# Introduction

对于普通卷积，认为有两个特性：

Spatial-agnostic：使用同一个卷积核对整张特征图做卷积，提高了效率，同时也让在不用位置的同一个特征具有相同的解释性。（比如对鸟的图片，如果卷积核对鸟嘴比较敏感，那么不管鸟嘴的位置在图中的哪个部分都可以被同一个卷积核侦测到）。

但是这种特性也让CNN失去了在不同的视觉模式下适应不同空间位置的能力（比如经过不同角度旋转的狗图片，在视觉上是不同的角度的同一只狗，但是在CNN就不是同一只，也就说明了在train之前data augment中random crop，rotation是有用的）。

Channel-specific：对于多个卷积核每个卷积核生成一个通道的特征图，这些特征图包含着不同的信息，也就说明了不同的信息会被编码在不同的通道，除此之外，CNN也更倾向于紧实小巧的卷积核，所以kernel size一般不超过3。

卷积核通道间的冗余性被证实，这浪费了CNN的flexibility，也使得人们怀疑是否有必要使用不同的channels

Involution：（在空间上相互独立，共享通道）

Spatial-specific：可以搜集一个较为宽泛的空间上的信息，克服了convolution的对应的缺点；在不同的空间位置上调整权重，在空间域上自动学习重要性

Channel-agnostic：

如果involution像convolution的参数那样固定矩阵和反向传播，就会失去对多分辨率输入的迁移能力，特定空间位置的involution kernel可以根据输入的特征向量与自己的操作实现唯一的产生，并且通过通道共享减少了参数的冗余性。基于这两点，一个involution操作的复杂度是与channels的数量线性相关

揭示了selfattention和involution的关系

Selfattention是involution的一个特例

# Design of involution

在Rednet中是通道-空间，通道独立，空间独立三种方式交替轮流执行，保证了在不损害模型准确率的情况下尽量的最小化模型

Involution.py中的call函数：

原文中有提到involution中的weights是从输入中产生的，也就是

Line40：对inputs进行conv，生成weights -> shape:B\*H\*W\*C

Line41~47：对weights进行reshape->B\*G\*1\*K\*K\*H\*W （上一步中C已经设定为G\*K\*K的数量）

Line48~54：对inputs分成一块一块的 out->shape:B\*H\*W\*(C\*K\*K)

Line55~56：对out进行reshape ->shape:B\*G\*GC\*K\*K\*H\*W（其中G\*GC=filter=C， filter=C是提前设定好的）

Line57：out shape -> B\*G\*GC\*K\*K\*H\*W

weights shape -> B\*G\*1\*K\*K\*H\*W

输出：shape B\*G\*GC\*(K\*\*2)\*H\*W

Line58：reduce\_sum：shape –>B\*G\*GC\*H\*W

再reshape成B\*(G\*GC)\*H\*W 🡺 返回结果shape：B\*H\*W\*C

对group的理解：

对于一个input设为X，其shape为H\*W\*C，即

设每一个pixel位置的x做运算，则需要H\*W个involution kernel，显然这样是不好的。所以让x于相邻位置的x’共享参数，设分组为G，那么，每个involution kernel被（H/G）\*（W/G）个pixel位置的x共享

