取 CIFAR-10[1]数据集中的图片演示。

图 11：IHPF，BHPF，GHPF 在图例中的效果。从左往右依次为 Fashion-MINIST 图例原图（28*28）、IHPF-1/8、BHPF-1/8-1、GHPF-1/8 效果图。CIFAR10 图例原图（32*32）、IHPF-1/8、BHPF-1/8-1、GHPF-1/8 效果图

实现离散傅里叶变化变换特征提取算法和 IHPF、GHPF、BHPF 的核心代码见附录 1、4

3 实验

3.1 数据集

本文在两个不同的数据集下进行实验。第一个是由 Zalando 公司旗下的研究部门提供[2]的 Fashion-MINIST 数据集，其涵盖了共 7 万个不同衣物商品的正视图，一共 10 种不同类别的。Fashion-MNIST 数据集的大小、格式和训练集与测试集划分与原始的 MNIST 完全一致，均为 60000 张图片构成训练集和 10000 测试集，每张图片为 28*28 像素的灰度图片。每张图片都是商品的标准正视图，不包括任何的旋转，侧视等更多的信息。另一个是更具有挑战性的‘CIFAR-10’数据集[1]，数据集包含

10 个类别的 60000 张 32*32 彩色图像，每个类别 6000 图 12 展示了每个类别的一些例子，有 50000 张训练图像和 10000 张测试图像。并且上述两种数据集中，这些类别之间是完全互斥的，也就是一张图片中不可能既含有飞机也含有卡车。同时也不会包含图像的翻转、裁剪、降升采样等预处理。

图 12：CIFAR-10 示例图。每张彩色图片为横纵 32 像素，数据集中共包含了 10 个不同类别的物体

3.2 卷积神经网络模型

网络架构 视觉几何群网络（Visual Geometry Group Network）VGG 虽然在 2014 年 LSVRC 上获得了第二名，第一名是谷歌网络（GoogleNet）。但是在不同任务迁移学习的表现上，VGG 的效果优于 GoogleNet，而且 VGG 常作为研究特征提取算法的实验网络。

VGG 网络的核心就是卷积层，以输入数据为 $M\times N$ 的 RGB3 通道彩色图片为例，定义函数 $\Im$，不同类别的层定义的函数也不同。输入图片 $x^i_{stq}=\{x^i_{stq}\}$ 为一个尺寸 $(M,N,3)$ 的整数矩阵，其中 $s=1,2,\cdots,M$， $t=1,2,\cdots,N$， $q=1,2,3$。接受域（Receptive Field） $R=R_{stq}$ 定义为一个深度与 x^i 相同，横纵宽度分别为 d_{rM} 和 d_{rN} 的矩阵，即 $(d_{rM},d_{rN},3)$， $s,t=1,2,\cdots,d_r$， $q=1,2,3$， R_{stq} 为任意实数。步长 d_s 定义为一个非负整数。在这里集合 $\{x^i_{stq}\,|A\le s\le A+a-1,B\le t\le B+b-1,C\le q\le C+c-1\}$ 表示为 $x^i[A:a,B:b,C:c]$。输出矩阵是一个大小为 $K\times T$ 的实矩阵，记作 I。那么

$$I_{1+k,1+t}=\Im(x^i[1+kd_s:d_{rM},1+kd_s:d_{rN},1:3],R)\quad k=1,2,\cdots,K,t=1,2,\cdots,T$$

其中

$$K=\frac{M-d_{rM}+1}{d_s}$$
$$T=\frac{N-d_{rN}+1}{d_s}$$

更多的，在实际操作中为了避免输出尺寸减小，会在图片周围进行 0 元素填充，填充的宽度分别为 d_{pM} 和 d_{pN}。那么整数

$$K=\frac{M+2d_{pM}-d_{rM}+1}{d_s}$$
$$T=\frac{N+2d_{pN}-d_{rN}+1}{d_s}$$

在实际情形中，卷积层通常会有不止一个的接受域，每个接受域之间是相互独立的，但大小是相同的。将不同的接受域下计算出的输出矩阵 V 进行堆叠从而形成三维矩阵，接受域的个数 D 就是三维矩阵的深度 D。综上所述可得输出矩阵的大小为 (K,T,D)。

在卷积层（Convolutional Layer）中，函数 $\Im(A,B)$ 为点积函数， A 和 B 为尺寸同为 (S,T,Q) 的三维矩阵。函数 $\Im$ 定义为:

$$\Im(A,B)=\sum_{q=1}^{Q}\sum_{t=1}^{T}\sum_{s=1}^{S}A_{stq}B_{stq}$$

在最大池化层（MaxPooling Layer）中，步长 d_s 的大小与接受域的宽度对应相等。接受域中的元素全部为固定值 1。 A 和 B 大小同上， B 中的元素全为 1，函数 $\Im(A,B)$ 定义为

$$\Im(A,B)=\max_{A_{stq}\in A}\{A_{stq}B_{stq}\}$$

全连接层（Fully Connected Layer）的定义与矩阵的乘法相同，将传入数据排成一个一维向量与矩阵相乘。与传入数据相乘的矩阵成为权重矩阵，全连接层中的权重矩阵数值为随机数值，并且在训练中都可以得到学习。随机丢弃层（Dropout Layer）[31]是在全连接层上加以改进后的层级，目的是为了防止在网络在训练的过程中发生过拟合的情况。在初始化时，与全连接层不同的是，矩阵初始化时会以一定的概率 p 将权重矩阵中的值固定为 0。而其他的值为随机实数并且可以得到学习。在卷积层，最大池化层之后，都会添加一个非线性函数层对输出的每个元素进行处理。这样的函数有很多，现在还广泛使用的是线性整流函数（ReLU），它的定义如下 18 式：输入矩阵 $A=\{A_{stq}\}$ 为一个三维矩阵

$$\mathrm{Re\,LU}(A)=\max_{A_{stq}\in A}\{0,A_{stq}\} \tag{18}$$

在本文中所使用的模型中，将 VGG 针对 CIFAR10 和 Fashion-MINIST 这种小型数据集进行了适当的简化得到 Small-VGG 神经网络模型。相比原有的 VGG 网络，Mini-VGG 网络的分类器层数和卷积层的数量更少，但是保留了原有网络的各种层级关系，其架构见表格 3。

表 3：在 Fashion-MINIST 和 CIFAR10 数据集分类任务的网络架构

<table>
<tr><td colspan="9">Mini-VGG 架构</td></tr>
<tr><td>Fashion-MINIST</td><td>Conv5-32</td><td>Conv-5-64</td><td rowspan="2">Max-Pool2</td><td>Conv-5-128</td><td rowspan="2">Max-Pool2</td><td rowspan="2">DropOut0.5</td><td rowspan="2">FC512</td><td rowspan="2">FC10</td></tr>
<tr><td>CIFAR10</td><td>Conv-1-32 + Conv-5-32</td><td>Conv-5-32</td><td>Conv-5-128</td></tr>
</table>

其中 Conv<接受域宽度>-<输出层数> 步长=2，MaxPool<步长=2>，Dropout<概率=0.5 每个层都以 ReLU 结尾而 Mini-VGG 在作为这样较小的数据集上的训练任务，极大的减少了深度神经网络的计算量和网络的参数量。模型架构核心代码见附录 2.

损失函数 为了衡量模型在测试集合上的预测结果，需要判定实际的输出与期望的输出的接近程度。将输出结果视为概率分布，引用信息论中的交叉熵概念（Cross Entropy），以用于度量两个概率分布间的差异性信息。并在此基础之上加以改善，设计出的损失函数。

假设算法期望分布为 p 而输出分布为 q，那么根据定义。衡量 p 与 q 二者的交叉熵 $H(p,q)$ 为

$$-\sum\nolimits_x[p(x)\ln q(x)+(1-q(x))\ln(1-q(x))]$$

但是在网络模型中强调的更多是在预测正确的方面，从而只考虑 $p(x)=1$ 的情况，即 $\ln(q(x))$。Softmax 函数作为求模型训练结果预测准确的概率，这个概率条件概率，它的计算公式为 19

$$P(Y=class\mid X=x_i)=\frac{e^{S_{class}}}{\sum_j e^{S_j}} \tag{19}$$

其中 $X=\{x_i\}$ 为训练集或者测试集， x_i 在网络中的输出为 S，预测结果为 Y， x_i 所属类别为 $class$。S 和 Y 都是一个长度为数据集类别的数列。

因为自然对数函数在这里不改变原函数的单调性，而且更好的抑制了数据过大的情况。故将

Softmax 函数的定义和交叉熵的概念得到

$$\ln[P(Y=class \mid X=x_i)]=\ln\left[\frac{e^{S_{class}}}{\sum_j e^{S_j}}\right]$$

又因为在模型中，损失函数通常是越小越好。那么添加负号后得到了最终的损失函数交叉熵损失函数（Cross Entropy Loss）具体表达式为 20

$$loss(S,class)=-\ln\left[\frac{e^{S_{class}}}{\sum_j e^{S_j}}\right]=-S_{class}+\ln\left(\sum_j e^{S_j}\right) \tag{20}$$

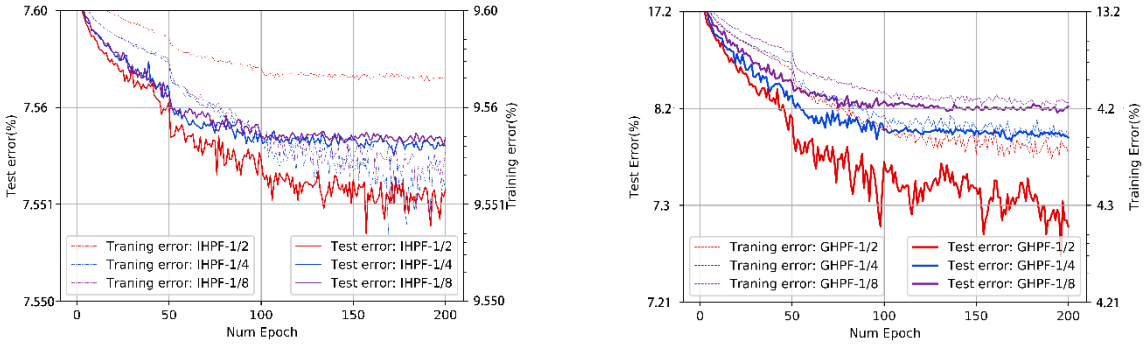
其中 x 指的是图片，$class$ 指的是类别编号。

4 结果演示

本文中实验主要在 Fashion-MINIST 数据集上和 CIFAR10 数据集上进行，而滤波器的选择上，主要考虑的参数为 D_0 占图片尺寸的比例 k。特别的，在布特沃斯滤波器（BHPF）中，还有阶数 n，这些参数都属于超参数（Hyperparameter）。为了确保尽可能准确地评估它们之间地差异，在模型数据预处理层中仅仅加入了离散傅里叶变换数据处理和标准化（Batch Normalization）。在实验中所有滤波器都有 50% 的可能性不产生作用，而测试集合的图片全部保持原始状态不变，引入一定的随机性是为了避免在实验结果中出现过拟合的情况，进而降低实验效果。

4.1 Fashion-MINIST 数据集分类任务

图像分类的任务是对给定的图像进行类别的区分。此小节中分析的是选择 Fashion-MINIST 是 CIFAR-10 作为数据集训练分类的结果。其中每个数据集都包含了 10 类别的物体照片，每个类别的训练数据集均超过 1 万张。IHPF、BHPF、GHPF 在 $k=\{1/2,1/4,1/8\}$ 和 $n=\{1,2,4\}$ 下，Fashion-MINIST 数据集为代表的灰度图上的效果。



(a) IHPF-FashionMINIST 实验　　(b) GHPF-FashionMINIST 实验

图 13：IHPF-FashionMINIST 实验左 13(a)，GHPF-FashionMINIST 实验右 13(b)，比例系数 $k=\{1/2,1/4,1/8\}$，在 Mini-VGG 模型下训练 200 次，损失函数为 Cross Entropy Loss，见式 20，优化算法为随机梯度下降法(SGD)，初始学习率为 10^{-3}，初始动量(momentum) 为 0，训练\测试函数代码实现见附录 3。

实验一（理想高通滤波器 IHPF）使用 IHPF 对 Fashion-MINIST 分类任务评估了在不同的 k 值下训练的结果，具体结果如图 13(a)和表格 5 所示。在 $k=1/4$ 和 $k=1/8$ 时，二者相差较小，但后者达到几乎相同的结果却多花费了 5000 秒以上的时间。IHPF-1/4 的最优结果比 IHPF-1/8 的正确率高出 0.08% 同时也花费了更少的时间。对于 IHPF-1/2，虽然在 Fashion-MINIST 数据集上取得了三个中最好的效果，在测试集合上的正确率达到了 92.4%， 但是也花费了最长的时间。综合所有结果，$k=1/4$ 是

效率是三者中最高的。

表 4：实验中每个滤波器的参数

IHPF	BHPF	GHPF
$k=1/2$	$n=1,k=1/2$	$k=1/2$
$k=1/4$	$n=2,k=1/4$	$k=1/4$
$k=1/8$	$n=4,k=1/8$	$k=1/8$

表 5：实验一训练时间（秒）

IHPF	训练	测试
$k=1/2$	4.74	0.24
$k=1/4$	3.30	0.20
$k=1/8$	4.17	0.23

表 6：R-HOG 和 IHPF-1/2 训练时间

	R-HOG	IHPF-1/2
时间(h)	14.28	4.98
第一(%)	66.16	8.11
第二(%)	69.15	8.15

实验二（高斯高通滤波器 GHPF）对于 GHPF 在 Fashion-MINIST 分类任务中的结果。当 $k=1/4$ 和 1/8 时，最终错误率降低到了 8.2%左右，其中前者略低于后者（约小于 1%），且二者训练的时间相

表 7：实验二训练时间（小时）

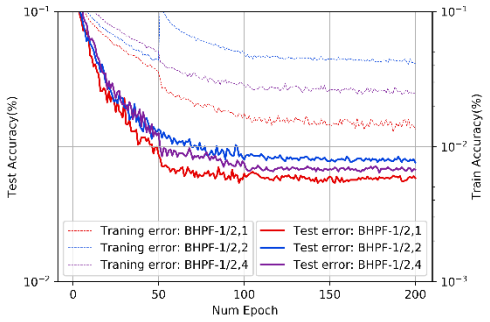
总时间		k		
		1/2	1/4	1/8
n	1	4.80	3.47	3.77
	2	4.61	5.02	4.02
	4	3.46	4.08	5.09

表 8：实验二 第一低/第三低 错误率（小时）

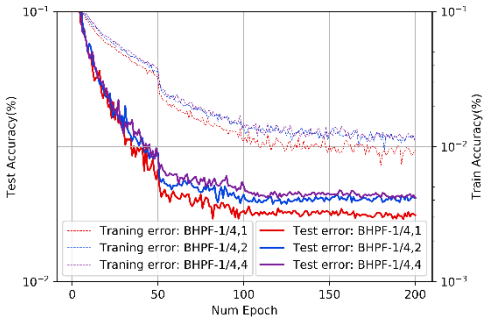
		k		
		1/2	1/4	1/8
n	1	7.32/7.35	7.69/7.77	7.76/7.78
	2	7.76/7.78	7.94/7.96	7.85/7.87
	4	7.53/7.56	8.03/8.05	8.06/8.08

差不大（1523.79 秒）。但是当 $k=1/2$ 时候，此时 GHPF 滤波器下的神经网络收敛最快，而且效果最好，花费的时间是三者之中最少的，更比 IHPF-1/4 快了 300 秒同时还达到了更优的效果。综合所有结果，$k=1/2$ 是效率是三者中最高的。

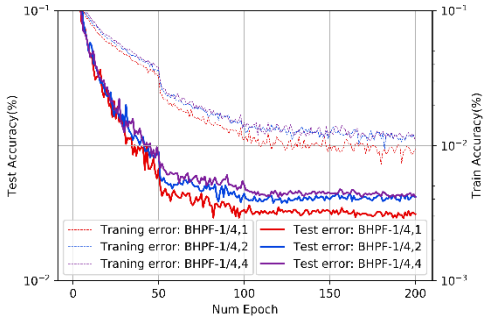
实验三（布特沃斯高通滤波器 BHPF）考虑 BHPF 中的参数 k 和阶数 n，这里记作 BHPF-k,n。本次实验中进行了九次训练，综合图 14(a)、14(b)、14(c)，和表格 7。可以看到，k 相同的时候时候，都



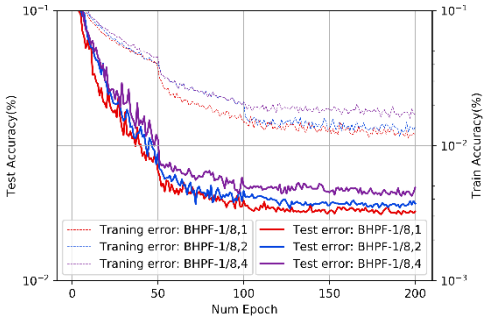
(a) BHPF-1/2 -Fashion-MINIST 实验



(b) BHPF-1/4 -Fashion-MINIST 实验



(c) BHPF-1/8 -Fashion-MINIST 实验



(d) BHPF-1 -Fashion-MINIST 实验

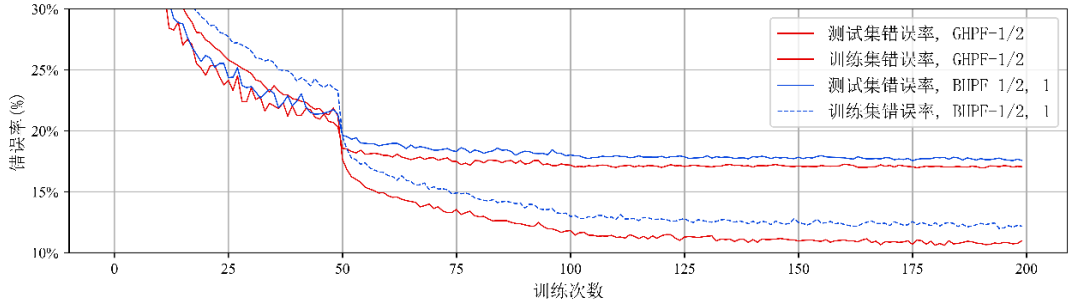
图 14：比例系数 $k=\{1/2,1/4,1/8\}$，幂指数 $n=\{1,2,4\}$，在 Mini-VGG 模型下训练 200 次，损失函数为 Cross Entropy Loss，见式 20，优化算法为随机梯度下降法(SGD)，初始动量(Momentum) 为 0.9，初始学习率为 10^{-3}，每当损失达到瓶颈时除以 10，训练\测试函数代码实现见附录 3。

是 $n=4$ 时候效果最好，但是每次训练错误率最低的情形却不一定出现在 $n=4$ 时。从表格 8 可以看出，错误率最低的是 BHPF-1/2,1，其次是 BHPF-1/2,2 和 BHPF-1/8,1。从表格 7 中可以知道，为了达到较低的错误率，BHPF-1/2,1 的总训练时间为 17274.19 秒，仅仅比最高的 18082.30 秒低 808.11 秒。而且，当 $k\times n=1/2$ 时，训练时间是相同 n 或 k 中训练时间最长的。而 BHPF-1/8,1 却只使用了 13561.20 秒。总的来看，综合错误率和训练效率，结合图 14(d)，虽然阶数 n 越高，训练收敛速度越快，整体效果越好，但是达到最优情形时，最有效率的是当 $k=1/8$，阶数 $n=1$ 时候的训练效果是这组训练中效果较好的。

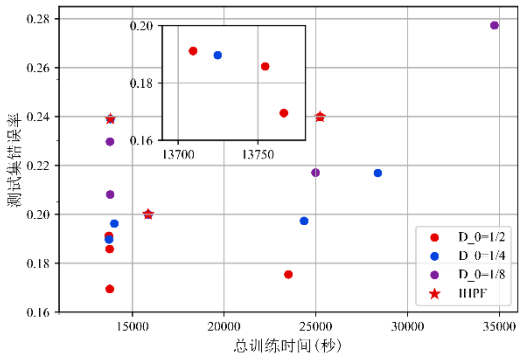
4.2 CIFAR10 数据集分类任务

相较于数据量更大的数据集 CIFAR-10，进行与 Fashion-MINIST 数据集相同的训练。图 15 总结了参数 k 和 D_0 对于各种滤波器效率的影响。从图中可以看出，两个参数具有明显的超参数特征，没有明显的规律显得较为凌乱。所以在有限的算力范围内进行了尝试后，GHPF-1/2 的效率最高。而且，从图 15(b)中看出，当截止频率过低时相比之下会明显提高错误率，并且大幅增加训练时间。与 k 相比，n 对 BHPF 的影响不如前者明显，图 15(c)中红圈内容显示出，在三种指数情况下，都存在训练时间较低的情况。

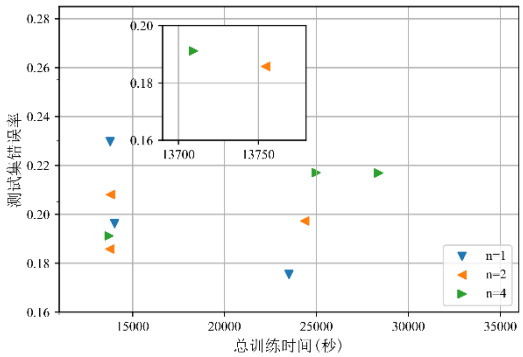
对于在此训练得到结果中最优的 GHPF 滤波器，训练过程中的错误率具体情况如图 15(a)。大约在第 100-125 训练，训练时间 4351.74 秒时达到瓶颈，并且训练集和测试集的错误率差值稳定在 6% 左右。作为对比，选取图 15(b)中的 BHPF-1/2,1 为例。最终错误率略高于前者（约 1%）从图 15(a)可以看出同样在约第 100-125 次训练时达到瓶颈，测试集错误率和训练集错误率都高于 GHPF 滤波器。而且，达到瓶颈所花费的时间为 6831.42 秒。



(a) GHPF-1/2 和 BHPF-1/2,1 在 CIFAR10 数据集分类任务的训练过程。在 Mini-VGG 下训练 200 次，存实函数为 Cross Entropy Loss，见式子 20。优化算法为随机梯度下降法（SGD），初始动量（Momentum）为 0.9，学习率为 10^{-3}，每当损失达到瓶颈时除以 10，训练\测试函数代码实现见附录 3。



(b) D_0 为 CIFAR10 任务训练效率的影响



(c) n 对 CIFAR10 任务训练效率的影响

图 15：数据集：CIFAR10，比例系数 $k=\{1/2,1/4,1/8\}$，幂指数 $n=\{1,2,4\}$

5 算法的讨论与发展前景

讨论 综合三者，基于离散傅里叶变换的图像特征提取算法与方向梯度直方图算法在处理超高分辨率的图片上有着明显优势，在处理的效果上与其相比也有较大的进步。

理想高通滤波器对频率的处理方式简单明了但是过于单一，完全抹去一部分和保留一部分频率域信息。这会也许导致损失了频率域中存在于低频中，但是却很有用的信息。从而到最后得到的特征信息损失了很多有用的信息。

在高斯高通滤波器中，结合式子 17 可以看出在过滤高频信息的过程中，与理想高通滤波器不同的对高频部分做出来一定的保留，另外还在低频与高频之间有一个平滑的过渡，对截止频率 D_0 之外的频率域部分也进行了适当的处理，这样就避免了滤波器的效果过于依赖截止频率 D_0 的设定。图 13(a)中，当 $k=1/2$ 时的理想高通滤波器虽然是三者之中效果最好的，但是与同样 $k=1/2$ 时的高斯高通滤波器相比在训练集错误率高了近 3%。

更多的，布特沃斯高通滤波器与理想高通滤波器相比，提供了更多的方案。引入了滤波器的阶数 n 这一参数。从图 9(b)中可以看到，阶数 n 越高时越接近理想高通滤波器，越低时越接近高斯高通滤波器。这种情形下，模型需要更多的调试才可以进行最终的分类任务训练，通常可以得到比前两种更好的效果，但同时也会在前期准备之中花费更多的时间。

综合本文对图像中物体边缘特征提取算法的探索与研究，建立了基于离散傅里叶变换的特征提取算法基础。该算法在频率域对图像进行频率域处理，从卷积定理[6]的角度看，频率域处理中滤波函数与频率域函数相乘的处理方式，实际等价于在原图像空间域滤波函数进行卷积。这不仅体现了频率域处理的特殊性，还反映了在图像处理上方式的多样性。在与 HOG 进行实验对比中也发现在频率域处理图像在速度上、精度上的显著进步。

发展前景 在现今的虚拟现实（Virtual Reality）和自动驾驶领域、混合现实等领域，图像识别是非常重要的一个部分。在往后的发展上，随着摄像设备技术的发展，得到的图片数据分辨率大大提升。在未来，随着第五代通信网络（5G）的普及，高清数据流媒体将成为主流。算法的设计目的更需要的是使用尽可能少的内存消耗来实现有效利用算力。

在方向梯度直方图[12]中为了实现较好的效率，对图像进行了降采样（Down Sampling），进而损失了高分辨率带来的更多的重要信息。而基于离散傅里叶变换的特征提取算法与方向梯度直方图算法相比之下优势之一就是处理速度更快，这也就意味着可以从更高分辨率的图片中获取物体边缘信息。从而使神经网络对物体的检测，对图片的理解更加准确。在未来，频率域图像处理有着更大的发展空间。

附录

附录 1 DFT 图像边缘特征提取算法模块

```
class RandomFourier():
    def __init__(self,filter):
        self.filter = filter
    def __call__(self,pic):
        n = random.randint(0, 1)
        if n==0:
            return pic
        if n==1:
            img = np.array(pic)
            fshift = np.fft.fftshift(np.fft.fft2(img))
            fshift *= filter
            iimg = np.fft.ifft2(np.fft.ifftshift(fshift))
            iimg = np.abs(iimg).astype(np.float32)
            return iimg
    def __repr__(self):
        return self.__class__.__name__ + f' {self.filter.shape}'
```

附录 2 Mini-VGG 模型

```
class Mini-VGG(nn.Module):
    def __init__(self):
        super(Mini-VGG, self).__init__()
        self.features = nn.Sequential(
            nn.Conv2d(1, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2)),
            nn.ReLU(inplace=True),
            nn.Conv2d(32, 64, kernel_size=(5, 5), stride=(1, 1)),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=2, stride=2, padding=0),
            nn.Conv2d(64, 128, kernel_size=(5, 5), stride=(1, 1)),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=2, stride=2, padding=0)
            )
        self.classifier = nn.Sequential(
            nn.Dropout(p=0.5, inplace=False),
            nn.Linear(4*4*128, 512),
            nn.ReLU(inplace=True),
            nn.Linear(512, 10)
        )
    def forward(self, x):
        x = self.features(x)
        x = x.view(-1, 4*4*128)
        x = self.classifier(x)
        return x
```

附录 3 测试\训练函数

```
def train_model(model, dataloaders, loss_fn, optimizer, num_epochs=5):
    best_model_wts = copy.deepcopy(model.state_dict())
    best_acc = 0.
    val_acc_history = [] # Creat a list contrains accuracy history data.
    val_loss_history = [] # Creat a list contrains training loss history data.
    val_error = []
    lr_history = [] # learning rate history
    train_acc_history = []
    train_loss_history = []
    train_error_history = []
    lr = optimizer.param_groups[0]['lr']
    point = 0
    step_point = 0
    total_time = []
    for epoch in range(num_epochs):
        for phase in ["train", "val"]:
            running_loss = 0.
            running_corrects = 0.

            time_s = time.time()
            if phase == "train":
                model.train()
            else:
                model.eval()
            for inputs, labels in dataloaders[phase]:
                inputs, labels = inputs.to(device), labels.to(device)

                with torch.autograd.set_grad_enabled(phase=="train"): # phase=="train"
is bool type variable.
                    outputs = model(inputs) # bsize * 2
                    loss = loss_fn(outputs, labels)

                preds = outputs.argmax(dim=1)

                if phase == "train":
                    optimizer.zero_grad()
                    loss.backward()
                    optimizer.step()
                running_loss += loss.item() * inputs.size(0)
                running_corrects += torch.sum(preds.view(-1) == labels.view(-1)).item()

            time_e = time.time()
```

```
            time_phase = time_e - time_s
            epoch_loss = running_loss / len(dataloaders[phase].dataset)
            epoch_acc = running_corrects / len(dataloaders[phase].dataset)
            epoch_error = 1 - epoch_acc
            print("Epoch:{}, Phase: {}".format(epoch+1, phase))
            print("loss: {}, acc: {}".format(epoch_loss, epoch_acc))
            print("Phase {} Time: {}".format(phase, time_phase))
            total_time.append(time_phase)
            if phase == "val":
                best_acc = epoch_acc
                best_model_wts = copy.deepcopy(model.state_dict())

            if phase == "val":  # Save val accuracy and loss history.
                val_acc_history.append(epoch_acc)
                val_loss_history.append(epoch_loss)
                val_error.append(epoch_error)
                lr_history.append(lr)
            if phase =="train":
                train_acc_history.append(epoch_acc)
                train_loss_history.append(epoch_loss)
                train_error_history.append(epoch_error)

            print("lr: {}".format(optimizer.param_groups[0]['lr']))
            torch.cuda.empty_cache()
    model.load_state_dict(best_model_wts)
    return total_time, model, val_acc_history, val_loss_history, val_error, lr_history,
train_acc_history, train_loss_history, train_error_history
```

附录 4 IHPF、BHPF、GHPF 滤波器模块的代码实现

```
def IHPF(D_0, input_size):
    center = int(input_size / 2)
    filter = np.ones((input_size, input_size))
    for s in range(input_size):
        for t in range(input_size):
            D_st2 = (s+1 - center)**2 + (t+1 - center)**2
            if D_st2 < D_0**2:
                filter[s,t] = 0
    return filter

def BHPF(D_0, n, input_size):
    center = int(input_size / 2)
    filter = np.zeros((input_size, input_size))
    for s in range(input_size):
        for t in range(input_size):
            D_st2 = (s+1 - center)**2 + (t+1 - center)**2
            filter[s,t] = 1 / (1+(D_0/(np.sqrt(D_st2)))**(2*n))
    return filter

def GHPF(D_0, input_size):
    center = int(input_size / 2)
    filter = np.zeros((input_size, input_size))
    for s in range(input_size):
        for t in range(input_size):
            D_st2 = (s+1 - center)**2 + (t+1 - center)**2
            filter[s,t] = 1 - np.exp(-(D_st2/(2*D_0**2)))
    return filter
```

参考文献

[1] Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images[J]. 2009.

[2] Xiao H, Rasul K, Vollgraf R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms[J]. 2017.

[3] 李峻山，李旭辉，朱子江. 数字图像处理. 第三版. 北京: 清华大学出版社

[4] Wells PNT. Digital image processing: W.K. Pratt John Wiley, Chichester; 4th Edition 1991.

[5] 东红林，于莲芝. 基于 MATLAB 图像处理的频率域滤波分析及应用 [J]. 软件导刊，2017(10):209-212.

[6] 冈萨雷斯. 数字图像处理 [M]. 第三版. 北京: 电子工业出版社，2003.

[7] L. Gueguen, A., B., R. Liu, and J. Yosinski. Faster neural networks straight from jpeg. NIPS, 2018.

[8] R.Torfason, F.Mentzer, E.gstsson, M.Tschannen, R.Timofte, and L.Gool. Towards Image Understanding From Deep Compression Without Decoding. ICLR, 2018.

[9] K. XU, Z. Zhang, and F. Ren. Lapran: A Scalable Laplacian Pyramid Reconstructive Adversarial Network For Flexible Compressive Sensing Reconstruction. ECCV, 2018.

[10] C. Wu, M. Zaheer, H. Hu, R. Manmatha, A. Smola, and P. Krahenbuhl. Compressed Video Action Recognition. CVPR, 2018.

[11] M. Ehrlich and L. Davis. Deep Residual Learning in the JPEG Transform Domain. ICCV, 2019.

[12] Dalal N , Triggs B . Histograms of Oriented Gradients for Human Detection[C]. IEEE, 2005.

[13] Fei-Fei L , Perona P . A Bayesian hierarchical model for learning natural scene categories[C]. IEEE, 2005.

[14] H. Ahamed, I. Alam and M. M. Islam, HOG-CNN Based Real Time Face Recognition, 2018 International Conference on Advancement in Electrical and Electronic Engineering (ICAEEE), Gazipur, Bangladesh, 2018, pp.1-4.

[15] S. M. A. Sharif, Nabeel Mohammed, Sifat Momen,etc. Classification of Bangla Compound Characters Using a HOG-CNN Hybrid Model[M] Proceedings of the International Conference on Computing and Communication Systems. 2018.

[16] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. AISTATS, 2011.

[17] Tongwei Lu, Dandan Wang, and Yanduo Zhang, Fast object detection algorithm based on HOG and CNN, Proc. SPIE 10615, Ninth International Conference on Graphic and Image Processing (ICGIP 2017), 1061509 (10 April2018);

[18] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural computation,1(4):541–551, 1989.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradientbased learning applied to document recognition. IEEE, 86(11):2278–2324, 1998.

[20] D. Marr, Vision: A computational investigation into the human representation and processing of visual information. The MIT Press, 2010

[21] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift.ICML, 2015.

[22] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. IJCV.

[23] R. K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. NIPS, 2015.

[24] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. CVPR, 2016.

[25] L. Mao, M. Xie, Y. Huang and Y. Zhang, Preceding vehicle detection using Histograms of Oriented

Gradients, 2010 International Conference on Communications, Circuits and Systems (ICCCAS), Chengdu, 2010, pp. 354-358.
[26] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradientbased learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.
[27] Viola, Paul, and Michael Jones. Robust real-time face detection. IEEE, 2001.
[28] Krizhevsky, Alex. One weird trick for parallelizing convolutional neural networks. arXiv preprint arXiv:1404.5997 (2014).
[29] Szegedy, Christian, et al. Going deeper with convolutions. Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.
[30] Huang, Gao, et al. Densely connected convolutional networks. Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
[31] Hinton G E, Srivastava N, Krizhevsky A, et al. Improving neural networks by preventing co-adaptation of feature detectors[J]. arXiv preprint arXiv:1207.0580, 2012.
[32] Chen H , Wang Y , Xu C , et al. AdderNet: Do We Really Need Multiplications in Deep Learning?[J]. 2019.

DFT-Based Convolutional Neural Networks for Visual Recognition

Abstract：Histograms of Oriented Gradients (HOG) had been spread nearly all area of image outline feature extraction in neural networks. This paper proposes and explore a simple idea is that: Is there exist more efficient method of object outline extraction comparing to HOG? In this paper we propose and explore a new algorithm of image feature extraction based on Discrete Fourier Transform (DFT). We explicitly solve shorts of HOG contains the weak capability of different color space, short compute accuracy, long-time progress and inefficient ability of high-resolution image tasks. Considering the significant advantage of Discrete Fourier Transform, this algorithm detects the feature of object in image by switch color space to frequency space and then absorb outline of object through using different filters. We modify a small Visual Geometry Group Network (VGG) called Mini-VGG to test 3 separate high frequency filters on `Pytorch`. 3 separate high frequency filters (Idea High Pass Filter, Gauss High Pass Filter, Butterworth High Pass Filter) under different parameters compare to Rectangular-HOG (R-HOG) by Fashion-MINIST and CIFAR10 classification task on Mini-VGG. This new approach gives a near-perfect object outline feature extract algorithm. Finally, we validate the practice, accuracy, superior on some suspects. We also make a systematical discussion and the prospect development of the approach that DFT based object outline extraction.

Keywords：Discrete Fourier Analysis; Histograms of Oriented Gradients; Visual Geometry Group Network; High Pass Filter; CIFAR10; Fashion-MINIST