Presentation of CSI5139

Junzheng Wu
Jiahao Li

Overview

- The development trend of convolutional neural networks
- What is group convolution
- What is depth wise convolution
- Network structure of Mobilenet v1
- Low Dimension Data Collapses
- Degradation of Features(In RELU)
- Whole structure
- Implementation

The development trend of convolutional neural networks

Firstly

- deeper and more complicated networks(such as VGG and Googlenet)

Secondly

- not necessarily making networks more efficient with respect to size and speed

Problems caused by this trend

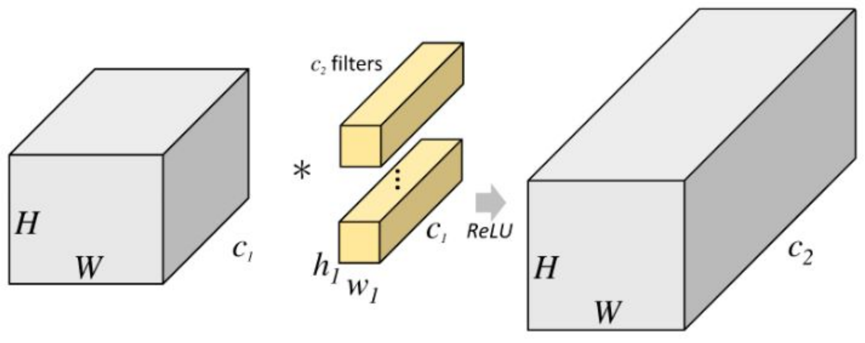
1. These CNN structures are difficult to matched to the design requirements for a computationally limited platform

2. Over-fitting caused by excessive parameters

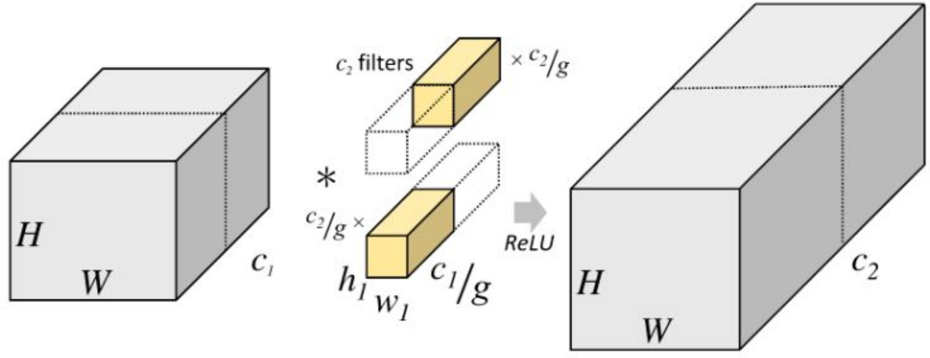
3. These structures are too dependent on large data sets

What is group convolution

Group convolution first appeared in AlexNet. Due to limited hardware resources at the time, the convolution operation could not be all placed on the same GPU when training AlexNet. Therefore, the author distributed the feature maps to multiple GPUs for processing. Finally, the author combines the results of multiple GPUs



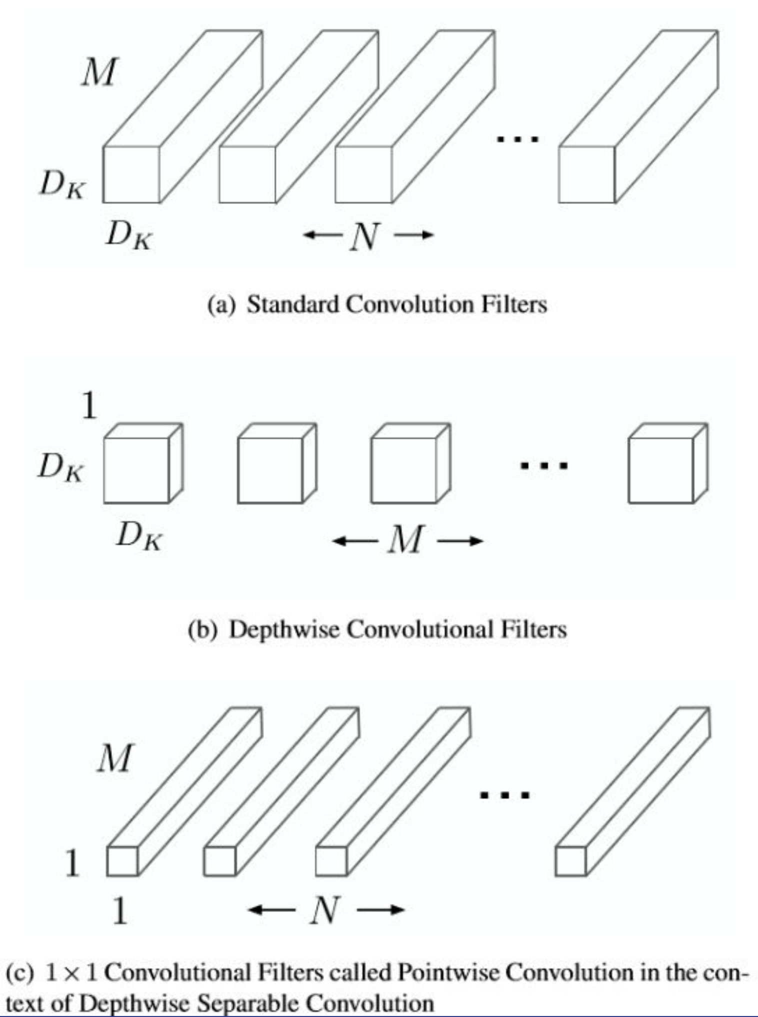
A normal convolutional layer. Yellow blocks represent learned parameters, gray blocks represent feature maps/input images (working memory).



A convolutional layer with 2 filter groups. Note that each of the filters in the grouped convolutional layer is now exactly half the depth, i.e. half the parameters and half the compute as the original filter.

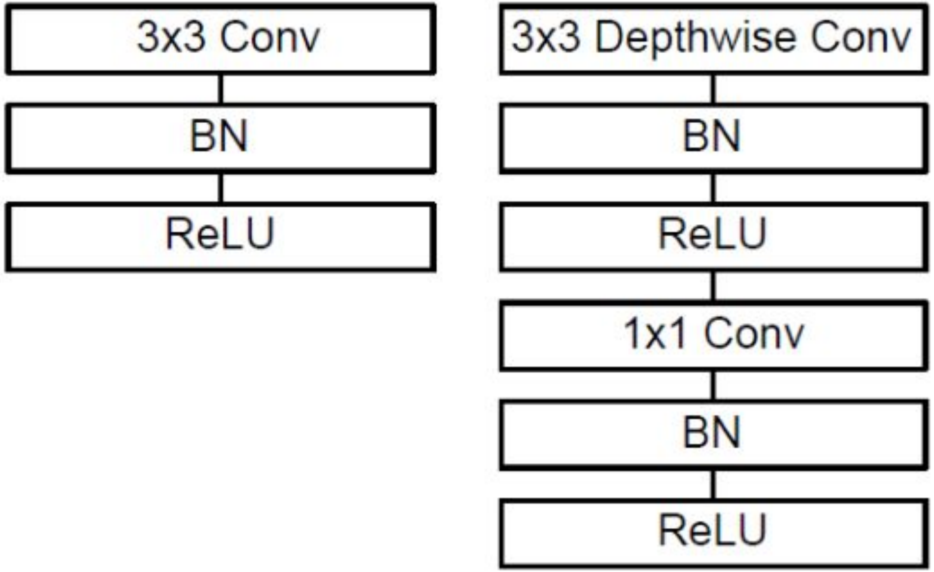
What is depth wise convolution

The MobileNet model is based on depthwise separable convolutions which is a form of factorized convolutions which factorize a standard convolution into a depthwise convolution and a 1×1 convolution called a point wise convolution



Network structure of Mobilenet v1

The MobileNet structure is built on depthwise separable convolutions as mentioned in the previous section except for the first layer which is a full convolution

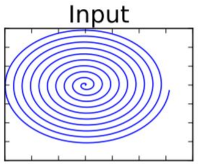


MobileNet v2
&
Try some new things...

The Problems

1. Low Dimension Data Collapses

Assumption:



A input $x_{2\times n}$

A random matrix and its inverse $T_{m\times 2}$ $T^{-1}_{2\times m}$

If we apply $y_{m\times n} = RELU(T_{m\times 2} \cdot x_{2\times n})$

Then projected back $x'_{2\times n} = T^{-1}_{2\times m} \cdot y_{m\times n}$

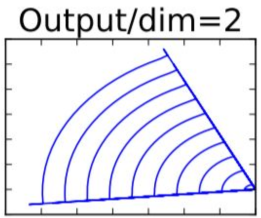
How much information will be reserved?

(Images comes from Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2)

The Problems

1. Low Dimension Data Collapses

Highly depends on output dimension n :



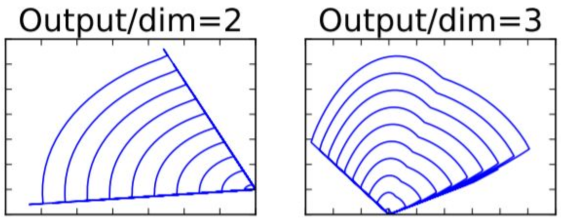
Remove RELU or increase the dimension ?

(Images comes from Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2)

The Problems

1. Low Dimension Data Collapses

Highly depends on output dimension n :



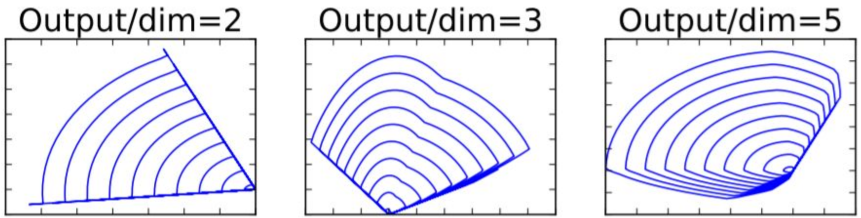
Remove RELU or increase the dimension ?

(Images comes from Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2)

The Problems

1. Low Dimension Data Collapses

Highly depends on output dimension n :



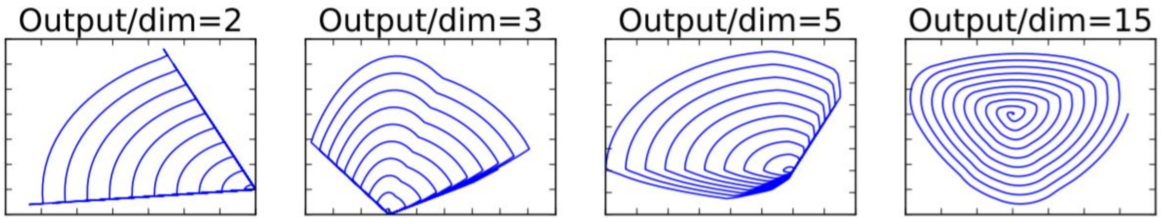
Remove RELU or increase the dimension ?

(Images comes from Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2)

The Problems

1. Low Dimension Data Collapses

Highly depends on output dimension n :



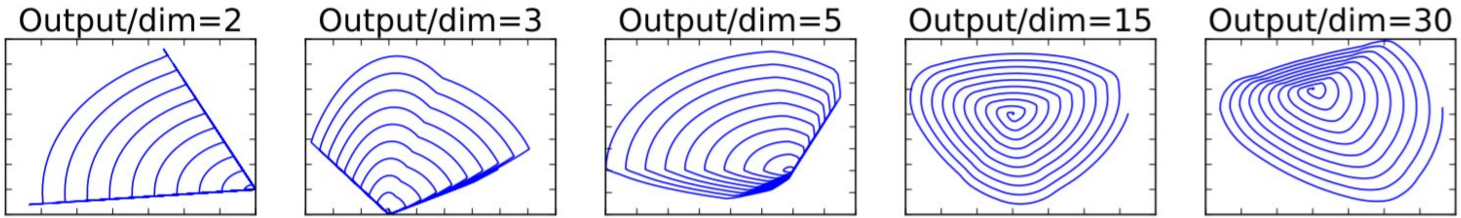
Remove RELU or increase the dimension ?

(Images comes from Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2)

The Problems

1. Low Dimension Data Collapses

Highly depends on output dimension n :

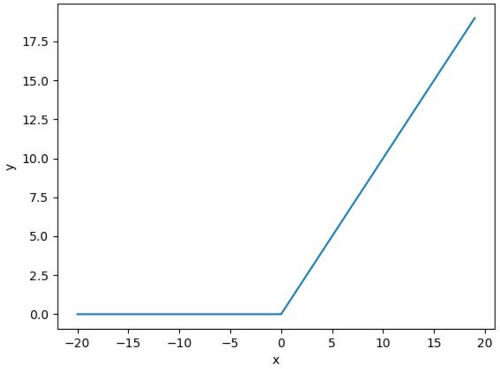


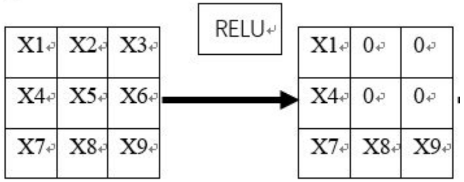
Remove RELU or increase the dimension ?

(Images comes from Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2)

The Problems

2. Degradation of Features(In RELU)

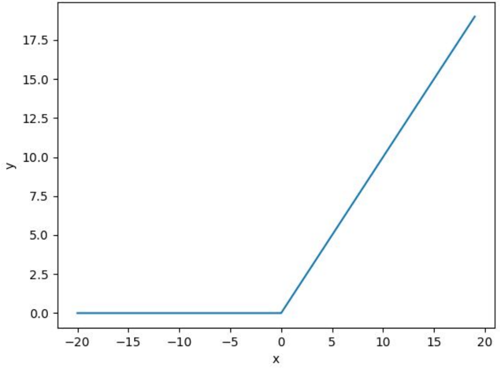


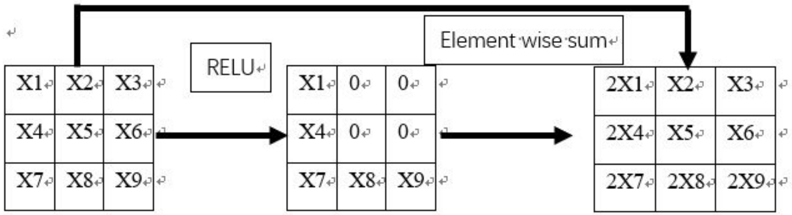


(Images comes from Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2)

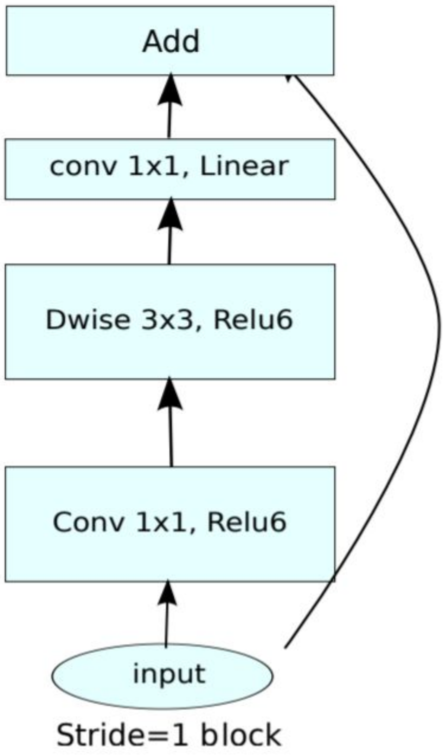
The Problems

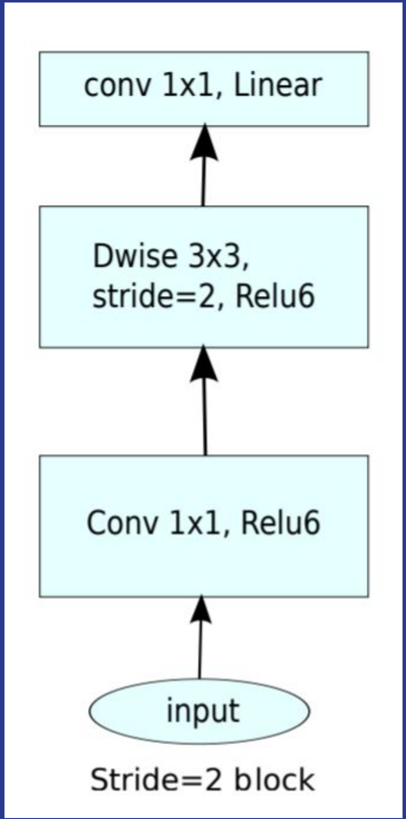
2. Degradation of Features(In RELU)





(Images comes from Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2)





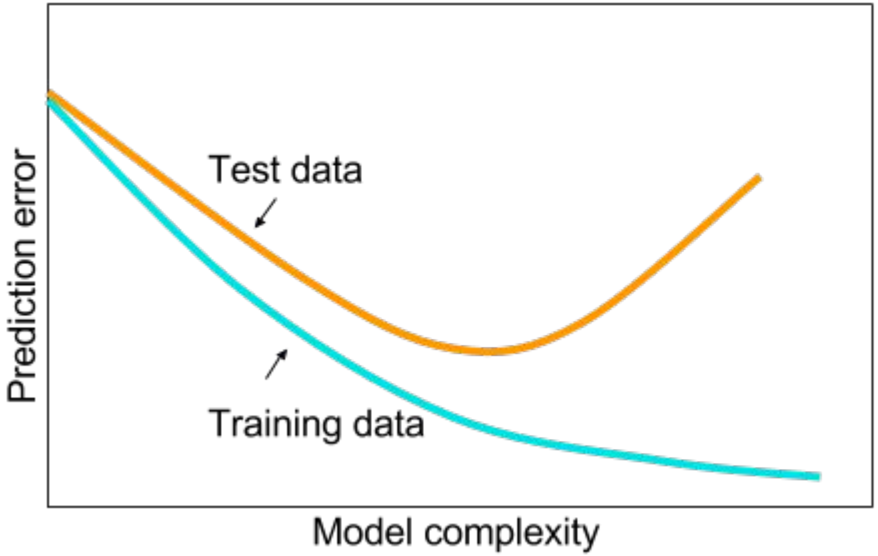
(Images comes from Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2

Whole Structure

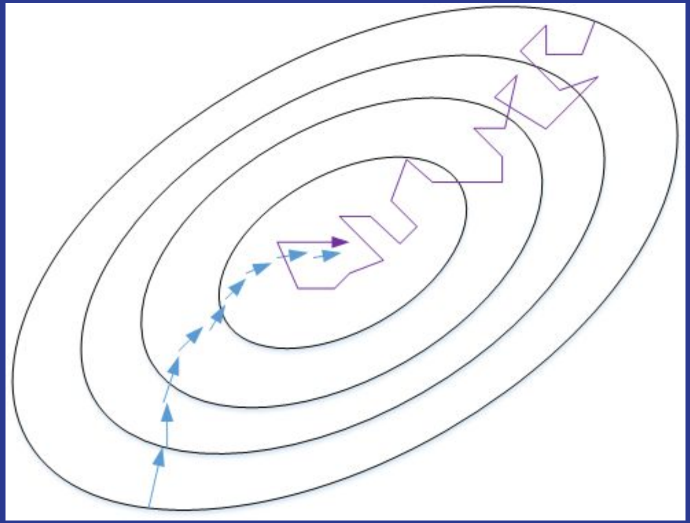
Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

(Images comes from Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2)

Overfitting



Training Time



Implementation

Firstly

- Implement MobileNet v2 in tensorflow

Secondly

- Compare with VGGNet and GoogleNet Inception v3
    a. Iterations to achieve the same accuracy
    b. Accuracy when fully trained
    c. Generalization capacity on small dataset

Q & A

Thanks