注意力机制Attention论文和代码大全-持续更新(一次写不完)

CV一线知名演员
学无止境，淘金不休

github.com/MenghaoGuo/A...

github.com/xmu-xiaoma66...

什么是注意力机制（Attention Mechanism）？

注意力机制的核心就是让网络关注其更需要更重要的地方，注意力机制就是实现网络自适应的一个方式。

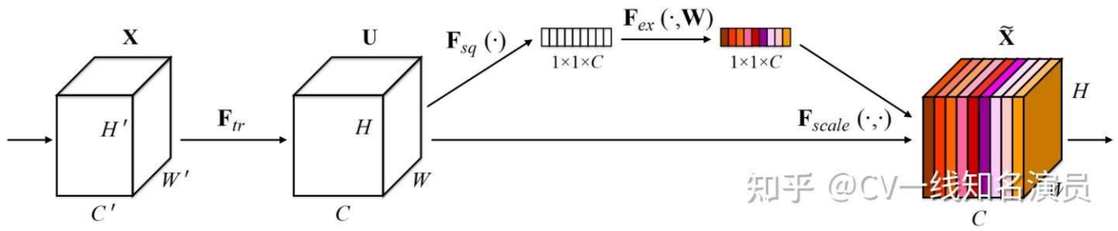
注意力机制：Channel Attention Module 通道注意力机制，Spatial Attention Module 空间注意力机制，通道&空间相结合

SENet(Squeeze-and-Excitation Networks) 通道注意力机制

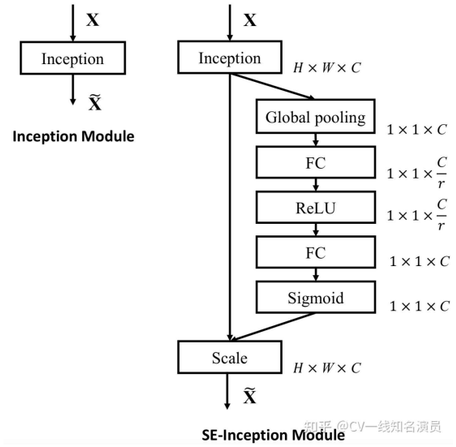
论文背景知识：SENet是Momenta公司&Oxford牛津大学，于2017年发表于CVPR2018；SENet是典型的通道注意力机制的代表作，且赢得了ImageNet2017最后一届的图象分类冠军 。

参考：论文地址*Squeeze-and-Excitation Networks* **GitHub** *github.com/hujie-frank/...*

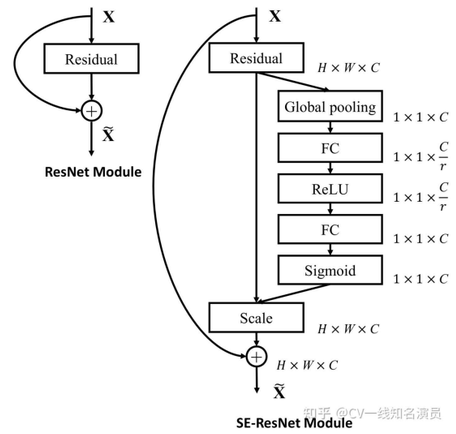
论文思想：通过显式的构建特征通道之间的相互依赖关系，通过学习的方式，自动获取每个特征通道之间的重要程度，然后根据此重要程度来提升有用的通道特征，并抑制当前任务用作用不大的通道特征。SENet结构如下图所示~



SENet框架图-Diagram of a Squeeze-and-Excitation building block



Schema of SE-Inception module



Schema of SE-ResNet module

> 全局平均池化 全局平均池化（Golbal Average Pooling）
> softmax sigmoid python除法//&/

```
#coding=utf-8
import os
import sys

import torch
from torch import nn

class SENet(nn.Module):
    def __init__(self,channel,ratio=4):
        super(SENet,self).__init__()
        self.avg_pool = nn.AdaptiveAvgPool2d(1)
        self.fc = nn.Sequential(
            nn.Linear(channel,channel//ratio,False),
            nn.ReLu(),
```

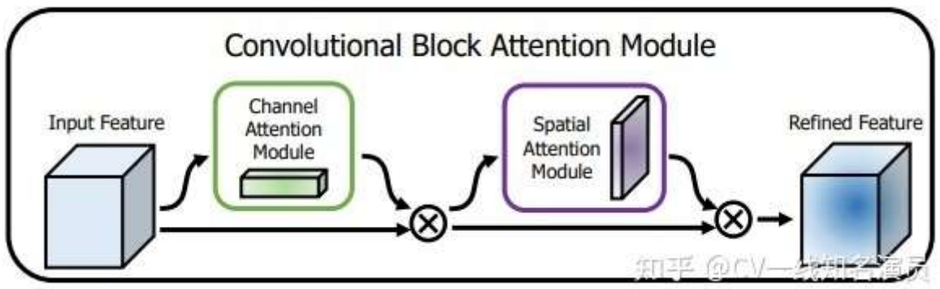
```
            nn.Linear(channel//ratio,channel,False),
            nn.Sigmoid()
        )
    def forward(self,x):
        b,c,h,w = x.size()
        avg = self.avg_pool(x).view([b,c]) # b,c,h,w->b,c
        fc = self.fc(avg).view([b,c,1,1]) # b,c->b,c//ratio->b,c->b,c,1,1
        return x*fc

model = SENet(512)
print(model)
inputs = torch.ones([2,512,128,128])
outputs = model(inputs)
```

CBAM(Convolutional Block Attention Module)通道注意力&空间注意力机制

论文背景知识：2018年发表于ECCV；CBAM是典型的通道注意力+空间注意力机制的代表作，且赢得了ImageNet2017最后一届的图象分类冠军 。

参考：论文地址*Squeeze-and-Excitation Networks* **GitHub** *github.com/hujie-frank/...*

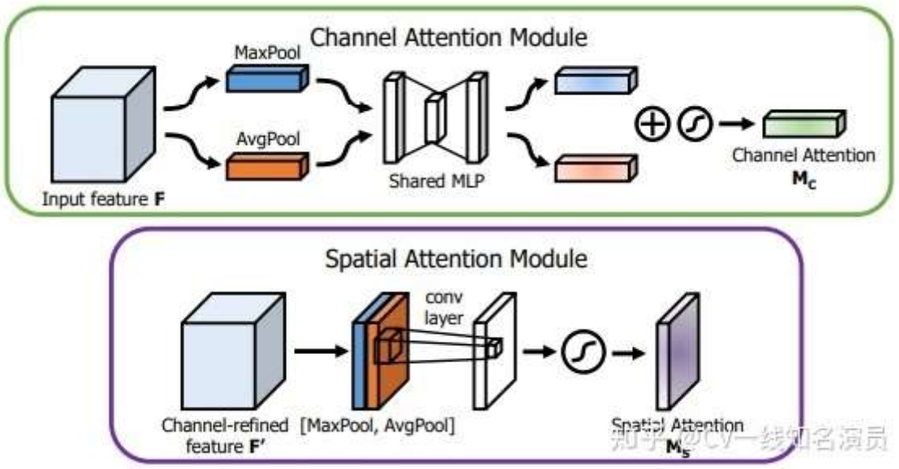
论文思想：通过显式的构建特征通道之间的相互依赖关系，通过学习的方式，自动获取每个特征通道之间的重要程度，然后根据此重要程度来提升有用的通道特征，并抑制当前任务用作用不大的通道特征。SENet结构如下图所示~



hhh

已赞同 2

分享



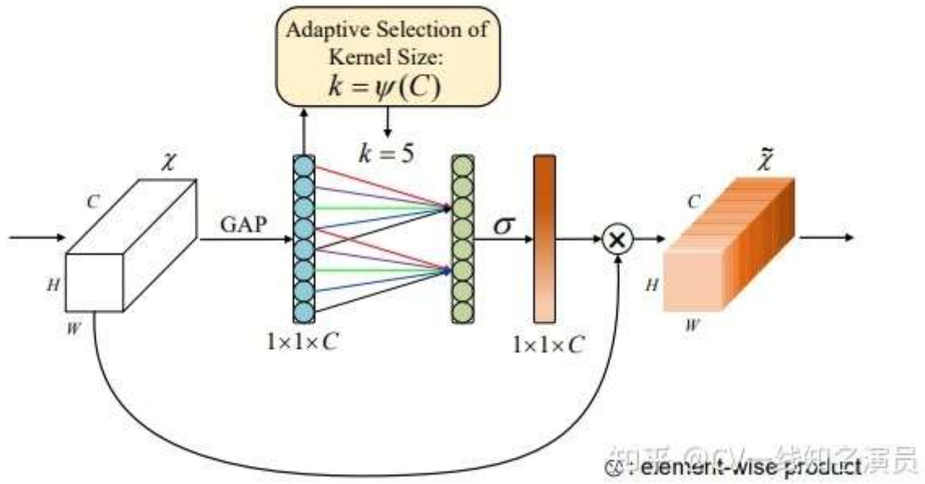
hhh

```python
#coding=utf-8
import os
import sys
import torch
from torch import nn

class ChannelAttention(nn.Module):
    def __init__(self,channel,ratio=4):
        super(ChannelAttention,self).__init__()
        self.max_pool = nn.AdaptiveMaxPool2d(1)
        self.avg_pool = nn.AdaptiveAvgPool2d(1)
        sef.fc = nn.Sequential(
            nn.Linear(channel,channel//ratio,False),
            nn.ReLu(),
            nn.Linear(channel//ratio,channel,False),
            nn.Sigmoid()
        )
        self.sigmoid = nn.Sigmoid()
    def forward(self,x):
        b,c,h,w = x.size()
        max_pool = self.max_pool(x).view([b,c])
        avg_pool = self.avg_pool(x).view([b,c])

        max_fc = self.fc(max_pool)
        avg_fc = self.fc(avg_pool)
```

```
        out = max_fc+avg_fc
        out = self.sigmoid(out).view([b,c,1,1])
        return out*x


class SpatialAttention(nn.Module):
    def __init__(self,kernel_size = 7):
        super(SpatialAttention,self).__init__()
        padding = 7//2
        self.conv = nn.Conv2d(2,1,kernel_size,1,padding,bias=False)
        self.sigmoid = nn.Sigmoid()
    def forward(self,x):
        b,c,h,w = x.size()
        max_pool = torch.max(x,dim=1,keepdim=True)
        avg_pool = torch.mean(x,dim=1,keepdim=True)
        pool_out = torch.cat([max_pool+avg_pool],dim=1)
        out = self.conv(pool_out)
        out = self.sigmoid(out)
        return out*x

class CBAM(nn.Module):
    def __init__(self,channel,ratio=4,kernel_size=7):
        super(CBAM,self).__init__()
        self.channel_attention = ChannelAttention(channel,ratio)
        self.spatial_attention = SpatialAttention(kernel_size)
    def forward(self,x):
        x = self.channel_attention(x)
        x = self.spatial_attention(x)
        return x

model = CBAM(512)
print(model)
inputs = torch.ones([2,512,128,128])
outpus = model(inputs)
```

ECANet CVPR2020 Efficient Channel Attention for Deep Convolutional Neural Networks

通道注意力机制，SENet的升级版本




```
#coding=utf-8
import os
import sys

import math

import torch
from torch import nn


class ECAnet(nn.Module):
    def __init__(self,channel,gamma=2,b=1):
        super(ECAnet,self).__init__()
        kernel_size = int(abs((math.log(channel,2)+b)/gamma)
        kernel_size = kernel_size if kernel_size%2 else kernel_size+1

        self.avg_pool = nn.AdaptiveAvgPool2d(1)
        padding = kernel_size//2
        self.conv = nn.Conv1d(1,1,kernel_size,padding=padding,bias=False)
        self.sigmoid = nn.Sigmoid()
    def forward(self,x):
        b,c,h,w = x.size()
        avg = self.avg_pool(x).view([b,1,c])
        out = self.conv(avg)
        out = self.sigmoid(out).view([b,c,1,1])
        return x*fc

model = ECAnet(512)
print(model)
inputs = torch.ones([2,512,128,128])
outputs = model(inputs)
```

Attention Is All You Need 2017NIPS

Transformer

发布于 2022-04-08 16:04

注意力机制　注意力　论文

推荐阅读

Self-Attention 加速方法一览：ISSA、CCNet、CGNL...

好久没更新，搬运之前的笔记一篇。 Attention 机制最早在NLP 领域中被提出，基于attention 的transformer结构近年在NLP的各项任务上大放异彩。在视觉任务中，attention也收到了很多的关注，...

林天威　发表于Video...

计算机视觉中的注意力：PyTorch中实现MultiHead...

deeph...　发表于deeph...

CV注意力机制论文阅读笔记

CV注意力机制Non-local ~ SE ~ CcNet ~ GC-Net ~ Gate ~ CBAM ~ Dual Attention ~ Spatial Attention ~ Channel Attention ~ ...【只要你能熟练的掌握加法、乘法、并行、串行四大法则，外...

AdamL...　发表于CV学习笔...

用于Transformer的6种注意力的数学原理和代码实现

deeph...　发表于deeph...

还没有评论

写下你的评论...