

## FSDL (2021) · 课程资料包 @ShowMeAI



视频  
中英双语字幕



课件  
一键打包下载



笔记  
官方笔记翻译



代码  
作业项目解析



视频 · B 站 [ 扫码或点击链接 ]  
<https://www.bilibili.com/video/BV1iL411t7jE>



课件 & 代码 · 博客 [ 扫码或点击链接 ]  
<http://blog.showmeai.tech/berkeley-fsdl>

深度学习  
神经网络  
循环神经网络

可解释性  
计算机视觉  
数据管理

持续集成

迁移学习  
Transformer  
部署模型

卷积神经网络  
深度神经网络调试  
监控模型  
测试

Awesome AI Courses Notes Cheatsheets 是 [ShowMeAI](#) 资料库的分支系列，覆盖最具知名度的 **TOP50+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

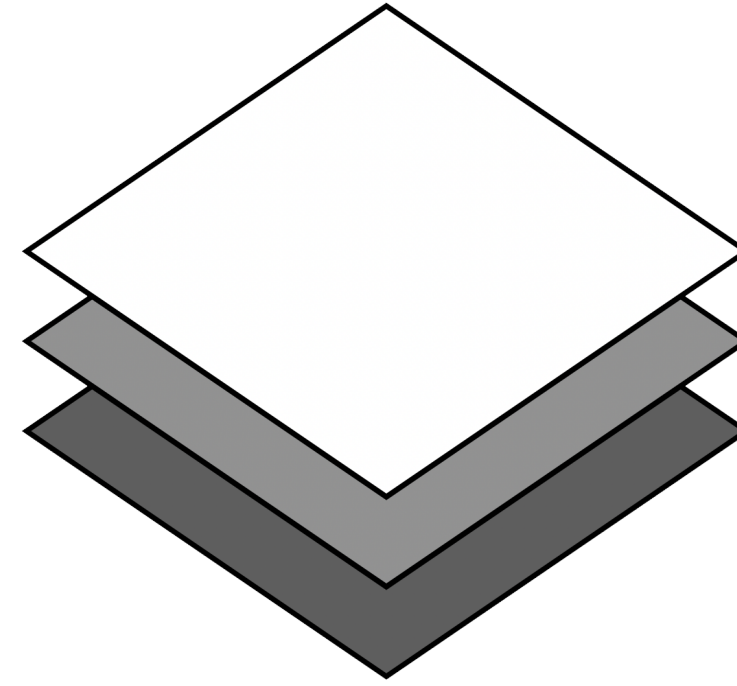
**点击** 课程名称，跳转至课程 **资料包** 页面，**一键下载** 课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n
# Awesome AI Courses Notes Cheatsheets · 持续更新中			
知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094



### 微信公众号

资料下载方式 2：扫码点击 **底部菜单栏**  
称为 **AI 内容创作者**？回复 [ 添砖加瓦 ]



# Lecture 2.A

## Convolutional Networks

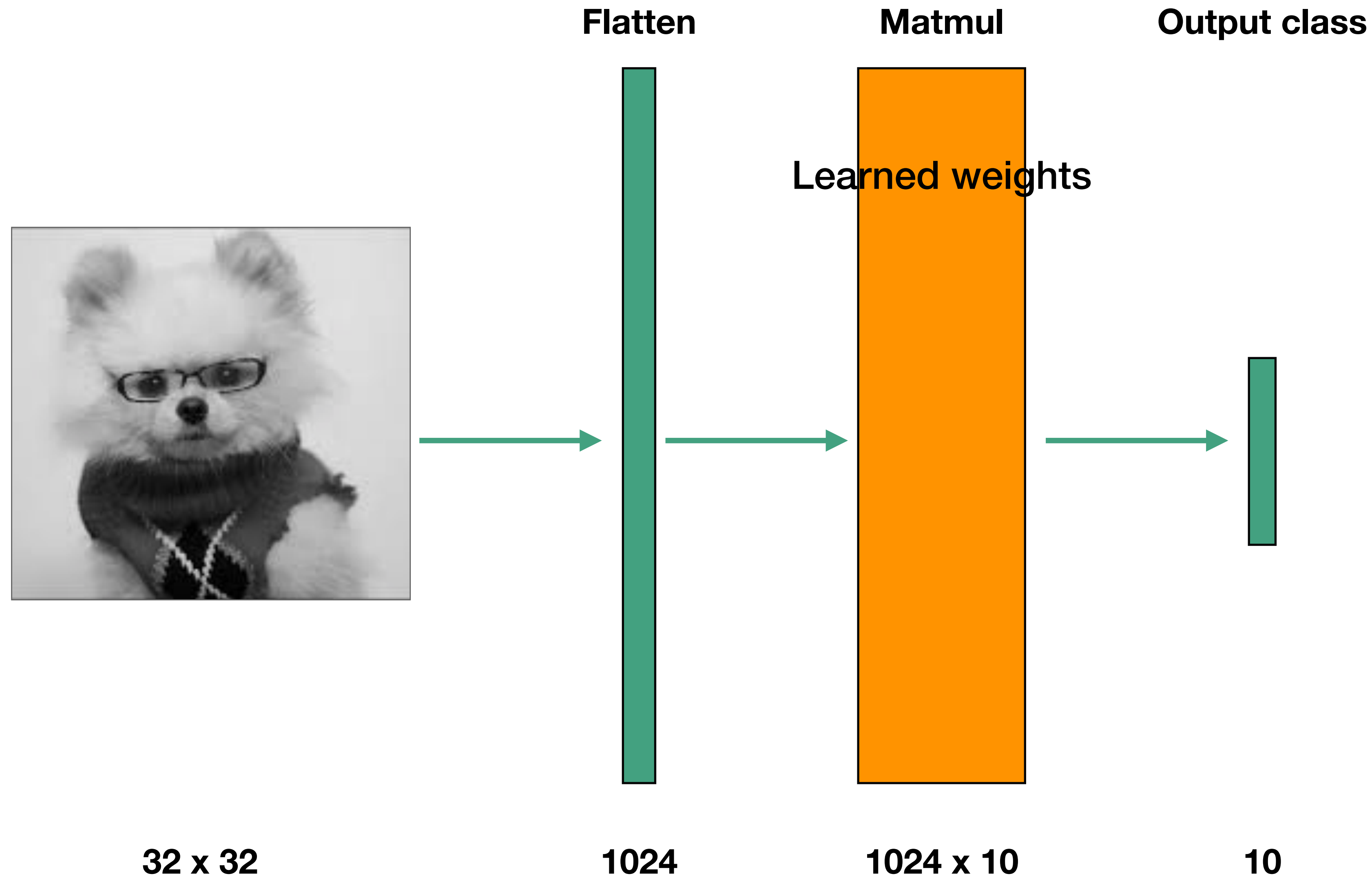
# Agenda

- 1. Review of the convolution operation**
2. Other important operations for ConvNets
3. Classic ConvNet architectures

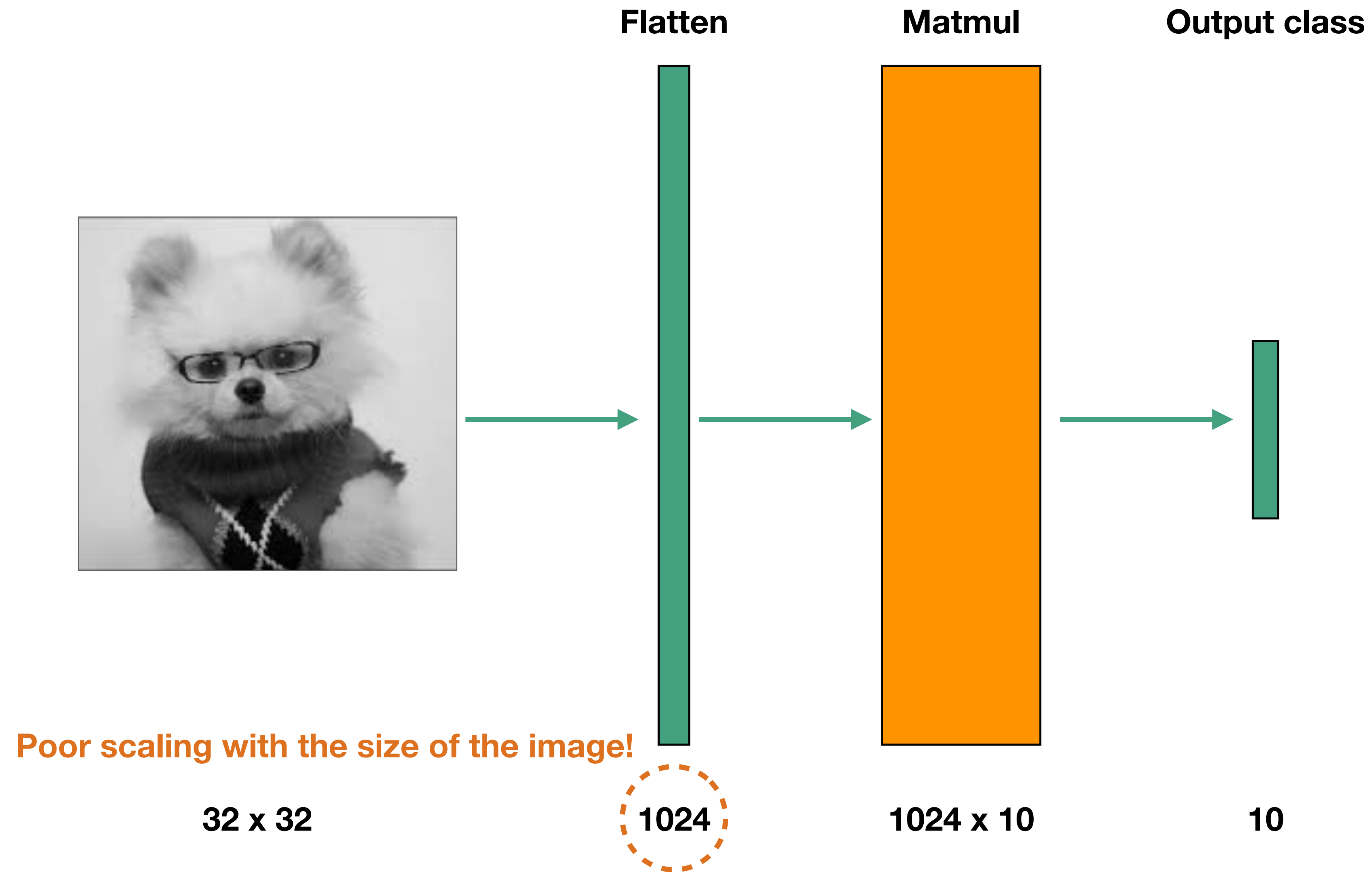
# Review of convolutions

- **What's a convolutional filter?**
- Filter stacks and ConvNets
- Strides & padding
- Filter math
- Implementation notes

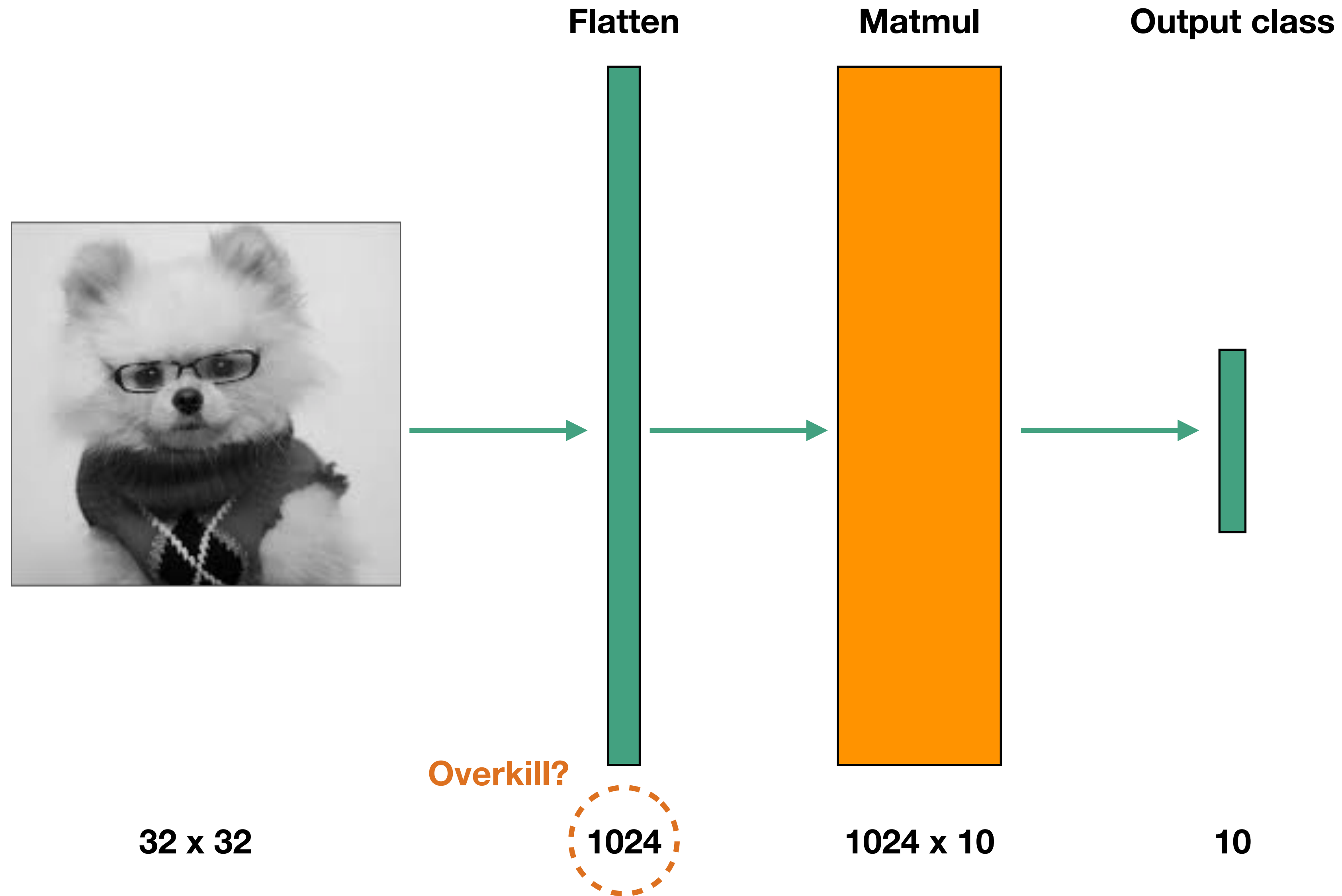
# Why not use FC for images?



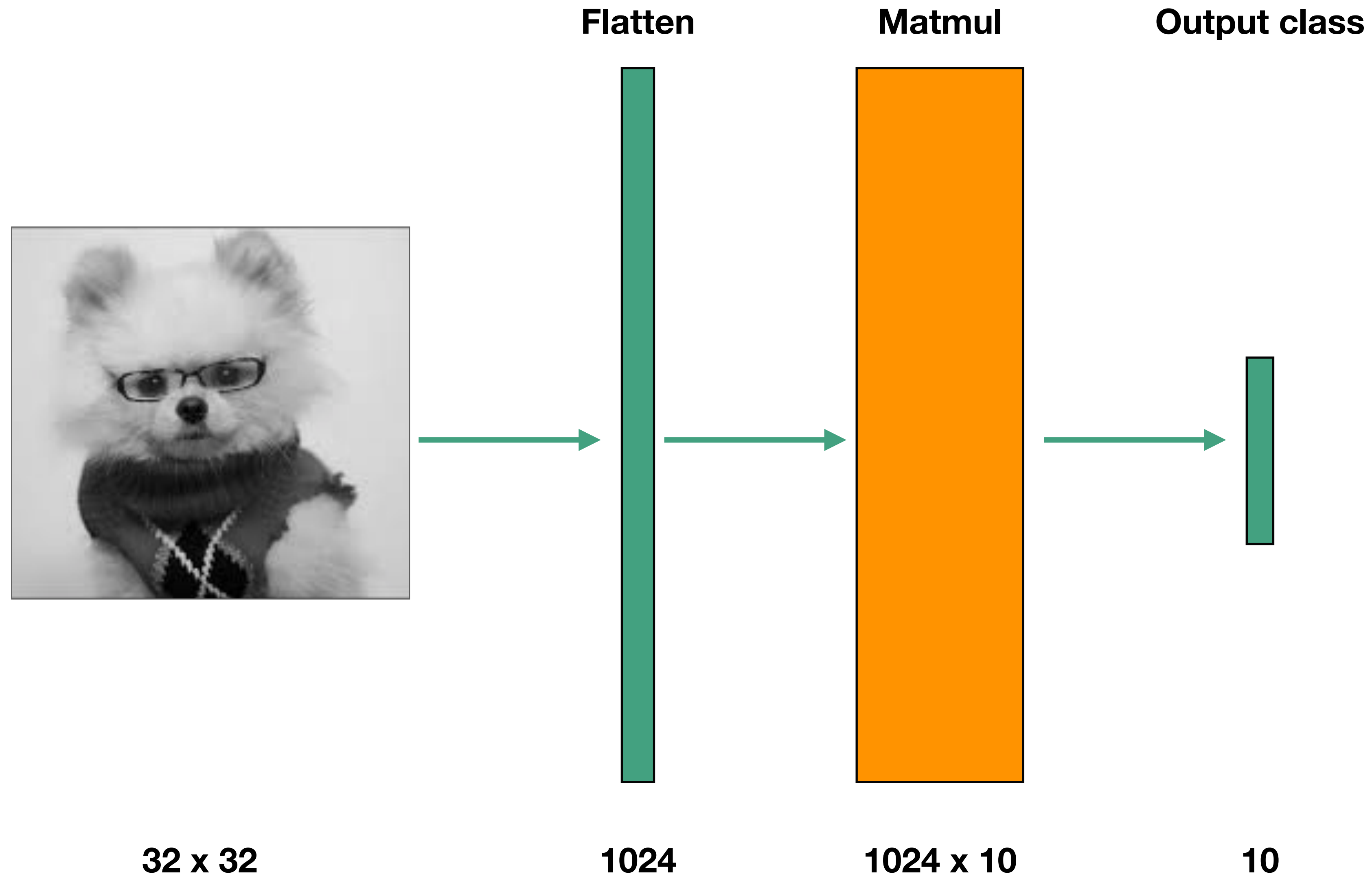
# Why not use FC for images?



# Why not use FC for images?

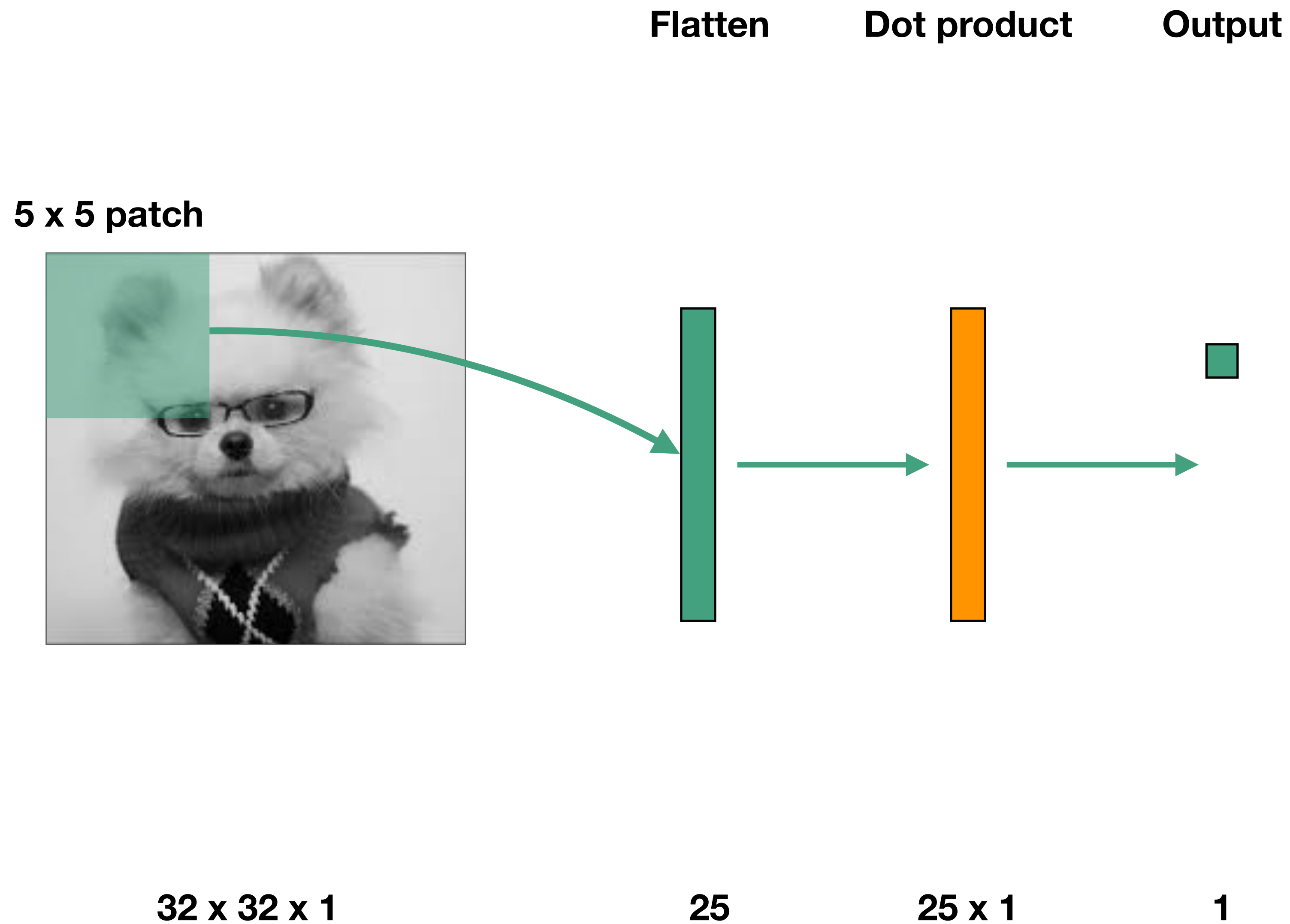


# Why not use FC for images?





# Convolutional filters



# Convolutional filters

Flatten

Dot product

Output

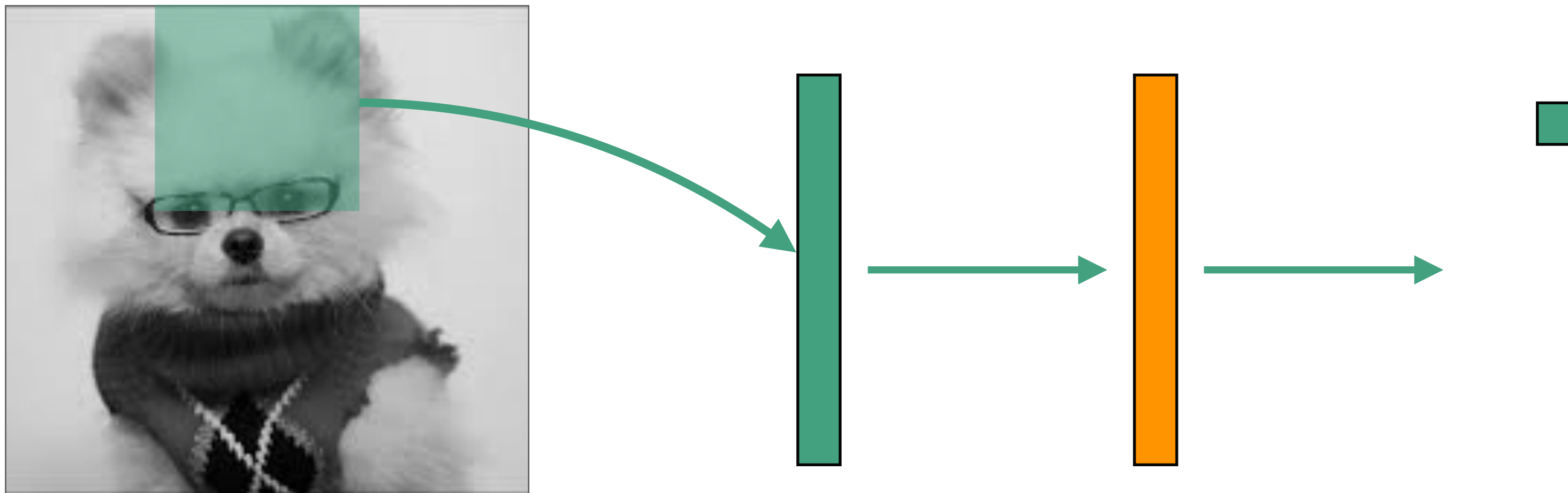


# Convolutional filters

Flatten

Dot product

Output

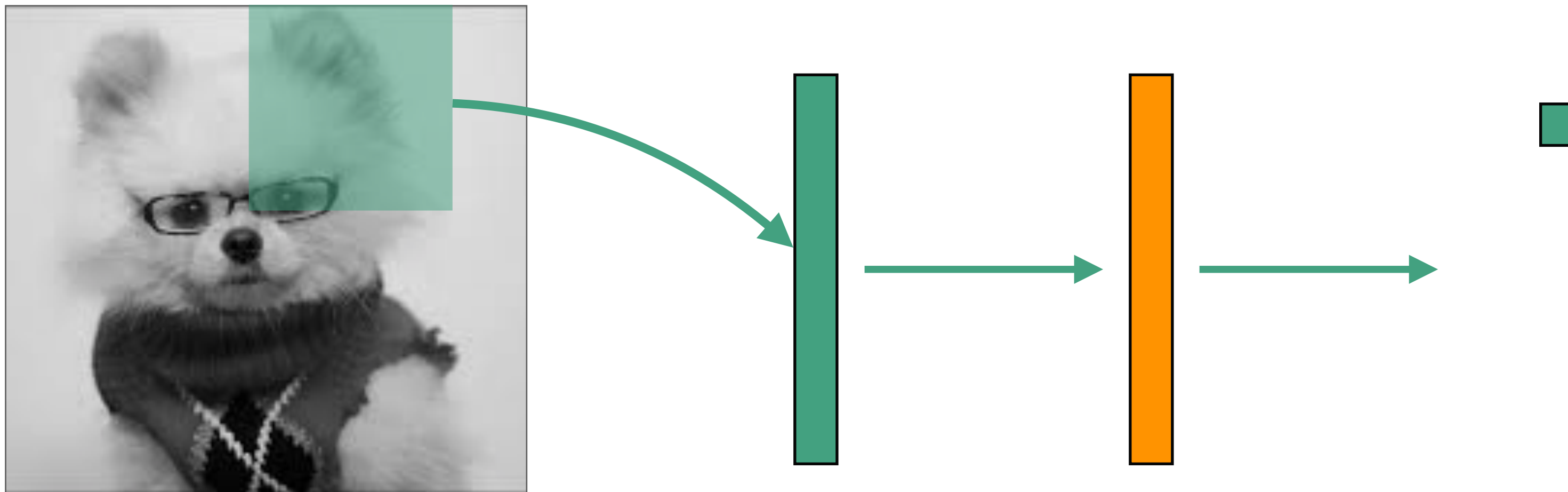


# Convolutional filters

Flatten

Dot product

Output

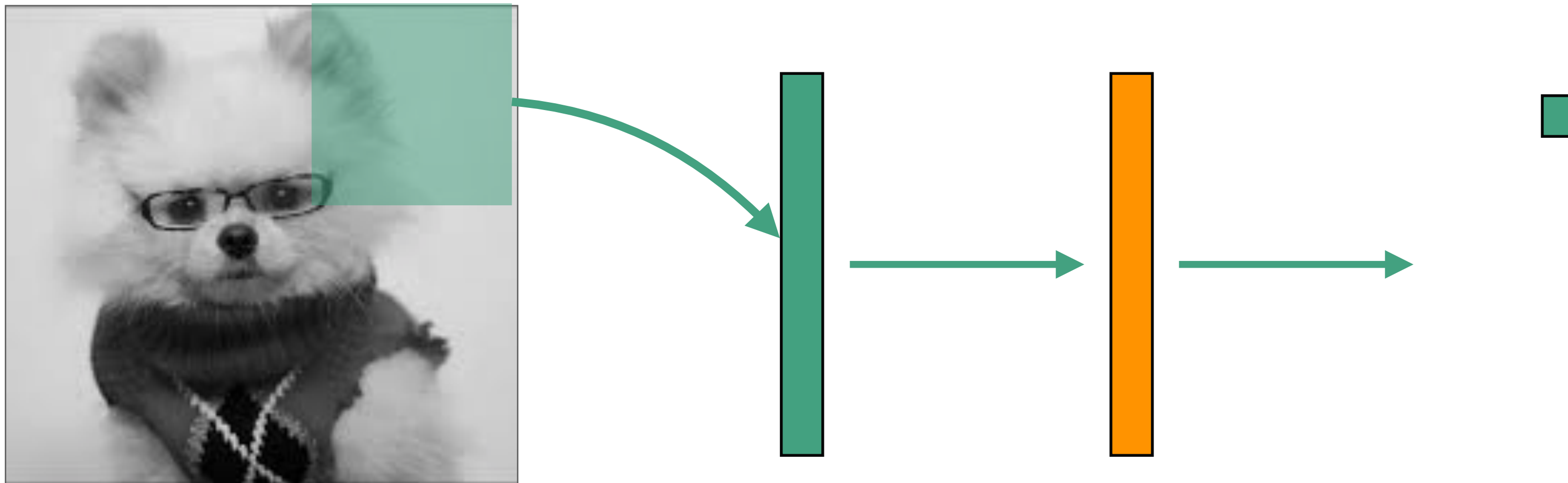


# Convolutional filters

Flatten

Dot product

Output



# Convolutional filters

Flatten

Dot product

Output



**...sliding continues...**

# Convolutional filters

Flatten

Dot product

Output



5 x 5 patch

32 x 32 x 1

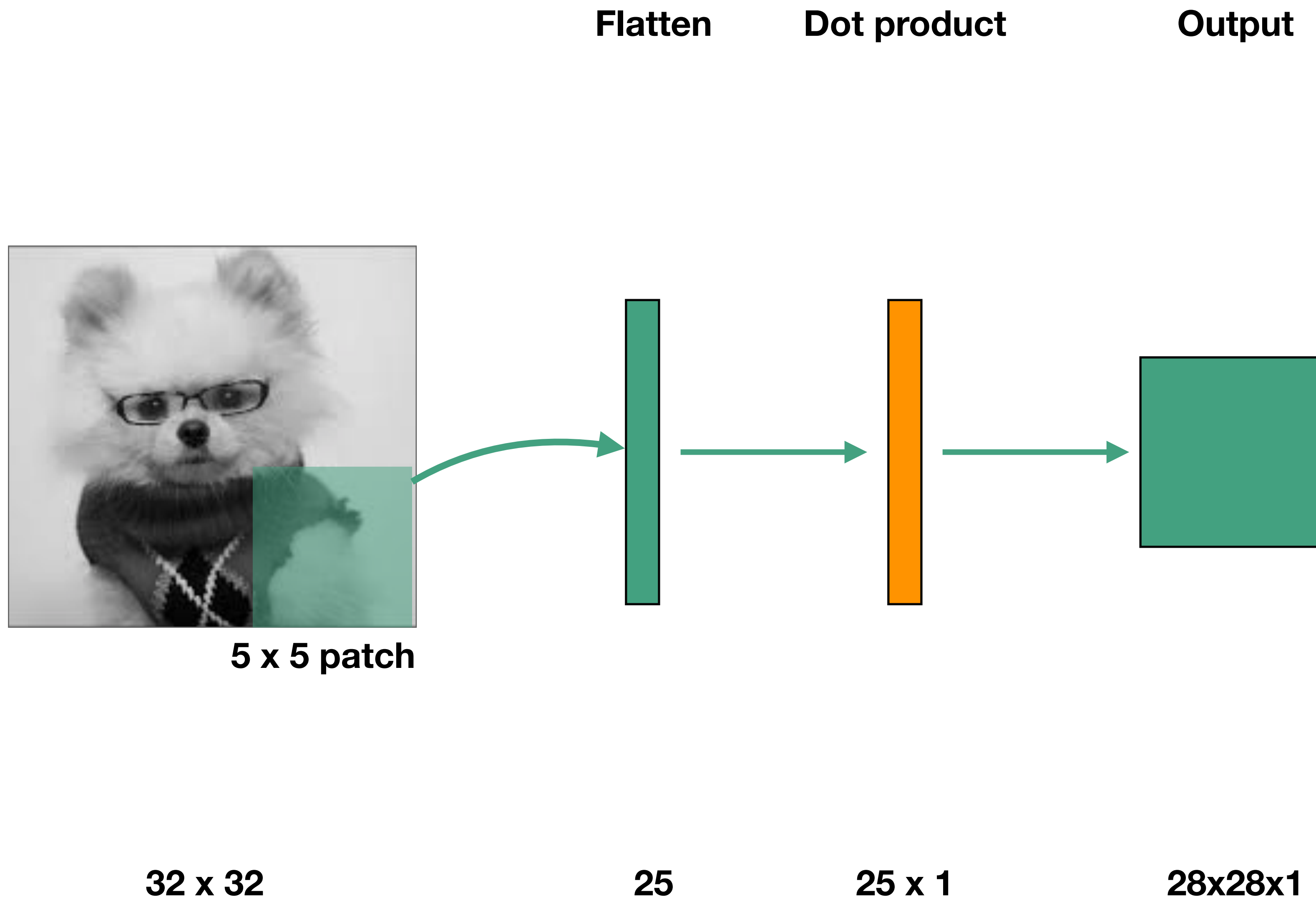
25

25 x 1

1




# Convolutional filters



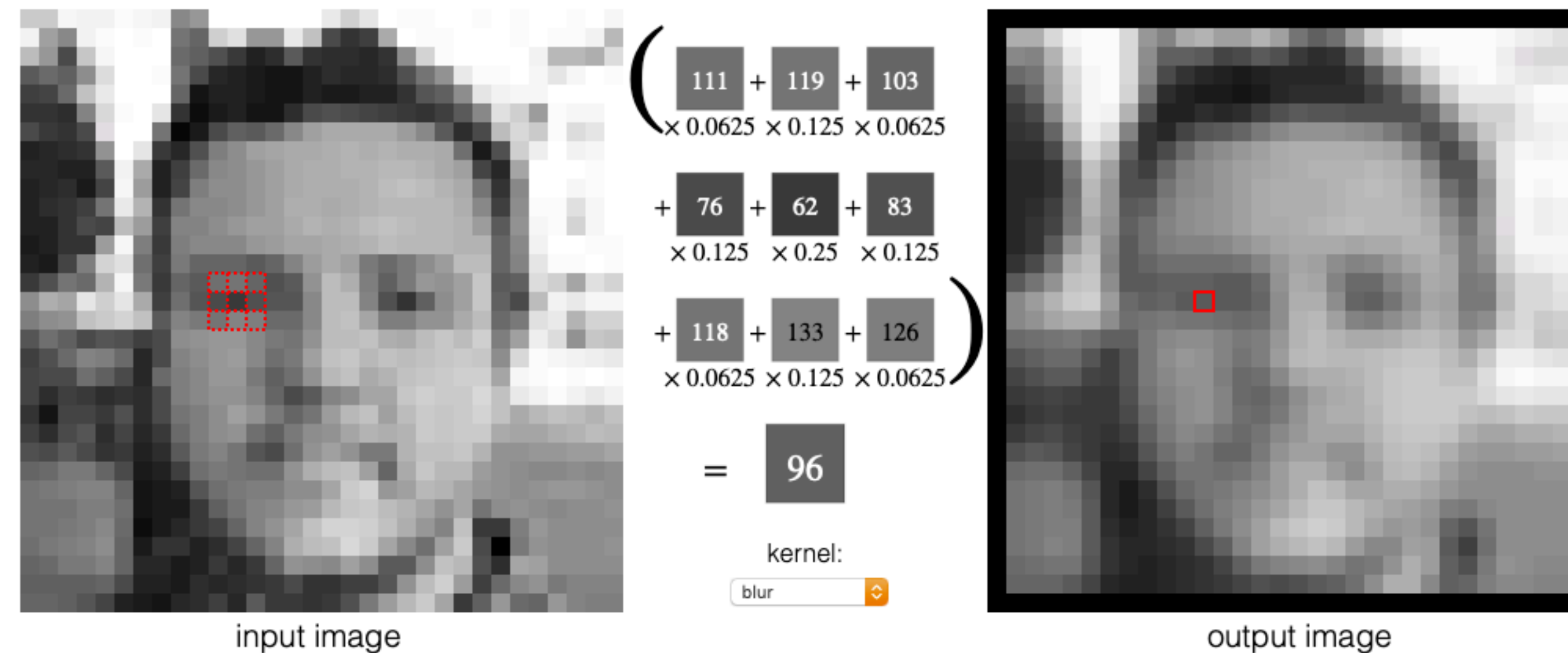
# What can a conv filter do?

Let's walk through applying the following 3x3 **blur** kernel to the image of a face from above.

blur 

$$\begin{pmatrix} 0.0625 & 0.125 & 0.0625 \\ 0.125 & 0.25 & 0.125 \\ 0.0625 & 0.125 & 0.0625 \end{pmatrix} \leftarrow \text{Instead of hard-coding the weights, we can learn them!}$$

Below, for each 3x3 block of pixels in the image on the left, we multiply each pixel by the corresponding entry of the kernel and then take the sum. That sum becomes a new pixel in the image on the right. Hover over a pixel on either image to see how its value is computed.

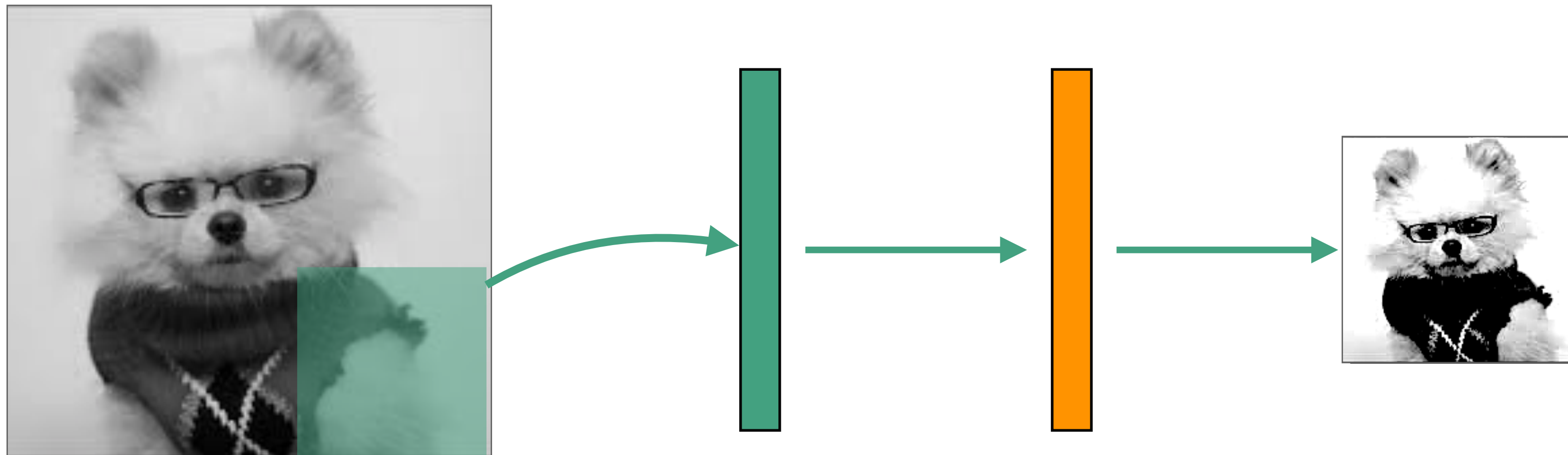


# Convolutional filters

Flatten

Dot product

Output



**32 x 32 x 3**

**75**

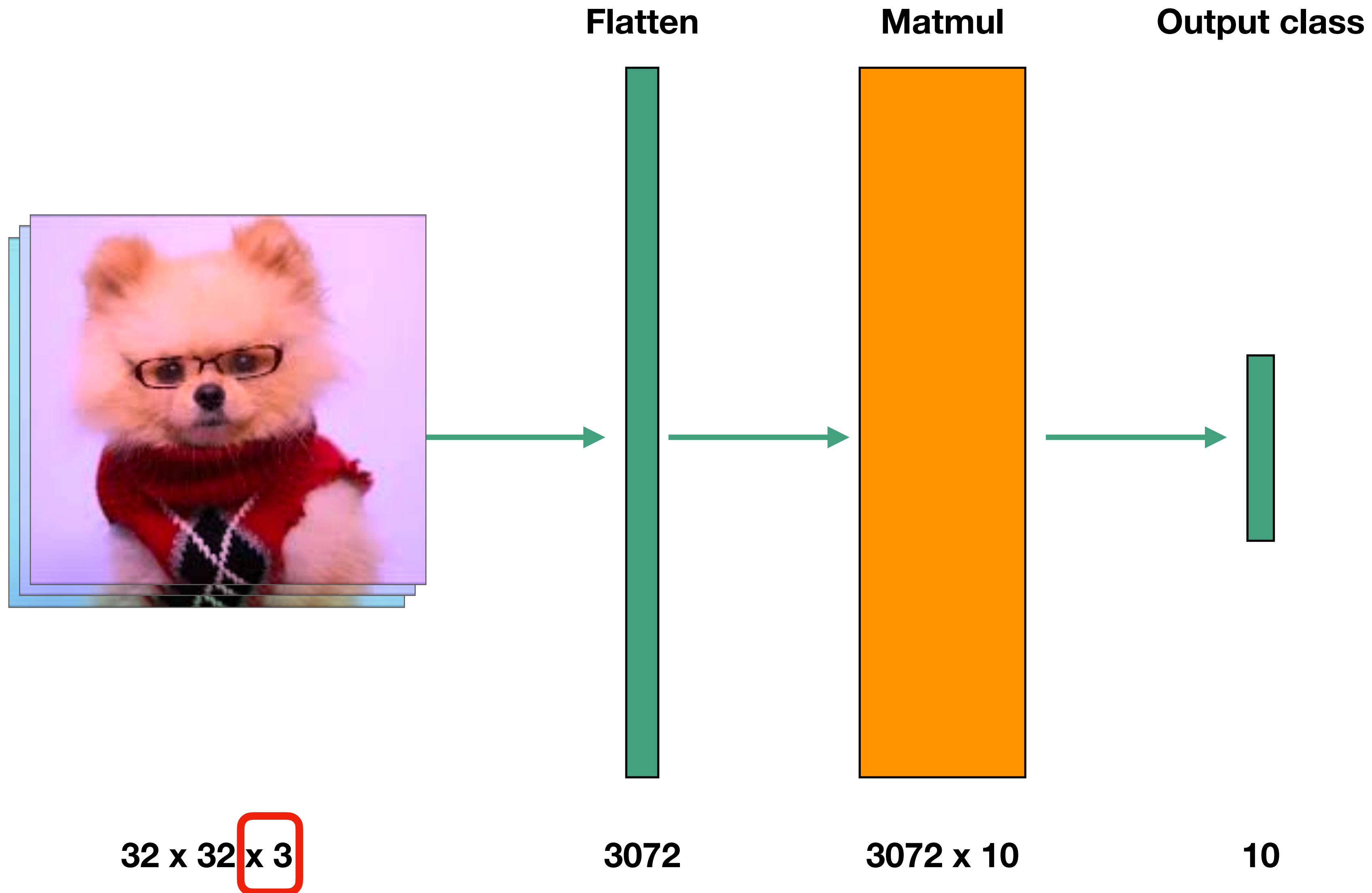
**75 x 1**

**28x28x1**

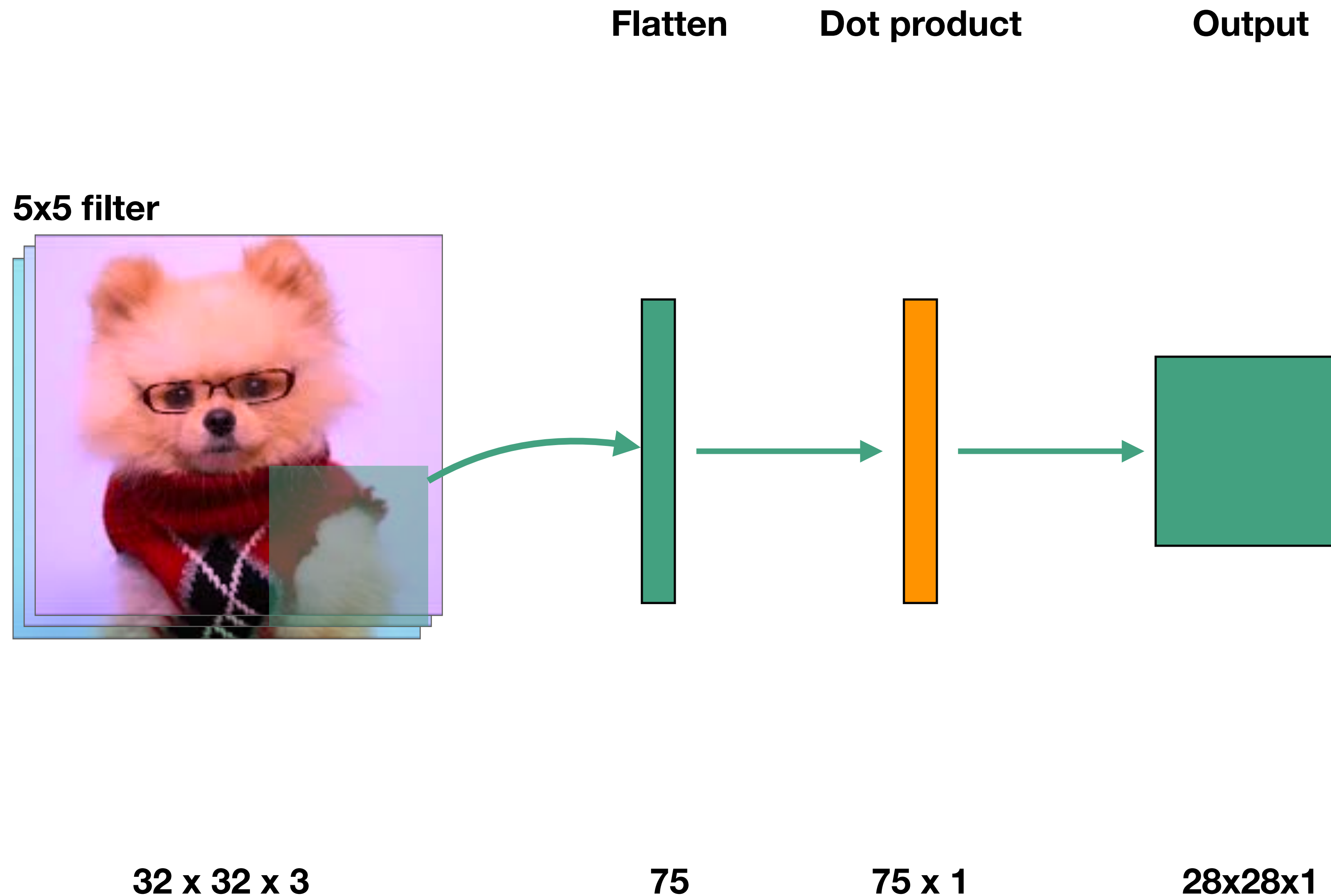
# Review of convolutions

- What's a convolutional filter?
- **Filter stacks and ConvNets**
- Strides & padding
- Filter math
- Implementation notes

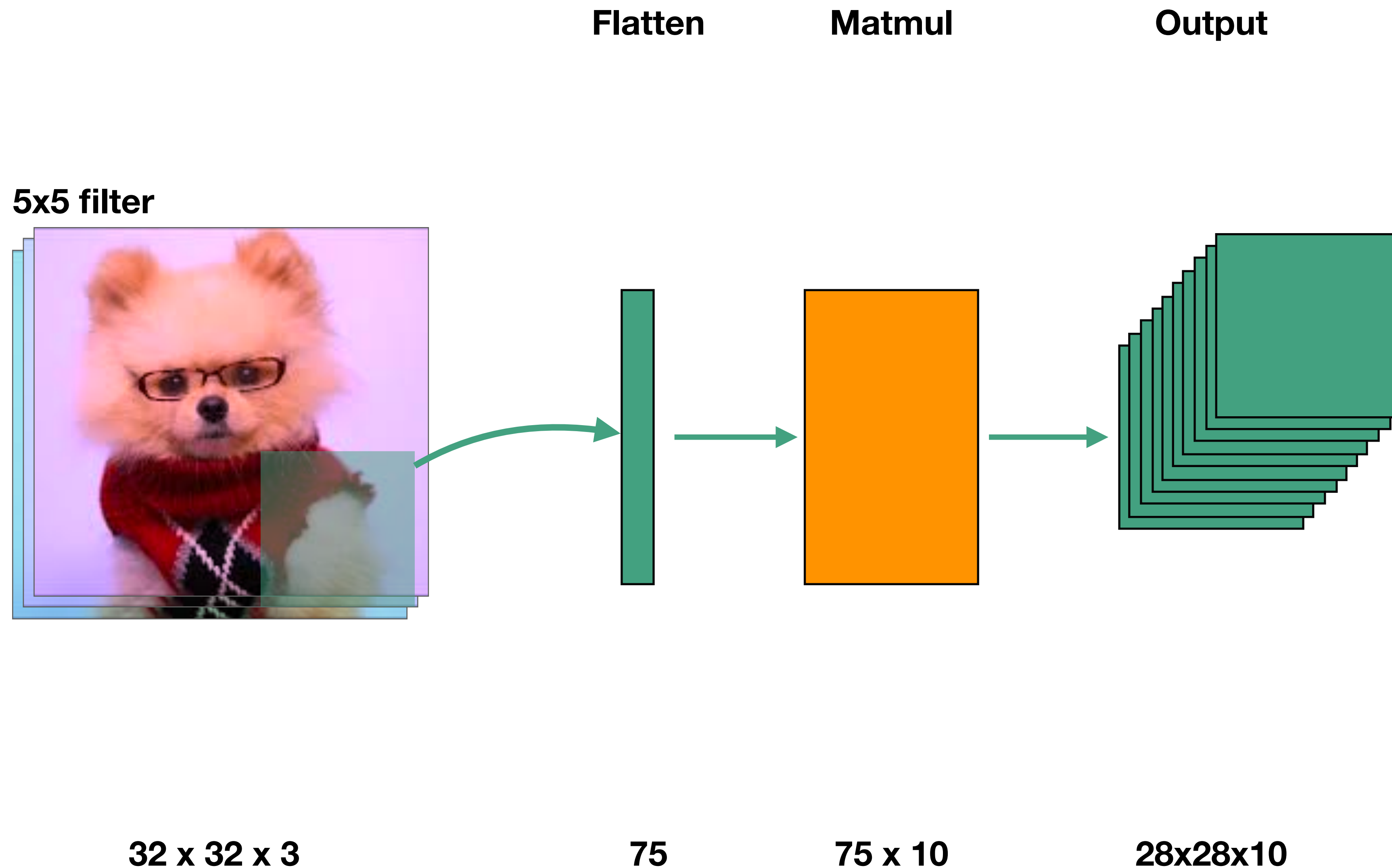
# Input can have multiple channels



# Input can have multiple channels

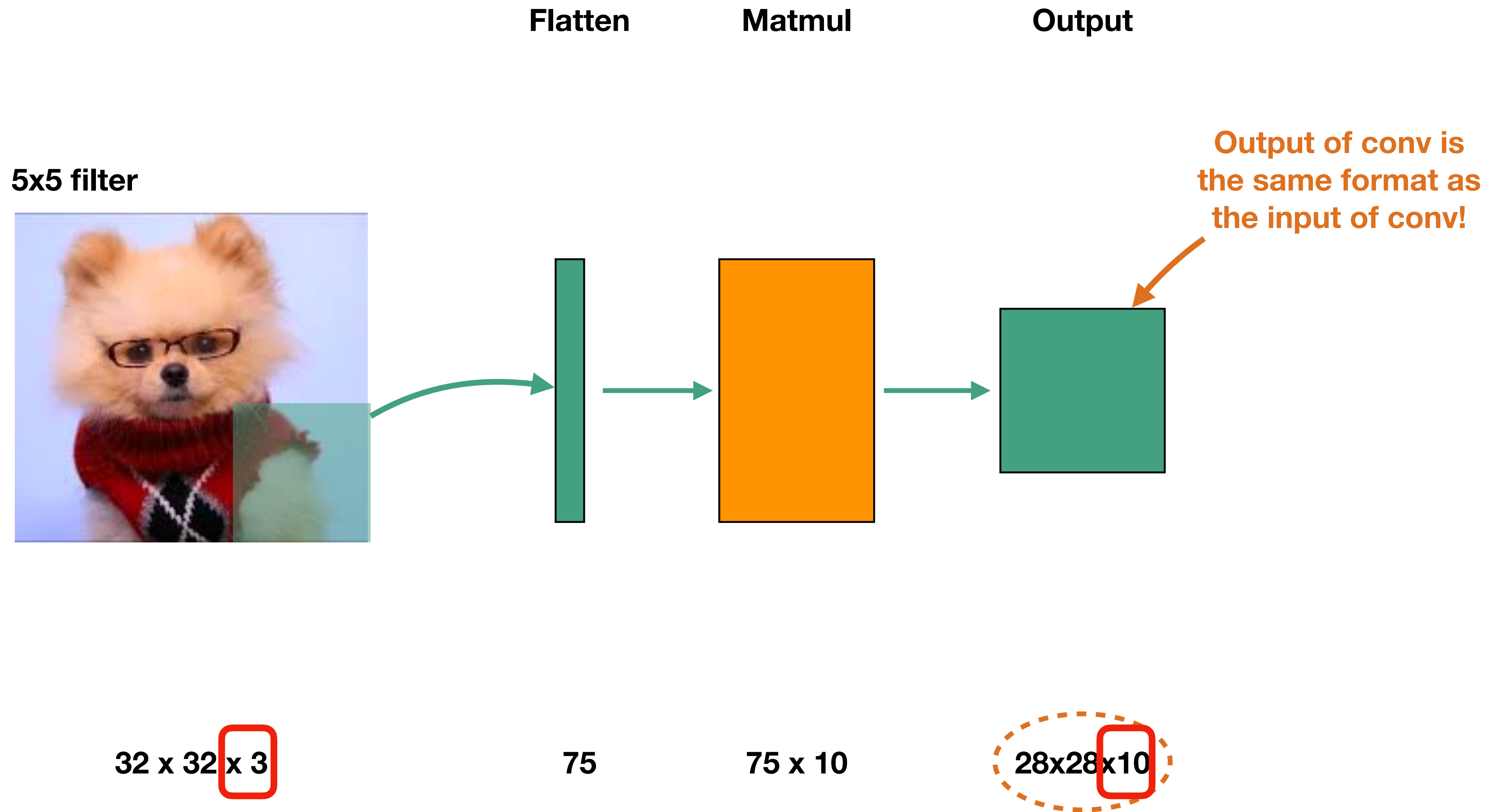


# Output can have multiple channels



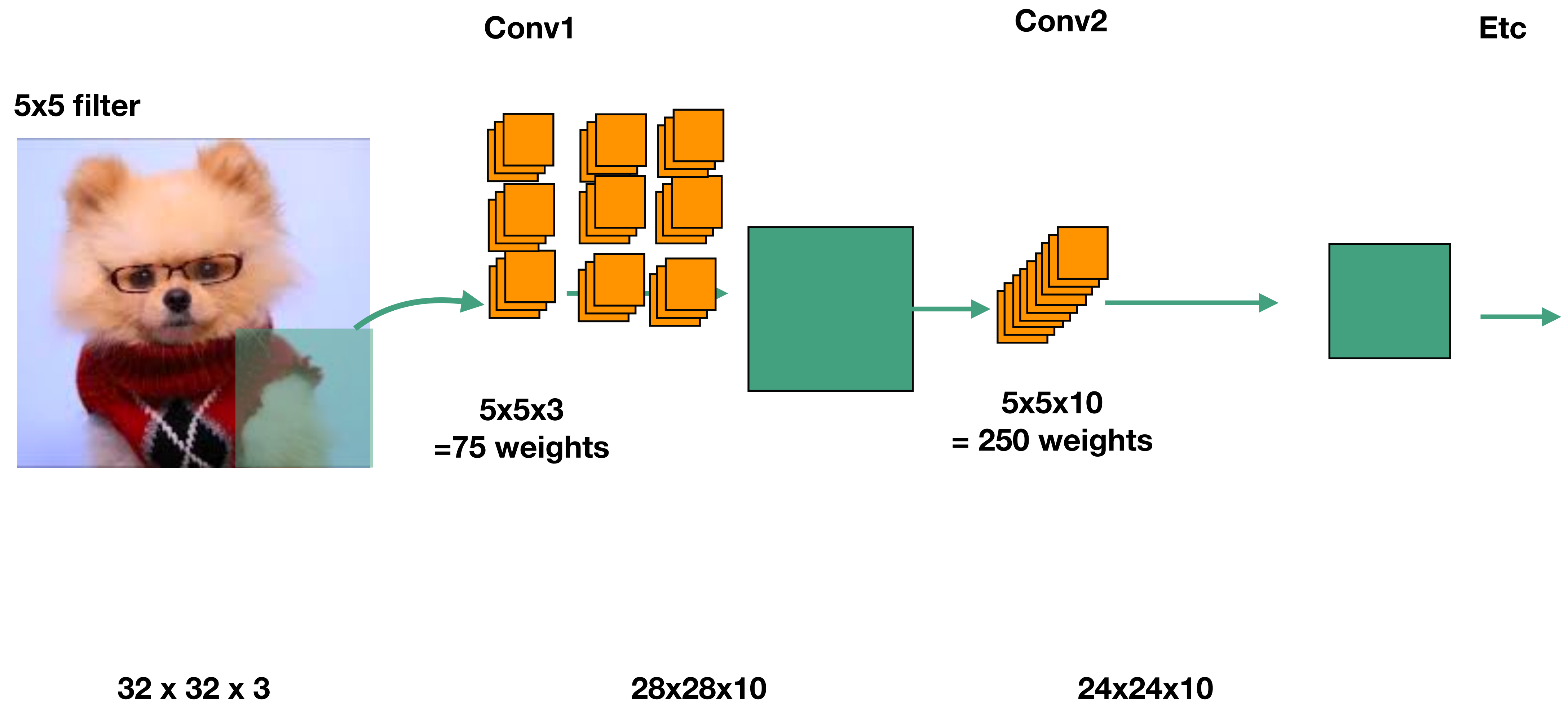


# Convolutional filter stacks

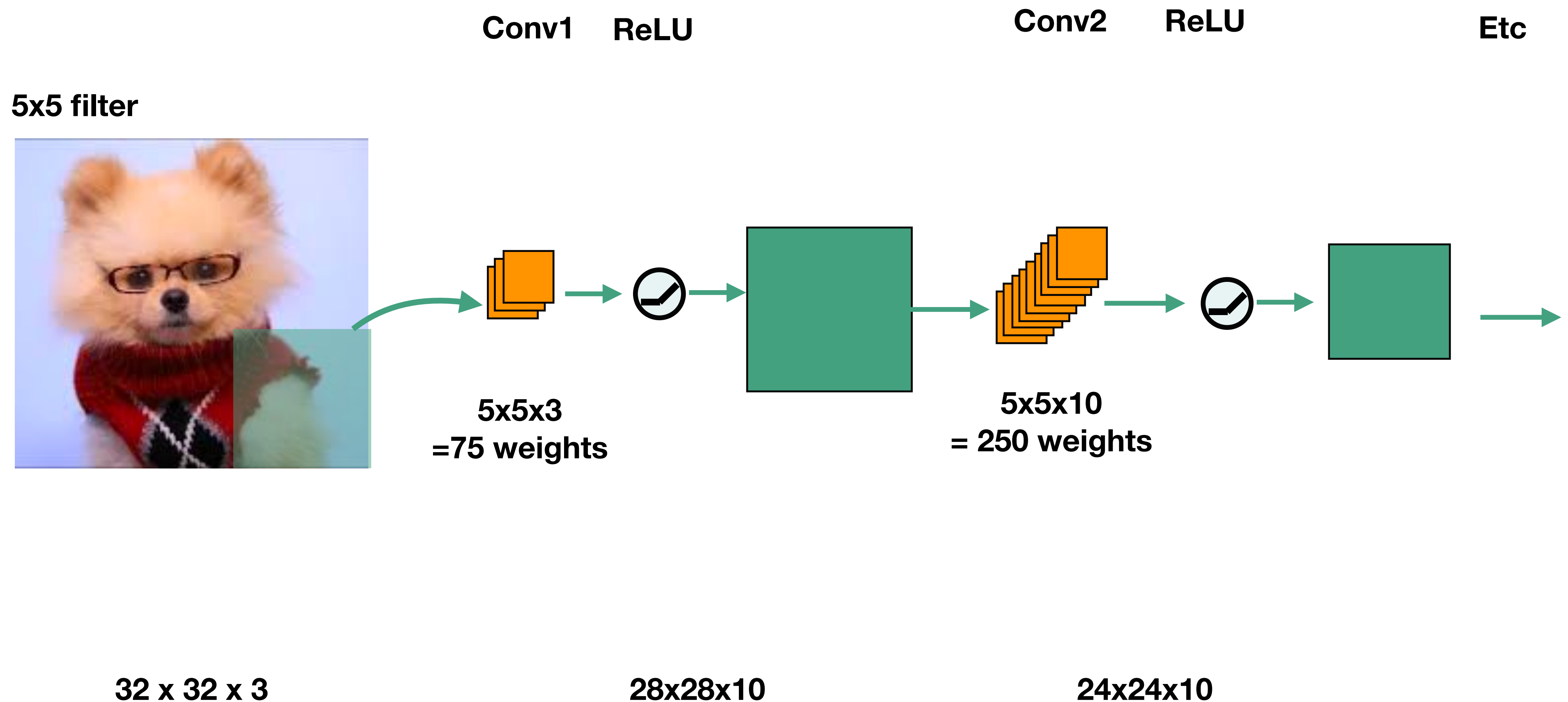




# Implication —> we can “stack” conv layers



Implication  $\rightarrow$  we can “stack” conv layers



# Questions?

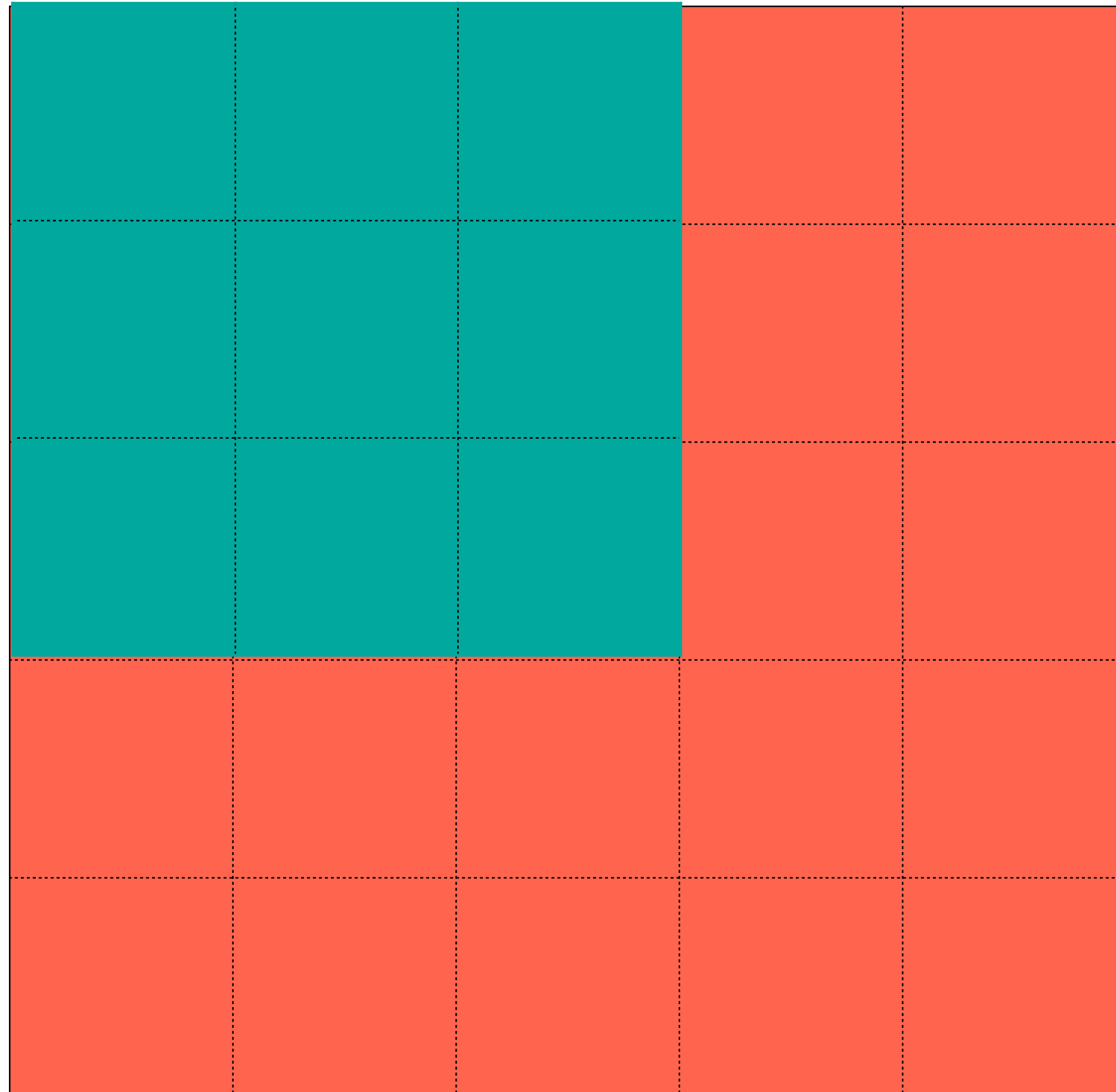
# Review of convolutions

- What's a convolutional filter?
- Filter stacks and ConvNets
- **Strides & padding**
- Filter math
- Implementation notes

# Strides

- Convolutions can subsample the image by jumping across some locations — this is called ‘stride’

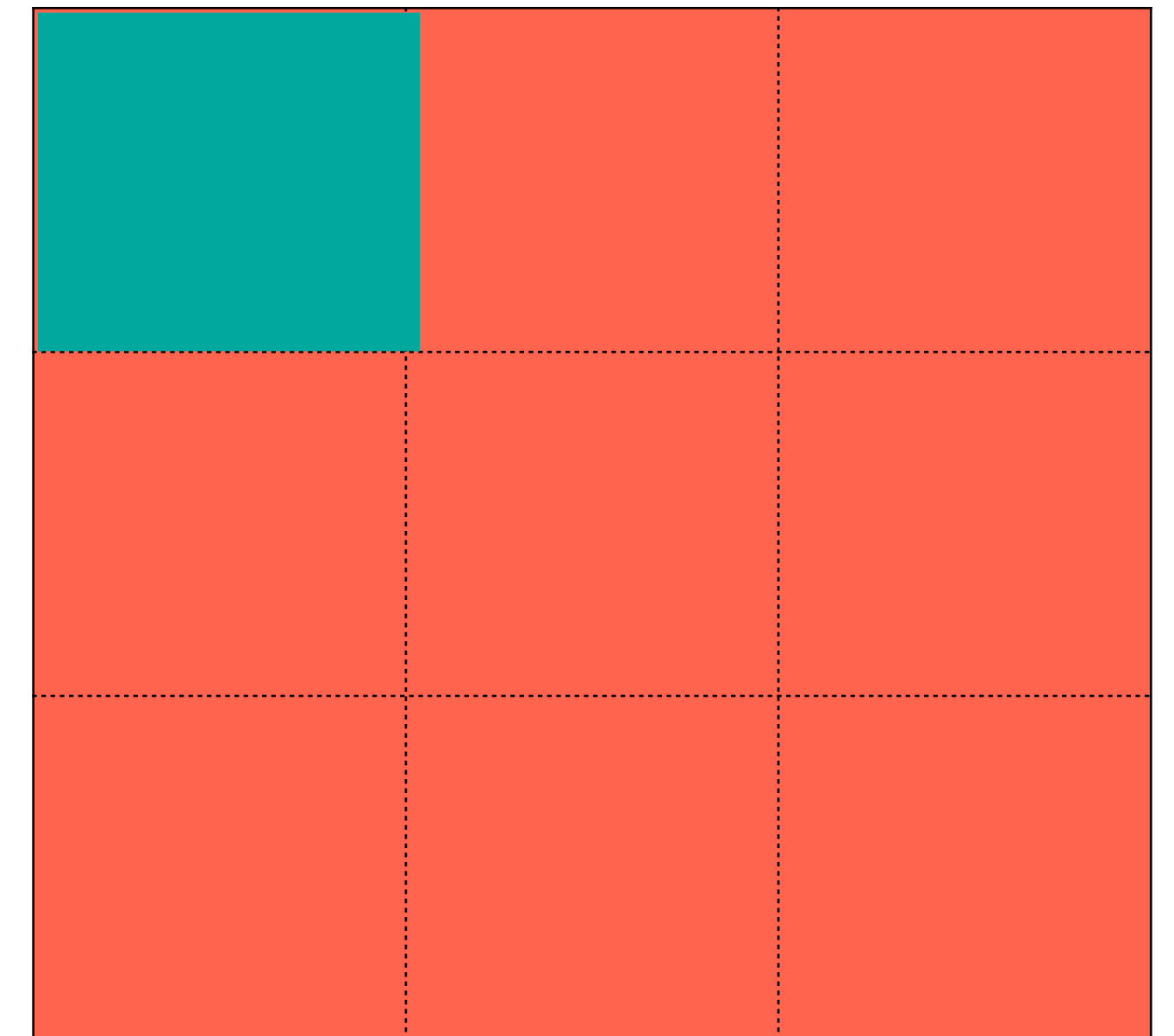
# Strides



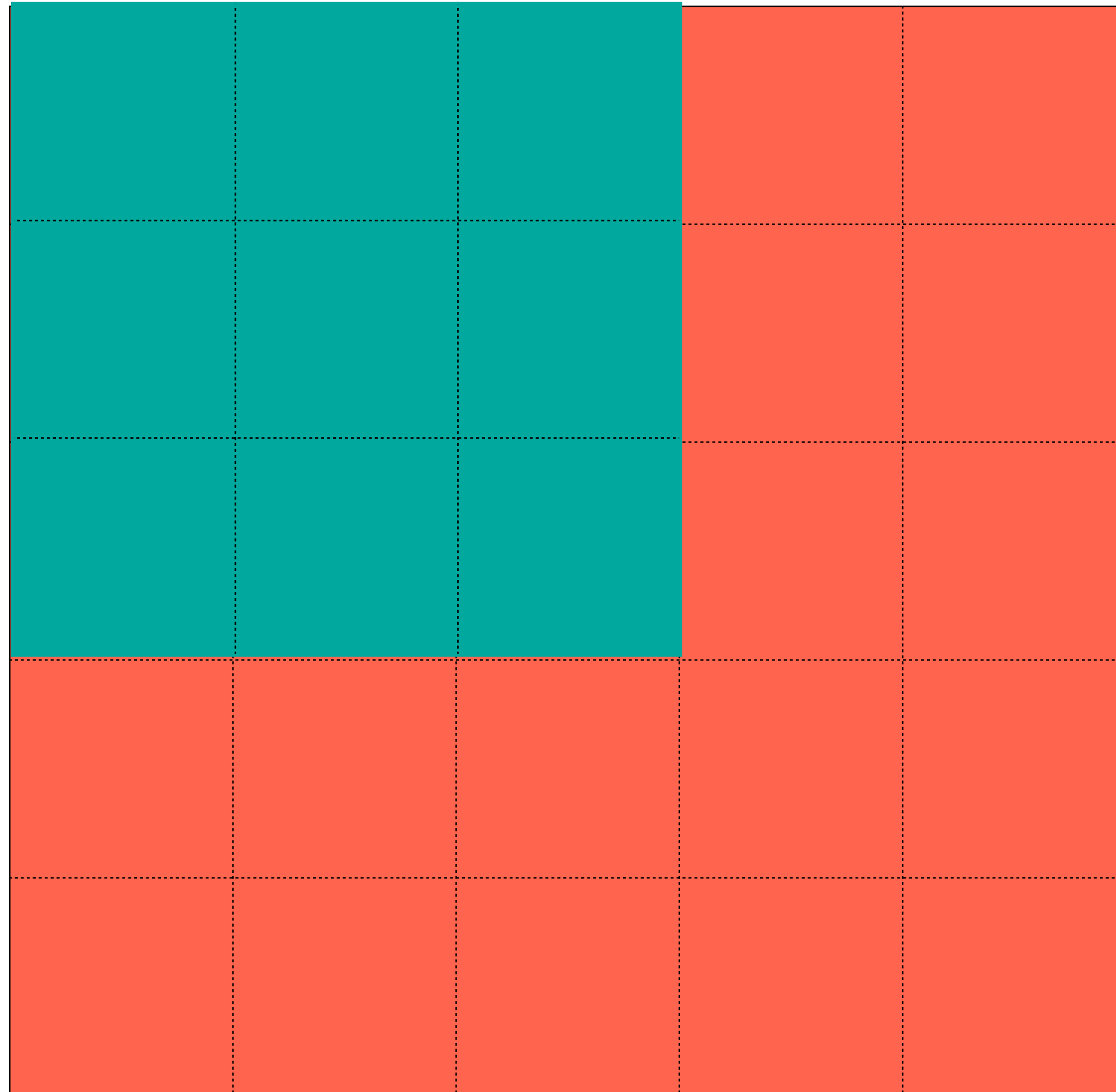
Conv2D

Filter = (3, 3)

Stride = (1, 1)



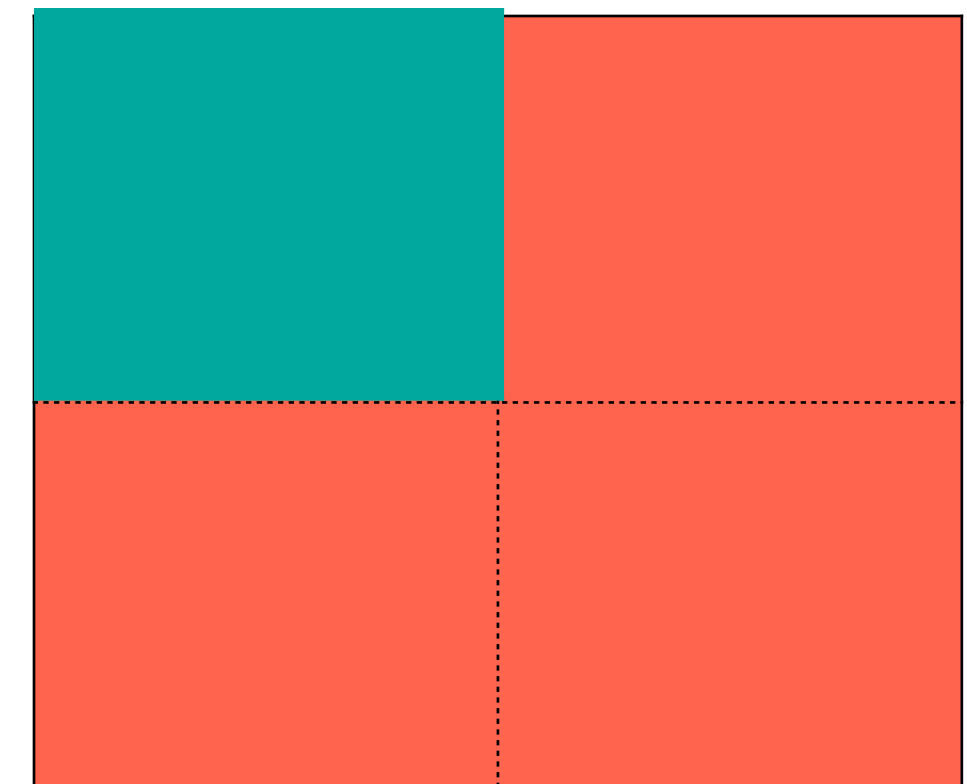
# Strides



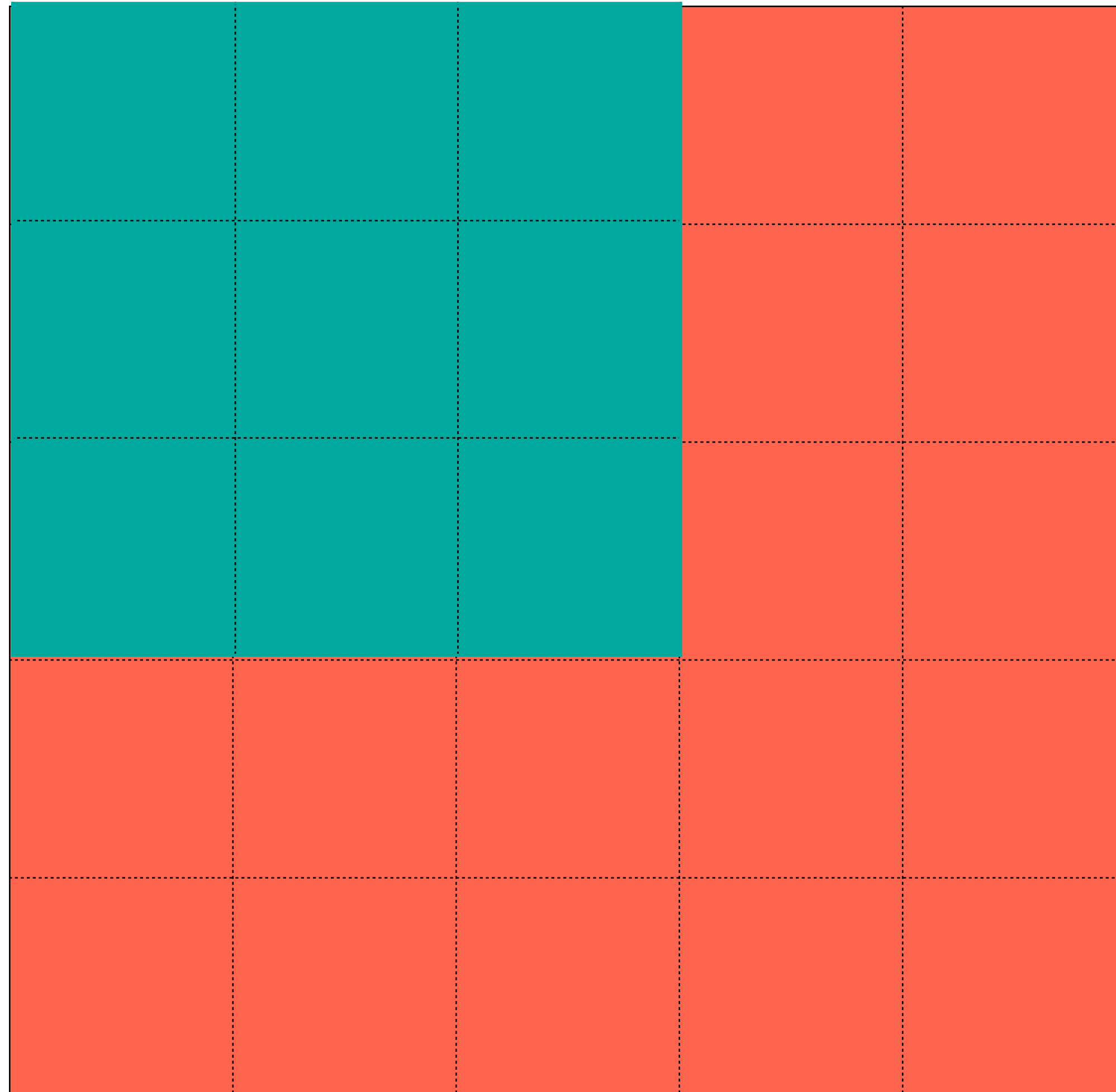
Conv2D

Filter = (3, 3)

**Stride = (2, 2)**



# Strides



Conv2D

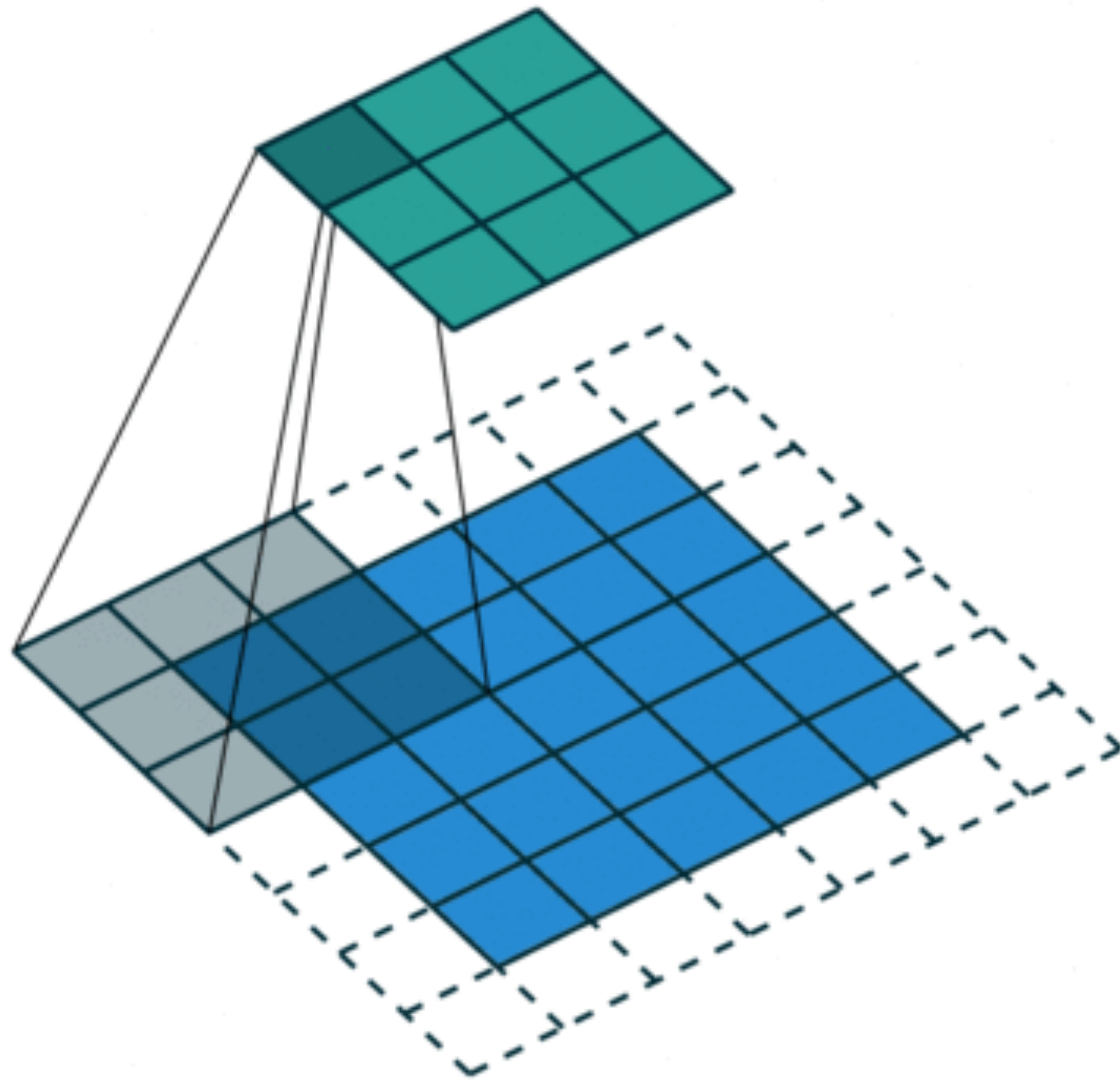
Filter = (3, 3)

**Stride = (3, 3)**

?



# Padding



- Padding solves the problem of filters running out of image
- Done by adding extra rows/cols to the input (usually set to 0)
- ‘SAME’ padding is illustrated here for filter=(3,3) with stride=(2,2)
- Not padding is called ‘VALID’ padding


# Review of convolutions

- What's a convolutional filter?
- Filter stacks and ConvNets
- Strides & padding
- **Filter math**
- Implementation notes


# Conv2D Math

- Input:  $W \times H \times D$  volume
- Parameters:
  - $K$  filters, each with size  $(F, F)$
  - ...moving at stride  $(S, S)$
  - ...with padding  $P$
- Output:  $W' \times H' \times K$  volume
  - $W' = (W - F + 2P) / S + 1$
  - $H' = (H - F + 2P) / S + 1$
- Each filter has  $(F * F * D)$  parameters
- $K * (F * F * D)$  total in the layer

# Conv2D Math

- Input:  $W \times H \times D$  volume
  - Parameters:
    - K filters, each with size  $(F, F)$
    - ...moving at stride  $(S, S)$
    - ...with padding  $P$
  - Output:  $W' \times H' \times K$  volume
    - $W' = (W - F + 2P) / S + 1$
    - $H' = (H - F + 2P) / S + 1$
  - Each filter has  $(F * F * D)$  parameters
  - $K * (F * F * D)$  total in the layer
- 
- **Commonly set to powers of 2 (e.g. 32, 64, 128)**

# Conv2D Math

- Input:  $W \times H \times D$  volume
  - Parameters:
    - $K$  filters, each with size  $(F, F)$
    - ...moving at stride  $(S, S)$
    - ...with padding  $P$
  - Output:  $W' \times H' \times K$  volume
    - $W' = (W - F + 2P) / S + 1$
    - $H' = (H - F + 2P) / S + 1$
  - Each filter has  $(F * F * D)$  parameters
  - $K * (F * F * D)$  total in the layer
- 
- **Commonly  $(5, 5)$ ,  $(3, 3)$ ,  $(2, 2)$ ,  $(1, 1)$**

# Conv2D Math

- Input:  $W \times H \times D$  volume
  - Parameters:
    - $K$  filters, each with size  $(F, F)$
    - ...moving at stride  $(S, S)$
    - ...with padding  $P$
  - Output:  $W' \times H' \times K$  volume
    - $W' = (W - F + 2P) / S + 1$
    - $H' = (H - F + 2P) / S + 1$
  - Each filter has  $(F * F * D)$  parameters
  - $K * (F * F * D)$  total in the layer
- • **'SAME' sets it automatically**

# A guide to convolution arithmetic for deep learning

Vincent Dumoulin<sup>1★</sup> and Francesco Visin<sup>2★†</sup>

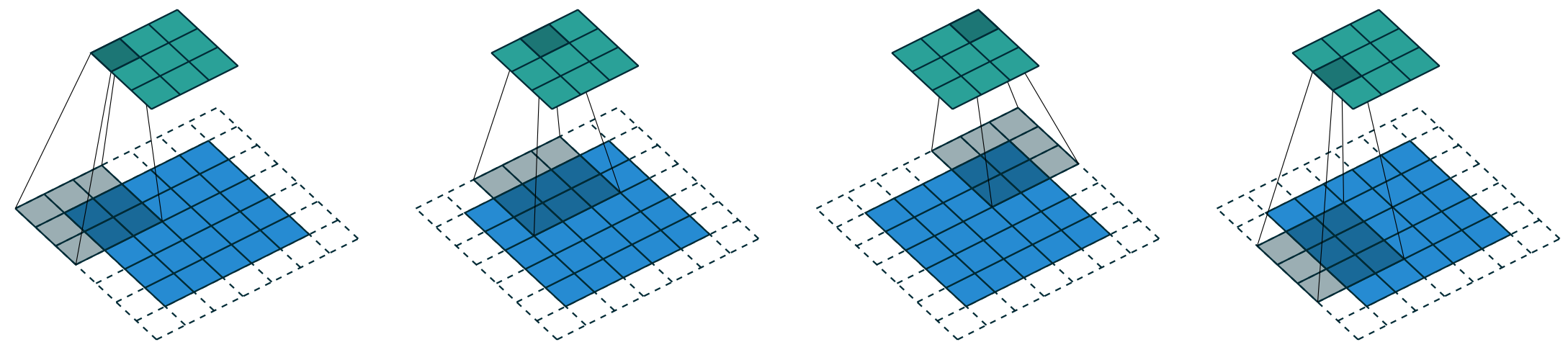


Figure 2.6: (Arbitrary padding and strides) Convolving a  $3 \times 3$  kernel over a  $5 \times 5$  input padded with a  $1 \times 1$  border of zeros using  $2 \times 2$  strides (i.e.,  $i = 5$ ,  $k = 3$ ,  $s = 2$  and  $p = 1$ ).

- Lots of cool visualizations and comforting equations

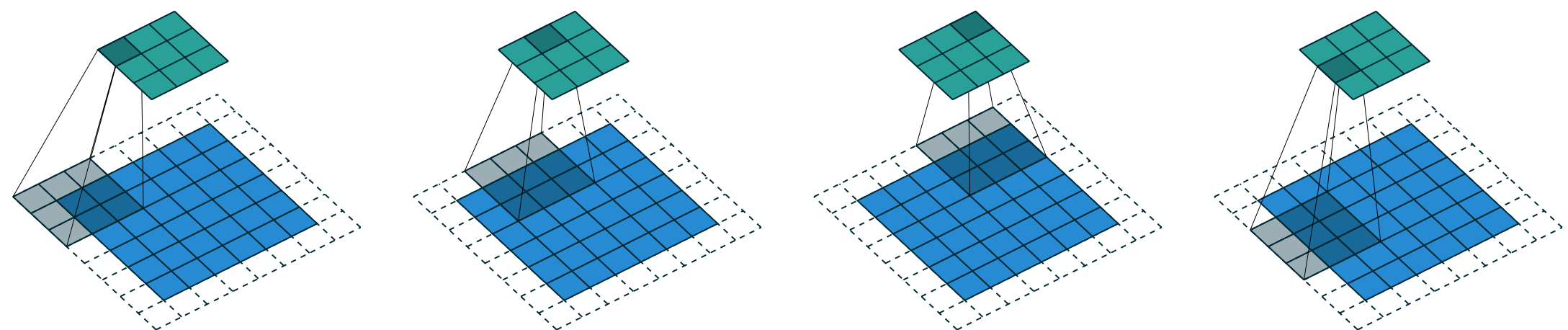


Figure 2.7: (Arbitrary padding and strides) Convolving a  $3 \times 3$  kernel over a  $6 \times 6$  input padded with a  $1 \times 1$  border of zeros using  $2 \times 2$  strides (i.e.,  $i = 6$ ,  $k = 3$ ,  $s = 2$  and  $p = 1$ ). In this case, the bottom row and right column of the zero padded input are not covered by the kernel.

# Review of convolutions

- What's a convolutional filter?
- Filter stacks and ConvNets
- Strides & padding
- Filter math
- **Implementation notes**

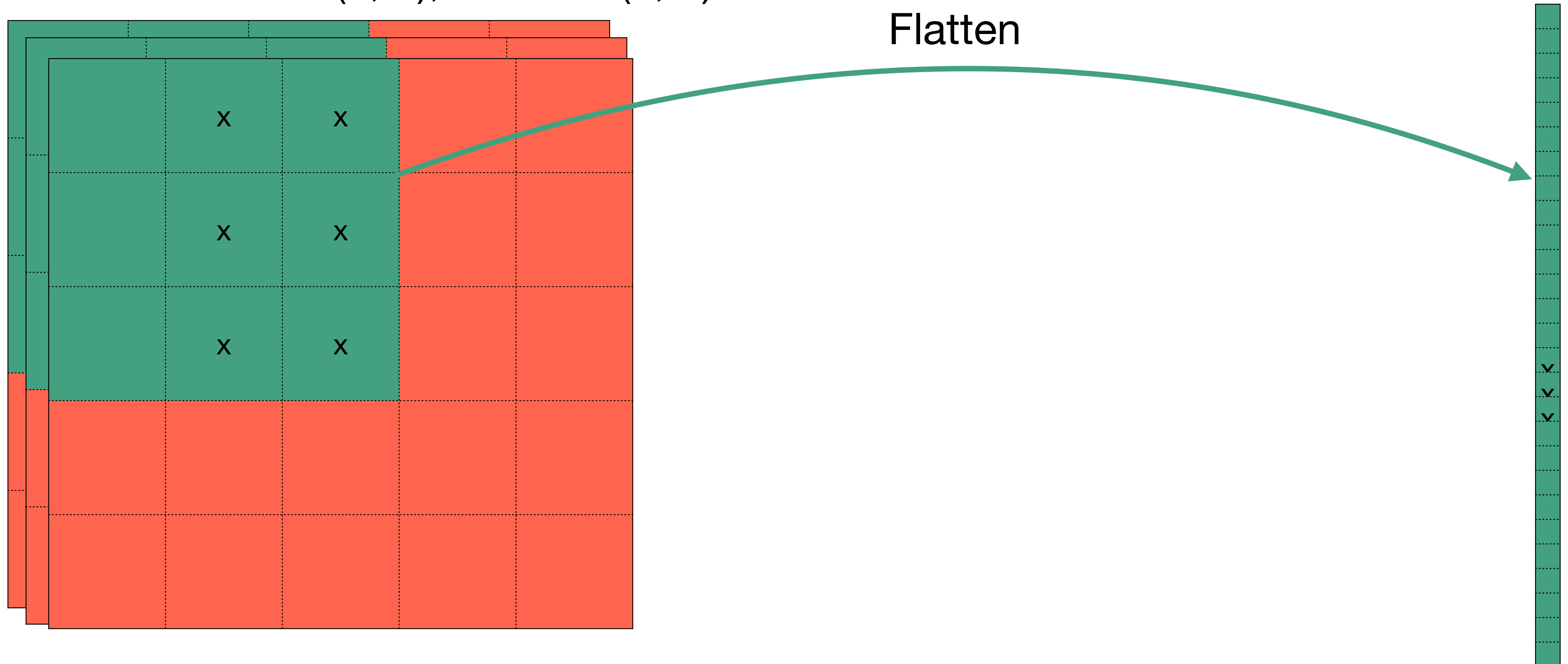


# Convolution implementation

Conv2D. Input = (5, 5, 3)

Filters = 32 of size (3, 3), Stride = (1, 1)

$3 * 3 * 3 = 27$ -dimensional

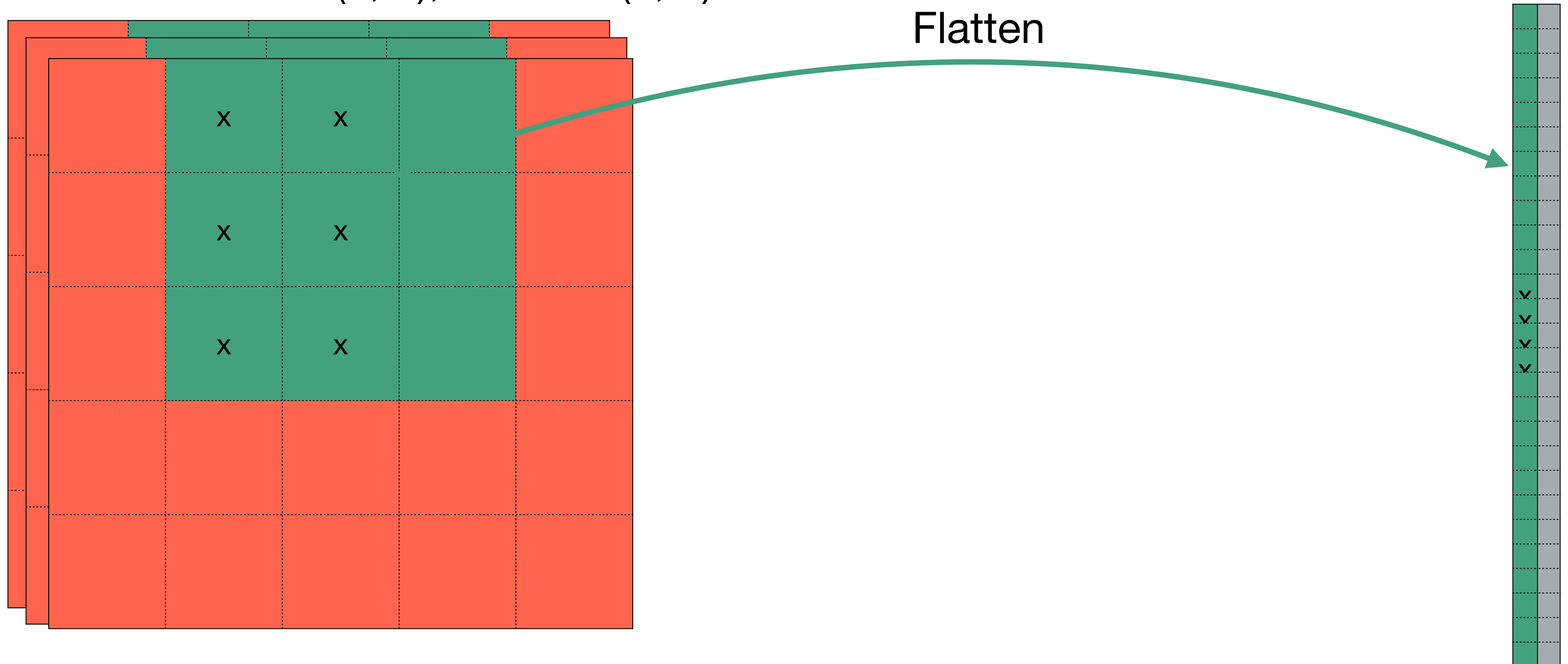


# Convolution implementation

Conv2D. Input = (5, 5, 3)

Filters = 32 of size (3, 3), Stride = (1, 1)

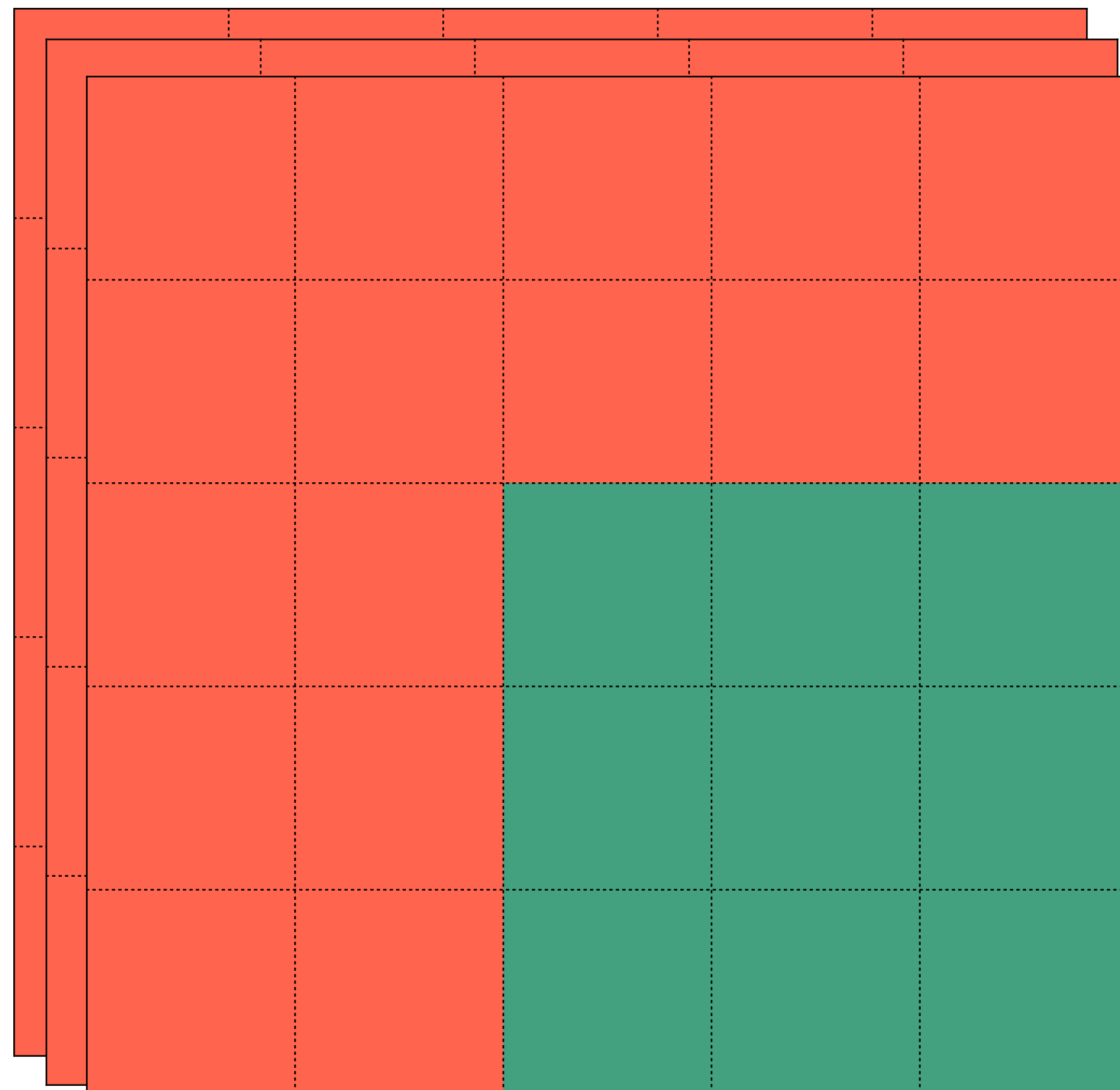
$3 * 3 * 3 = 27$ -dimensional



**...sliding continues...**

# Convolution implementation

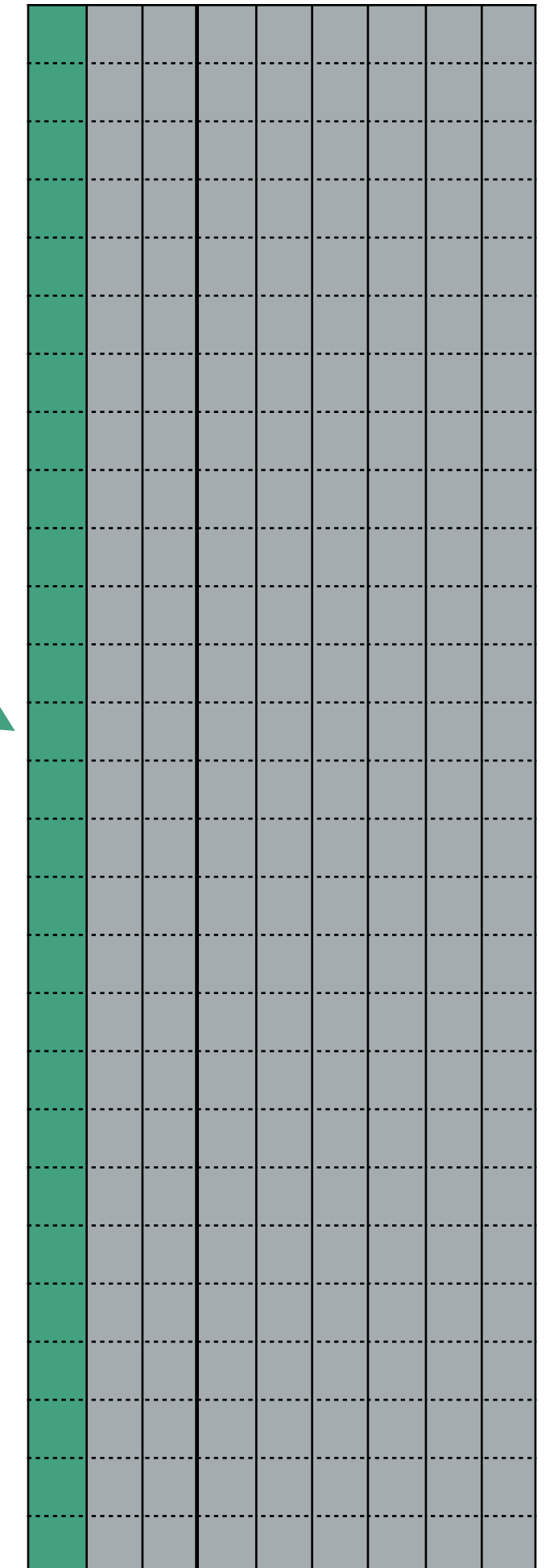
Conv2D. Input = (5, 5, 3)  
Filters = 32 of size (3, 3), Stride = (1, 1)



Flatten

**Im2Col**

X\_col  
(27 x 9)



*3x3 filter, 3-channel input*

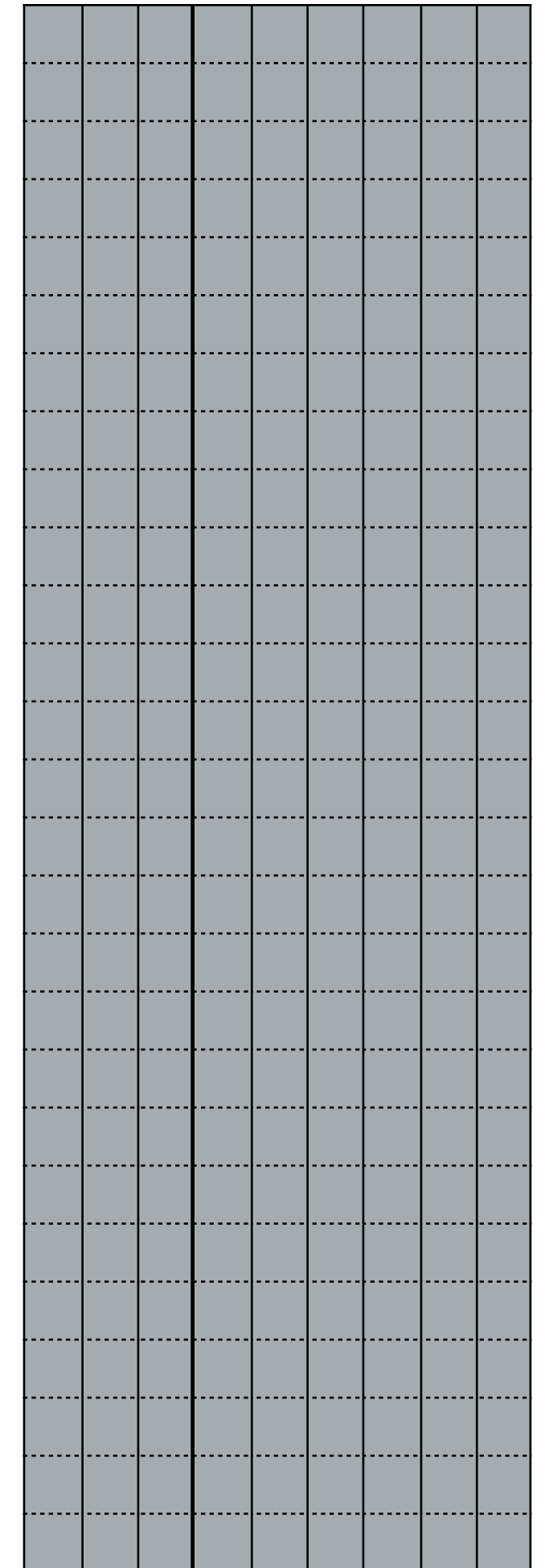
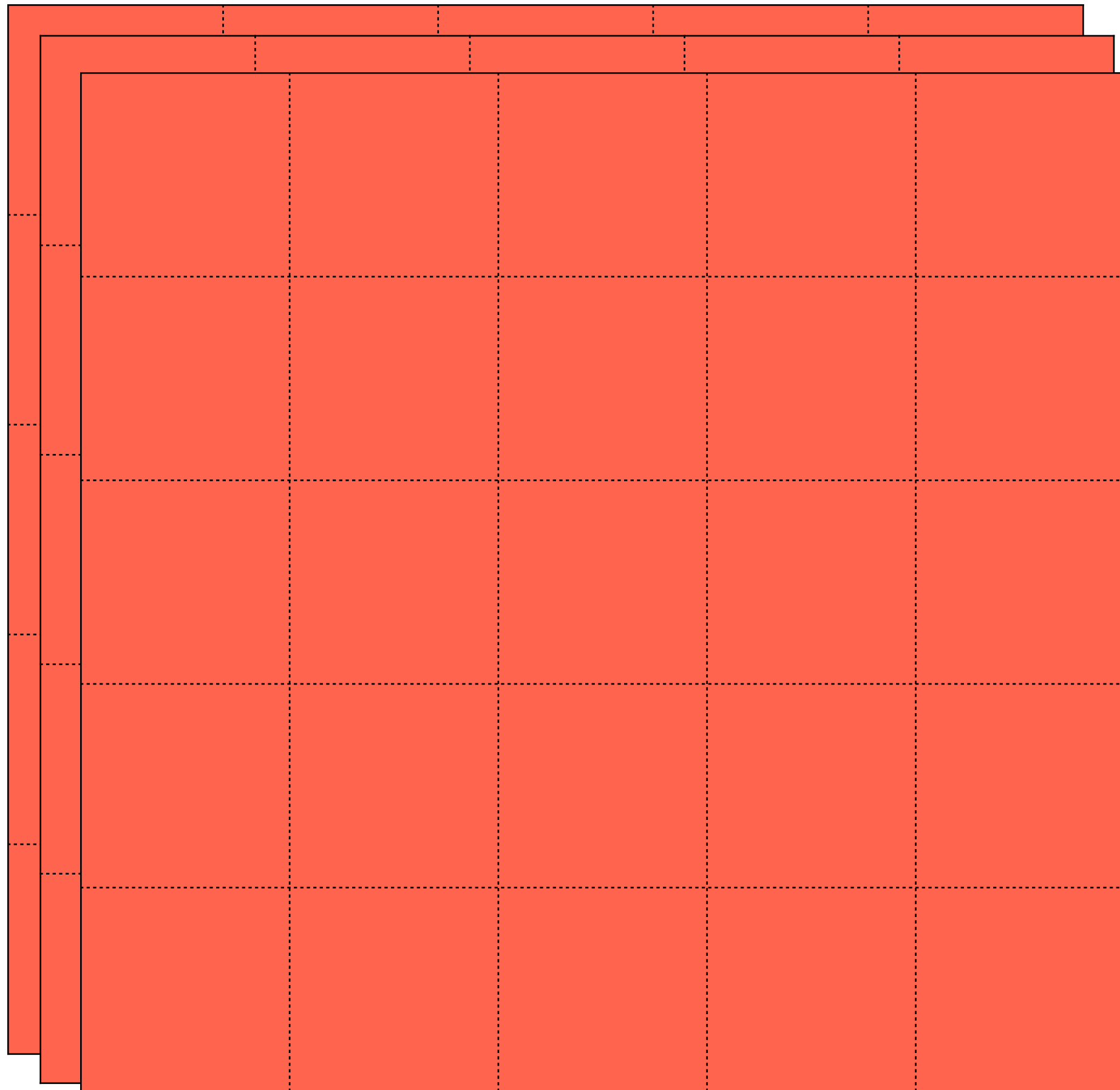
Conv2D. Input = (5, 5, 3)  
**Filters = 32 of size (3, 3), Stride = (1, 1)**



W\_row  
(32 x 27)



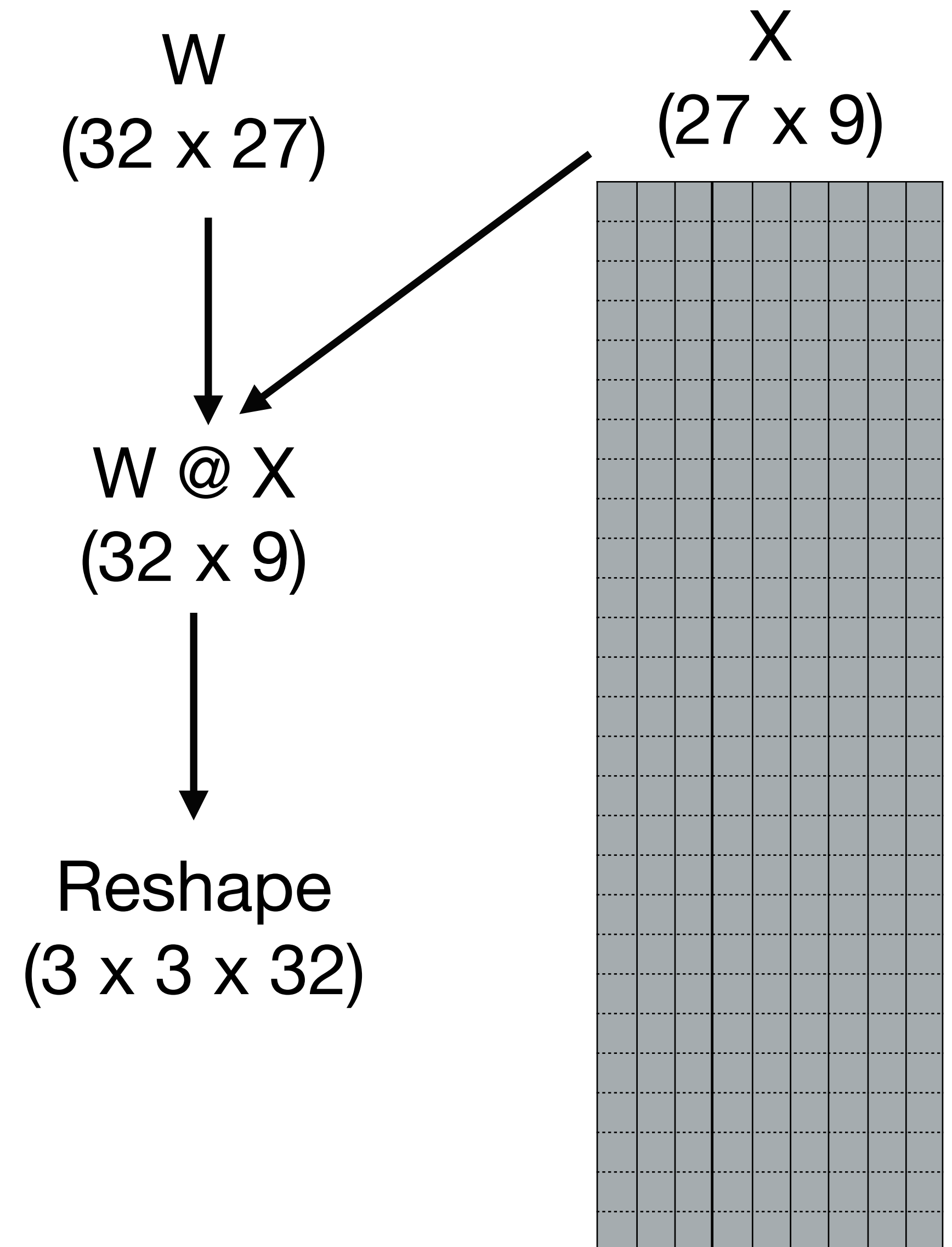
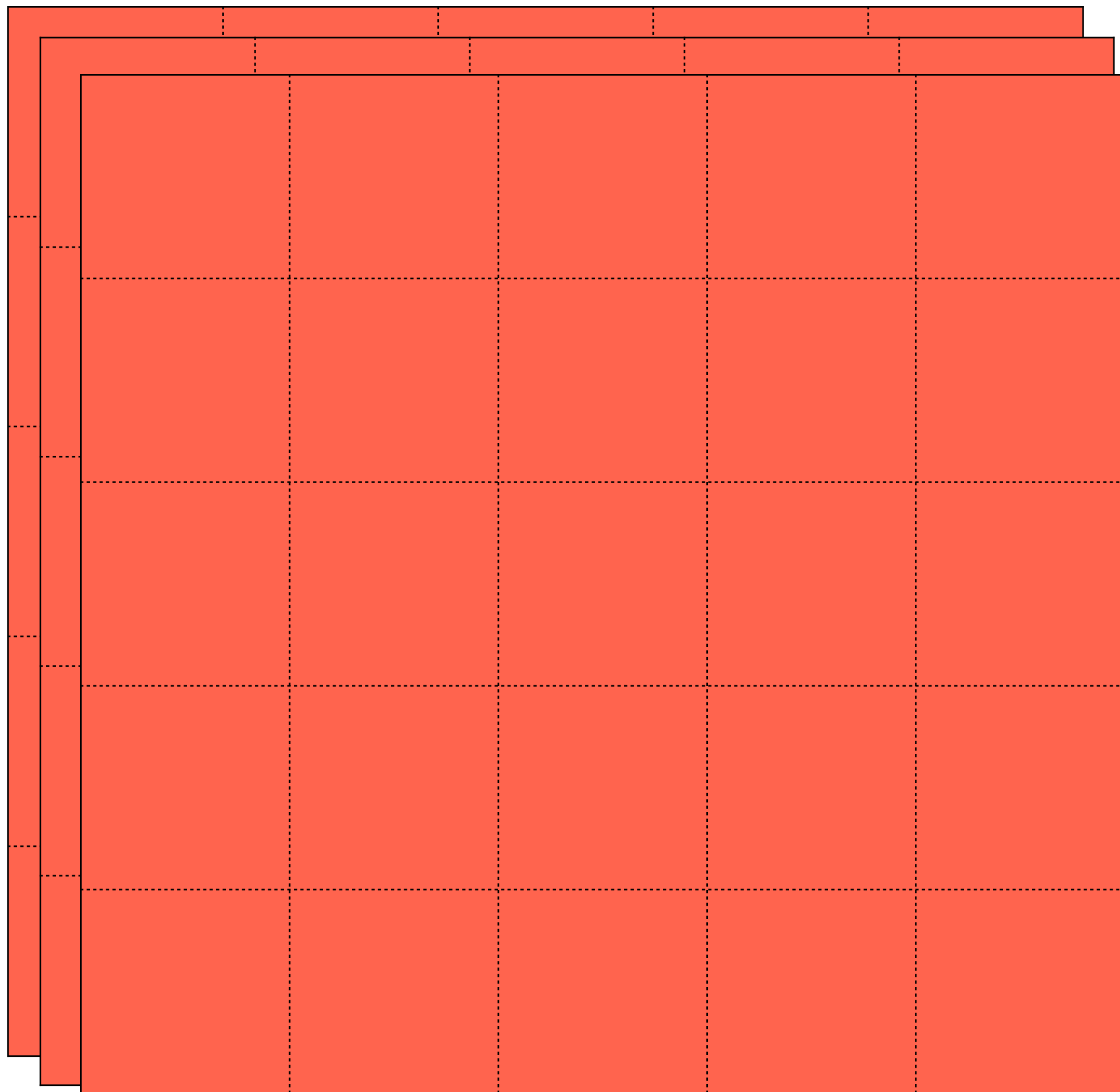
X\_col  
(27 x 9)



# Convolution implementation

Conv2D. Input = (5, 5, 3)

**Filters = 32 of size (3, 3), Stride = (1, 1)**



# Questions?

# Agenda

1. Review of the convolution operation
- 2. Other important operations for ConvNets**
3. Classic ConvNet architectures

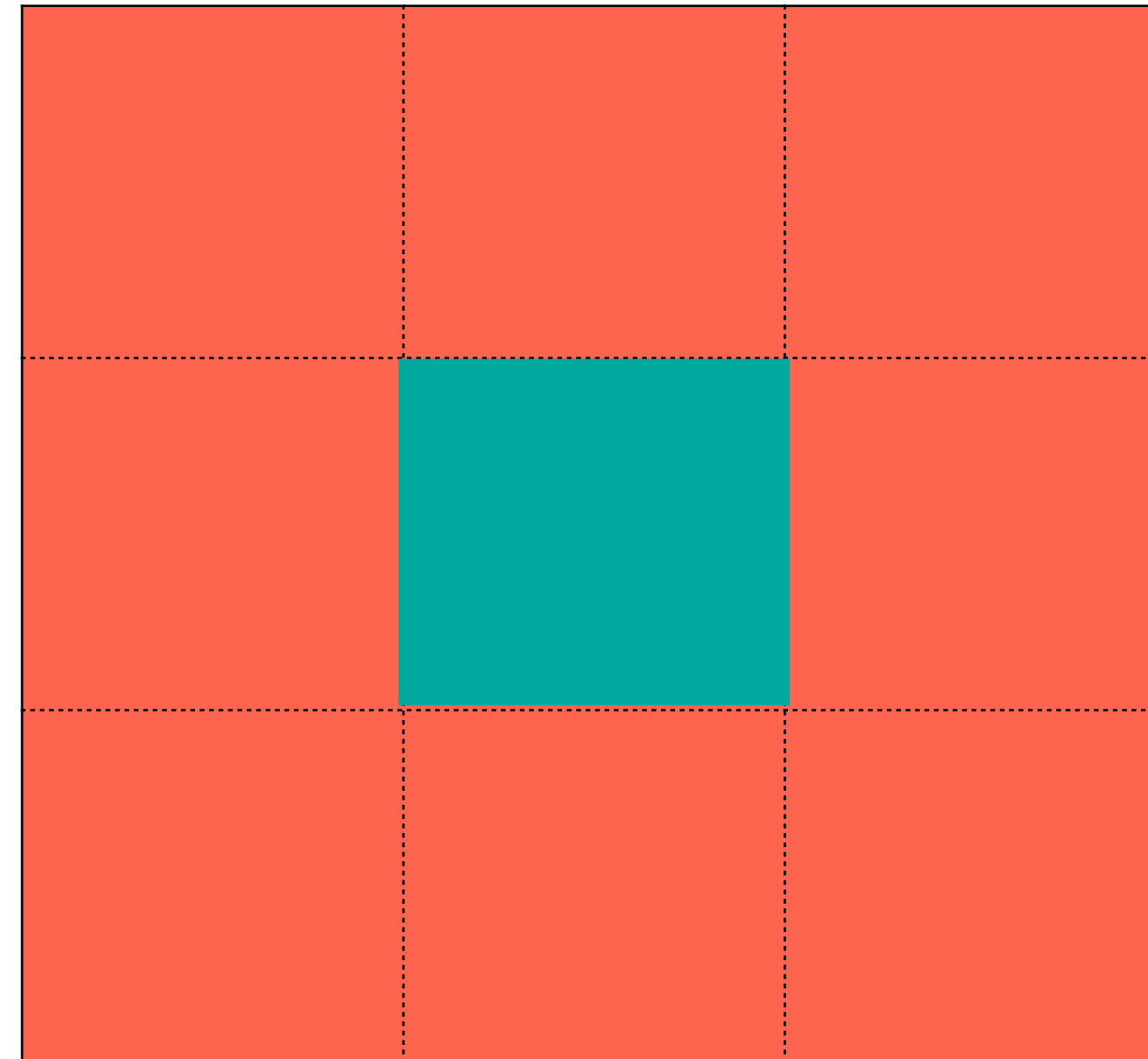
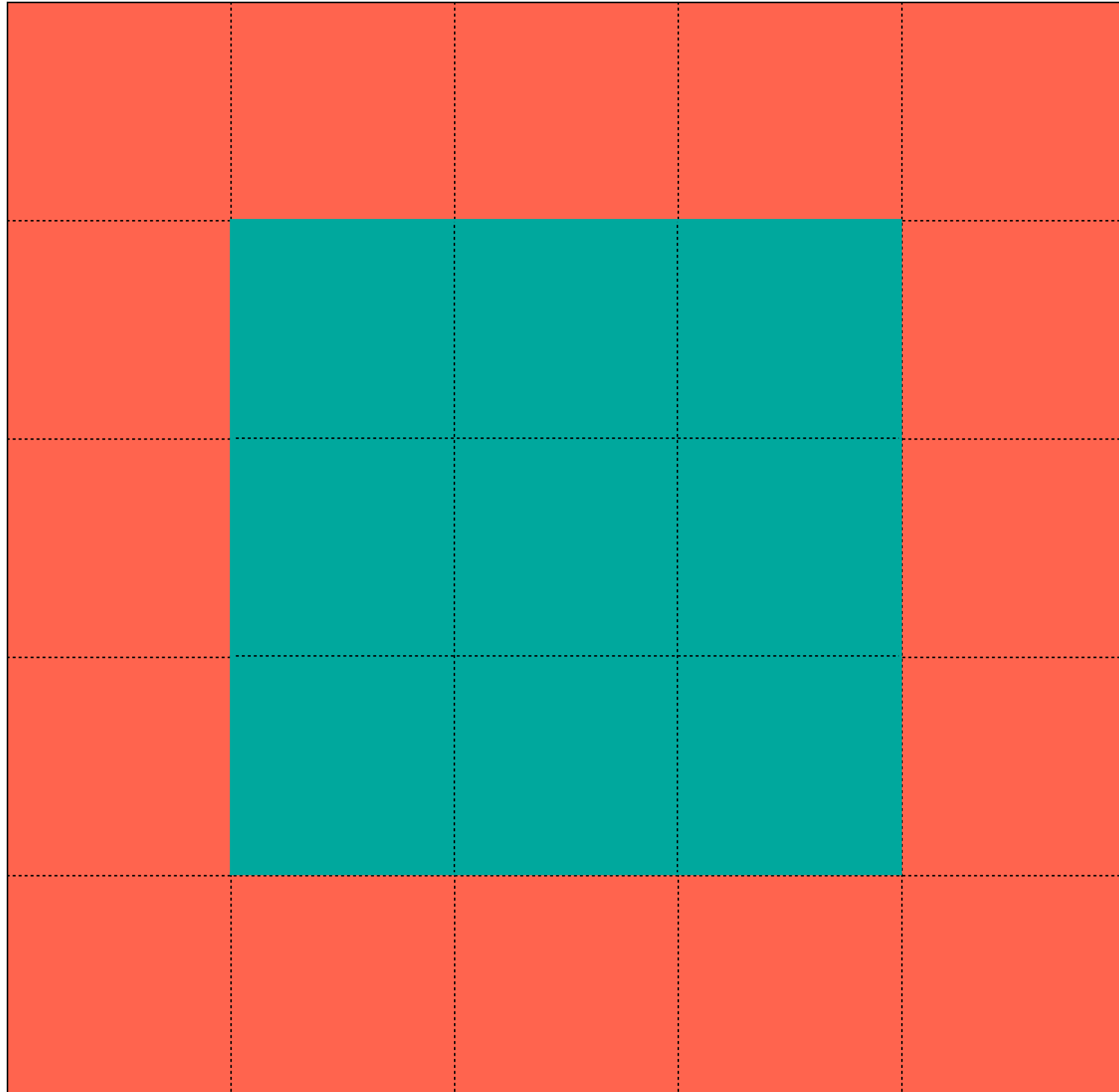


# Other important ConvNet operations

- **Increasing the receptive field (dilated convolutions)**
- Decreasing the size of the tensor
  - Pooling
  - 1x1-convolutions

# Receptive fields

Receptive field: 3x3

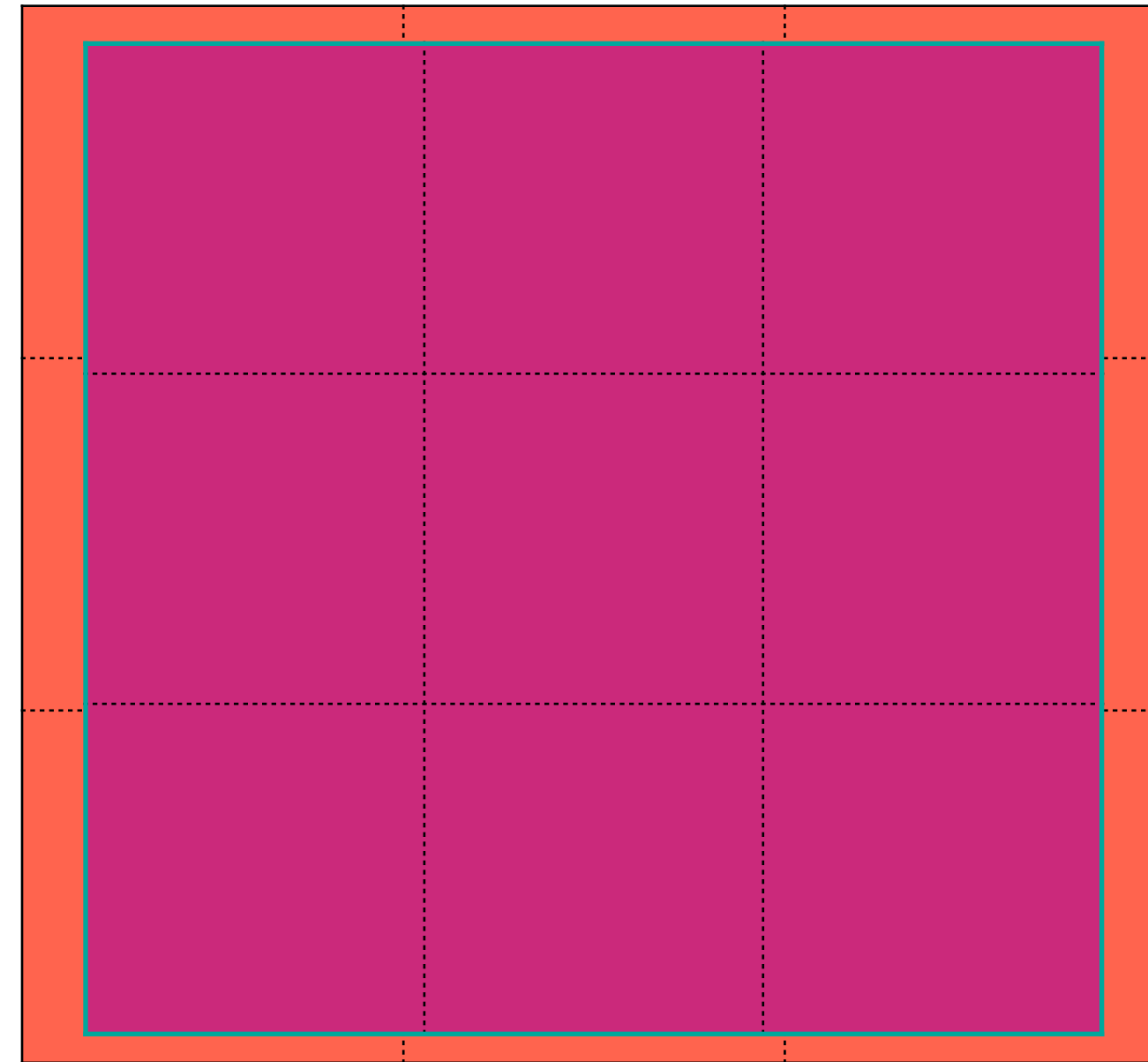
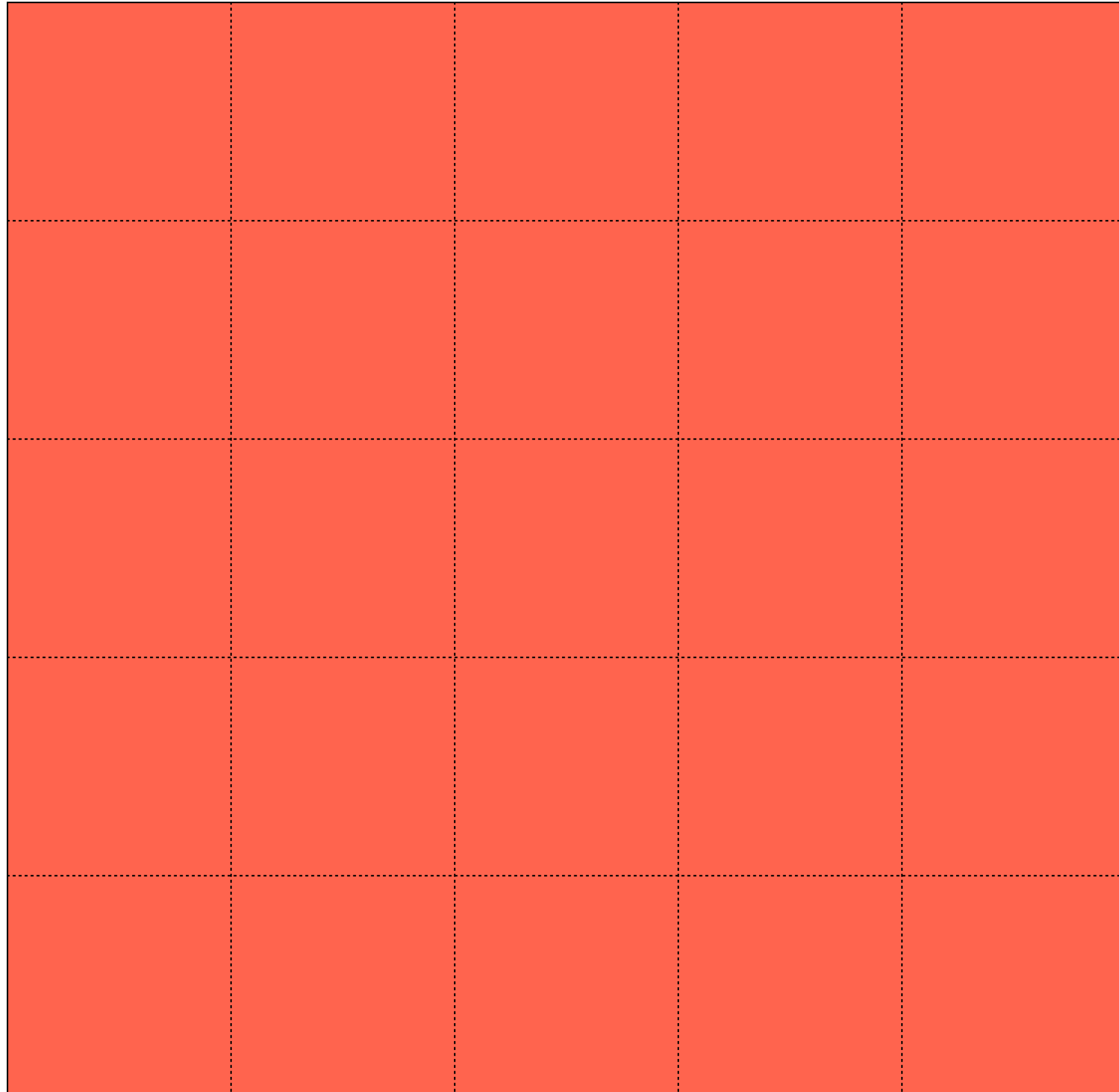


Conv2D

Filter = (3, 3)

Stride = (1, 1)

# Receptive fields



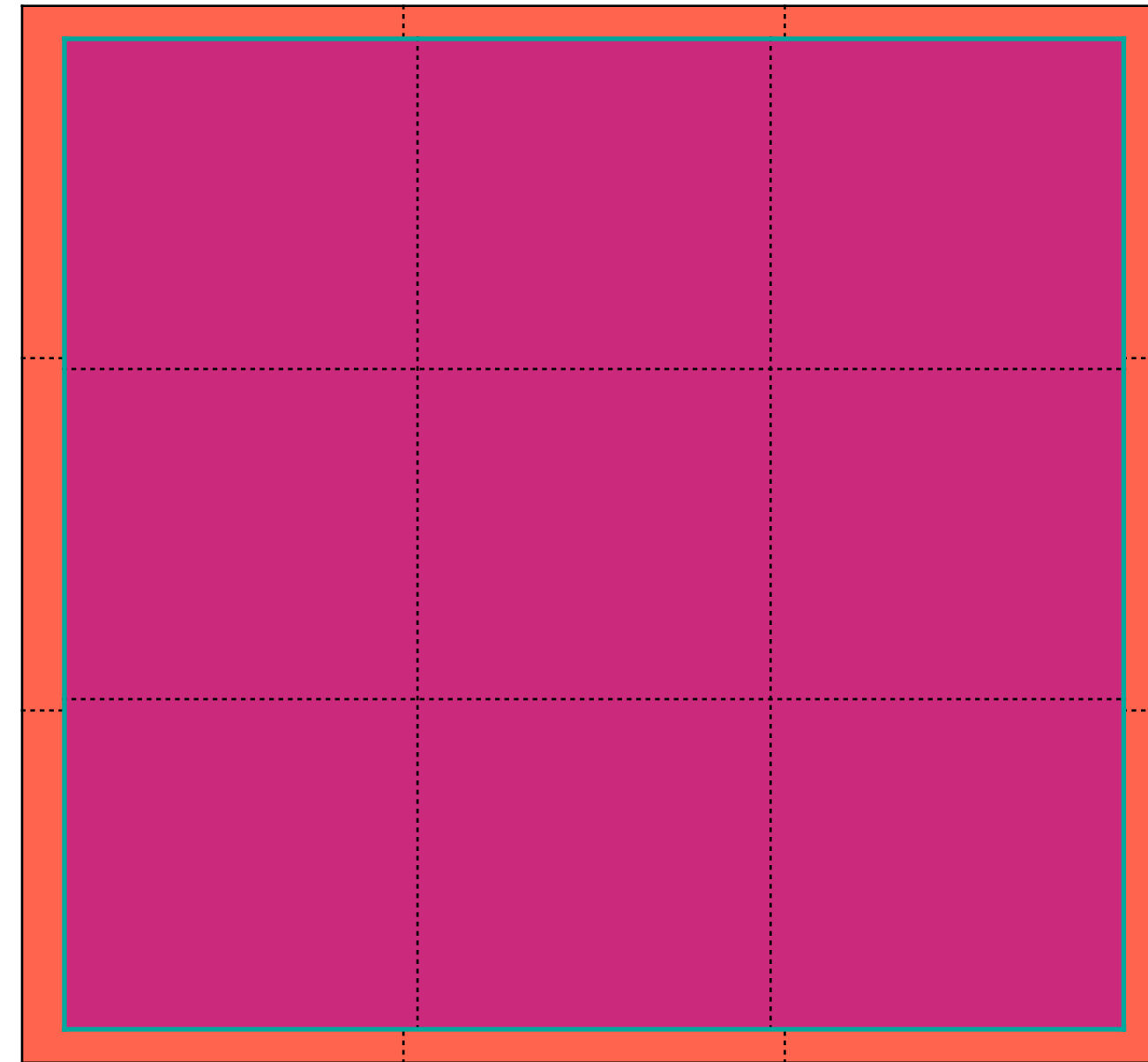
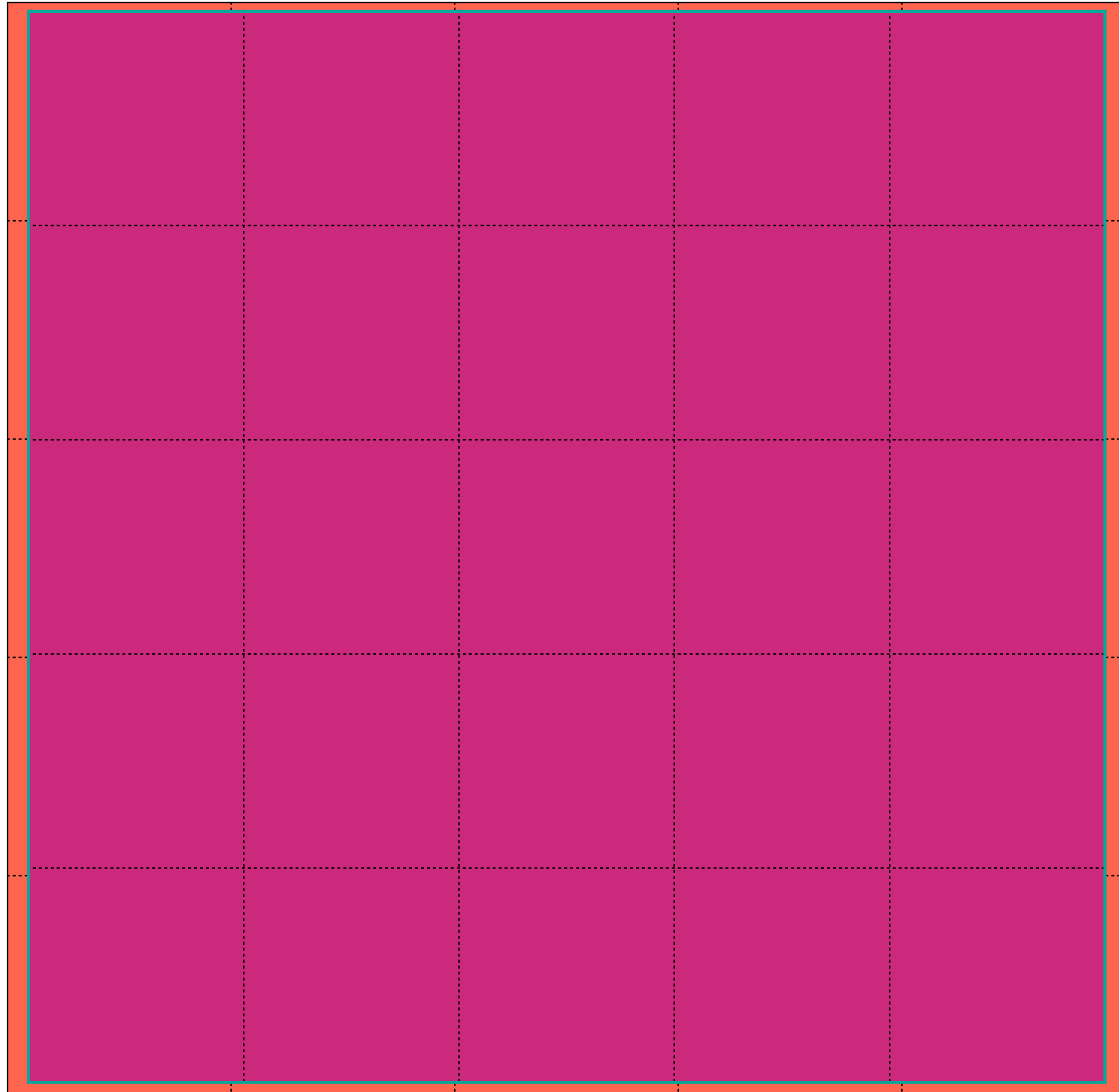
Conv2D  
Filter = (3, 3)  
Stride = (1, 1)



Conv2D  
Filter = (3, 3)  
Stride = (1, 1)

# Receptive fields

Original receptive field: 5x5



Conv2D  
Filter = (3, 3)  
Stride = (1, 1)

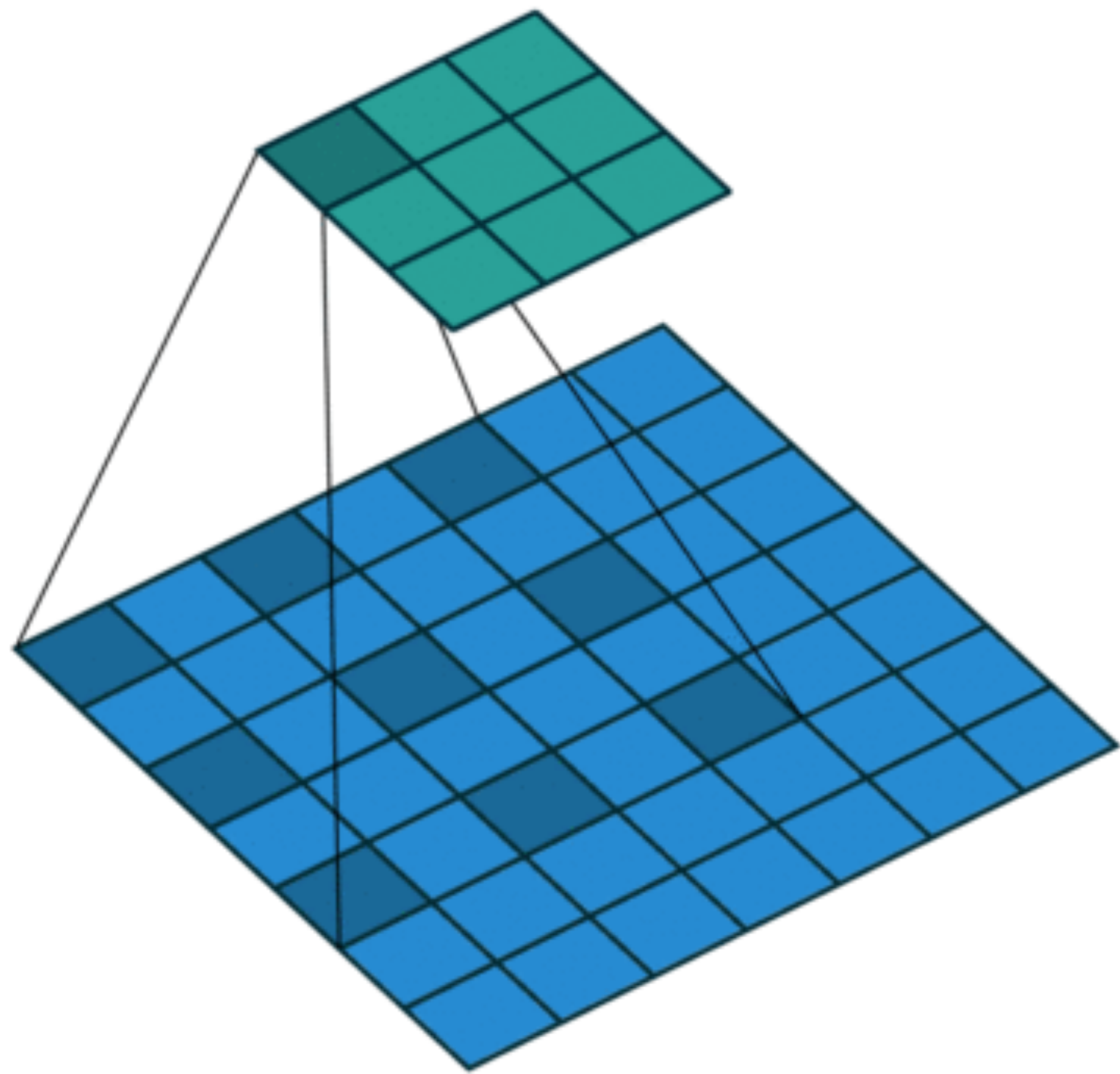


Conv2D  
Filter = (3, 3)  
Stride = (1, 1)

# Receptive fields

- Stacking convolutions one after the other increases the original receptive field: two (3, 3) convs get to a (5, 5) receptive field
  - (and tend to perform better than a single (5, 5) conv)
  - (with fewer parameters!)

# Dilated Convolution



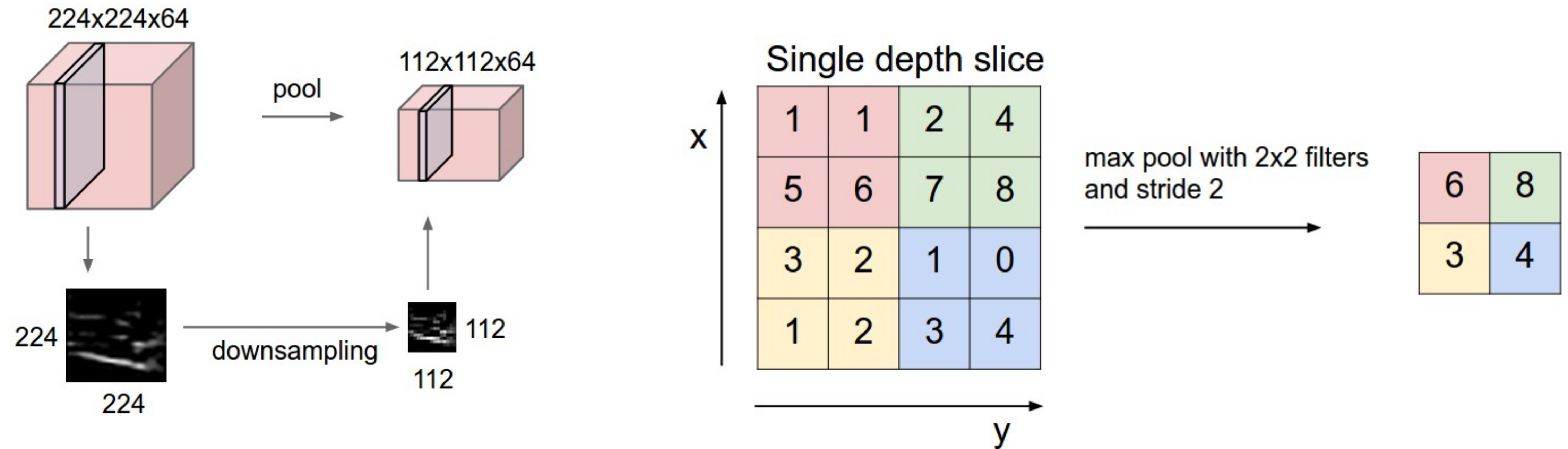
- Dilated convolutions can “see” a greater portion of the image by skipping pixels
- The  $(3, 3)$  1-dilated convolution illustrated here has a  $(5, 5)$  receptive field
- Stacking dilated convolutions up quickly gets to large receptive fields

# Other important ConvNet operations

- Increasing the receptive field  
(dilated convolutions)
- **Decreasing the size of the tensor**
  - Pooling
  - 1x1-convolutions



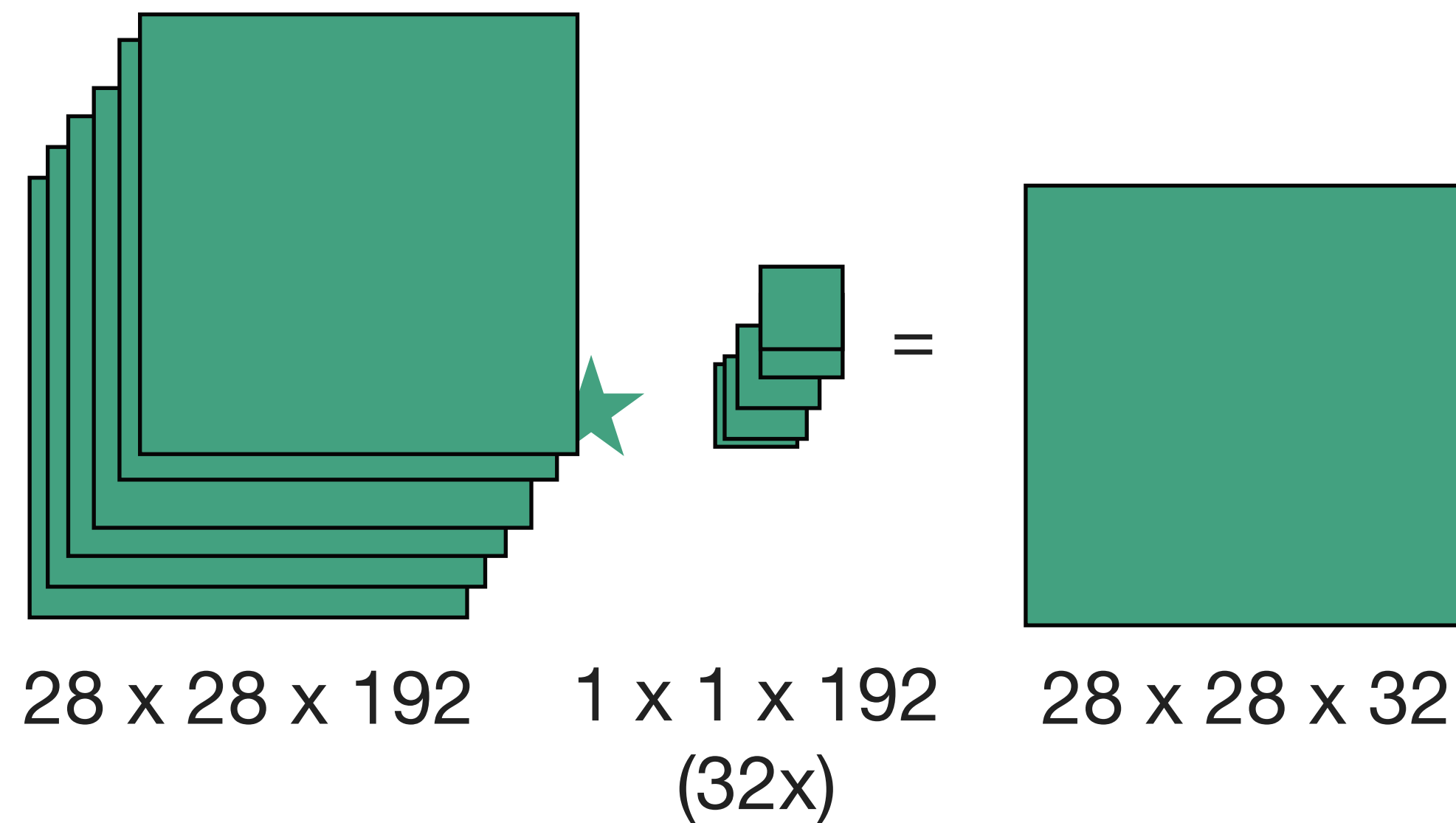
# Pooling



- Subsamples the image through average or max of region
- $2 \times 2$  max pooling is most common
- Recently fallen out of favor



# 1x1 Convolution



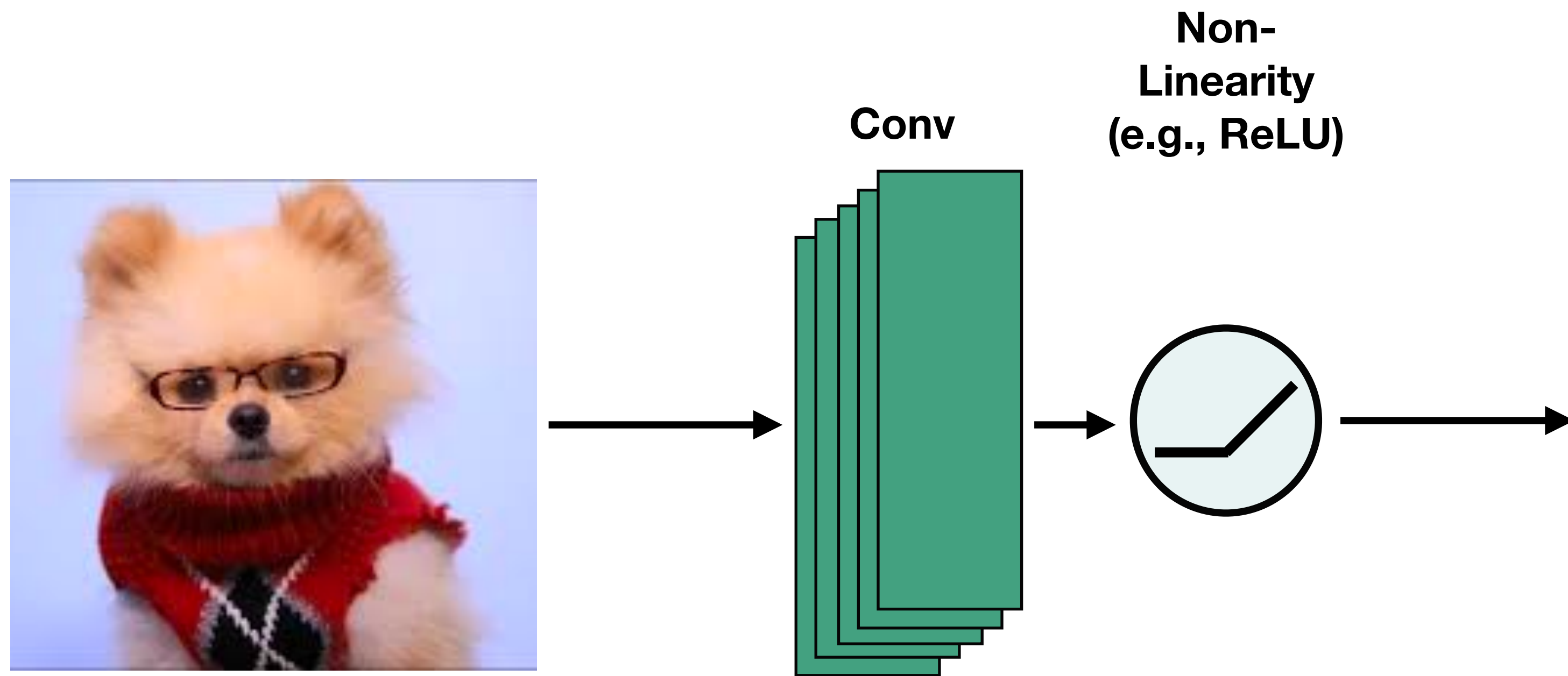
- A way to reduce the “depth” dimension of convolutional outputs
- Corresponds to applying an MLP to every pixel in the convolutional output
- Crucial to popular convnet architectures like Inception (GoogLeNet)

# Questions?

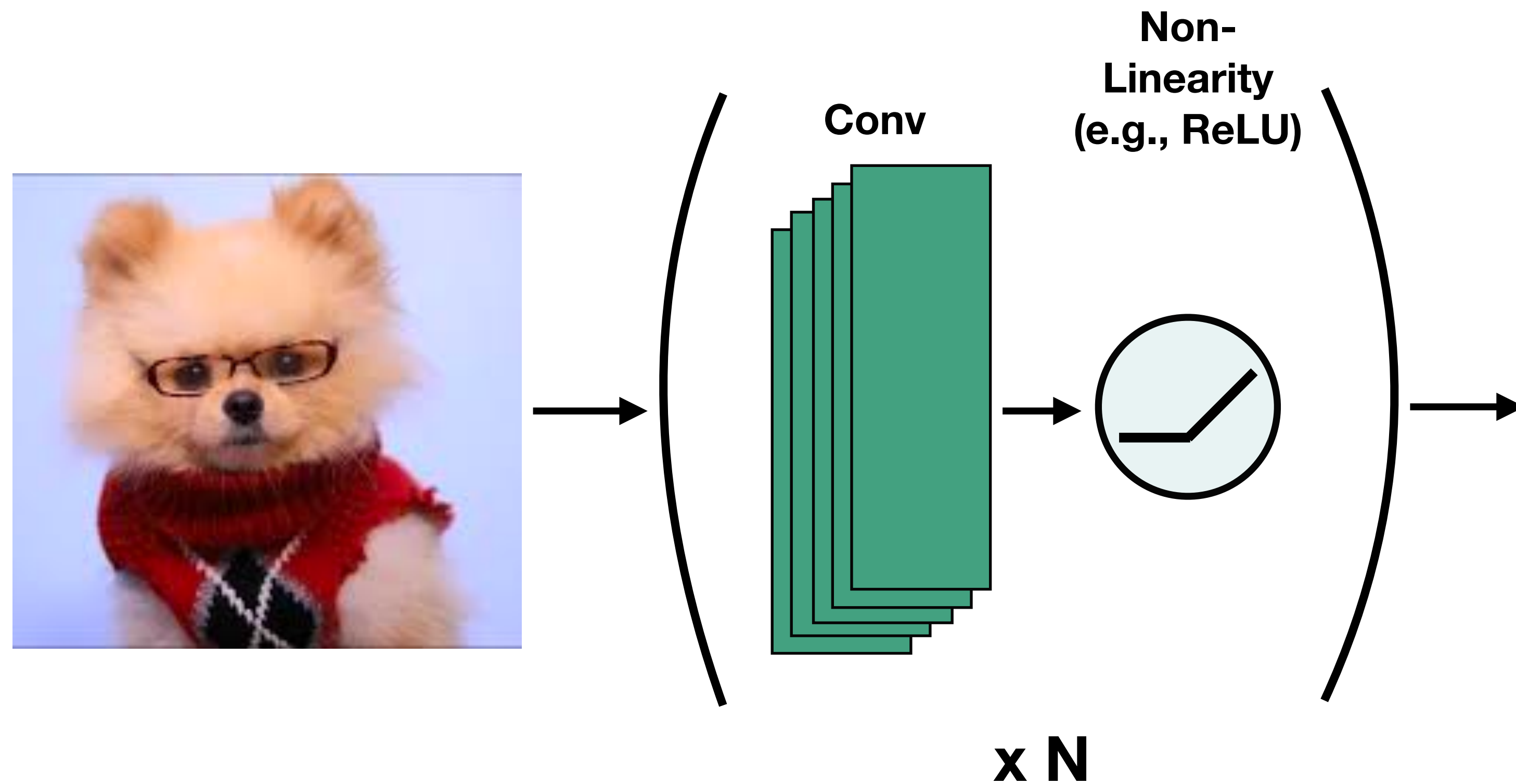
# Agenda

1. Review of the convolution operation
2. Other important operations for ConvNets
3. **Classic ConvNet architectures**

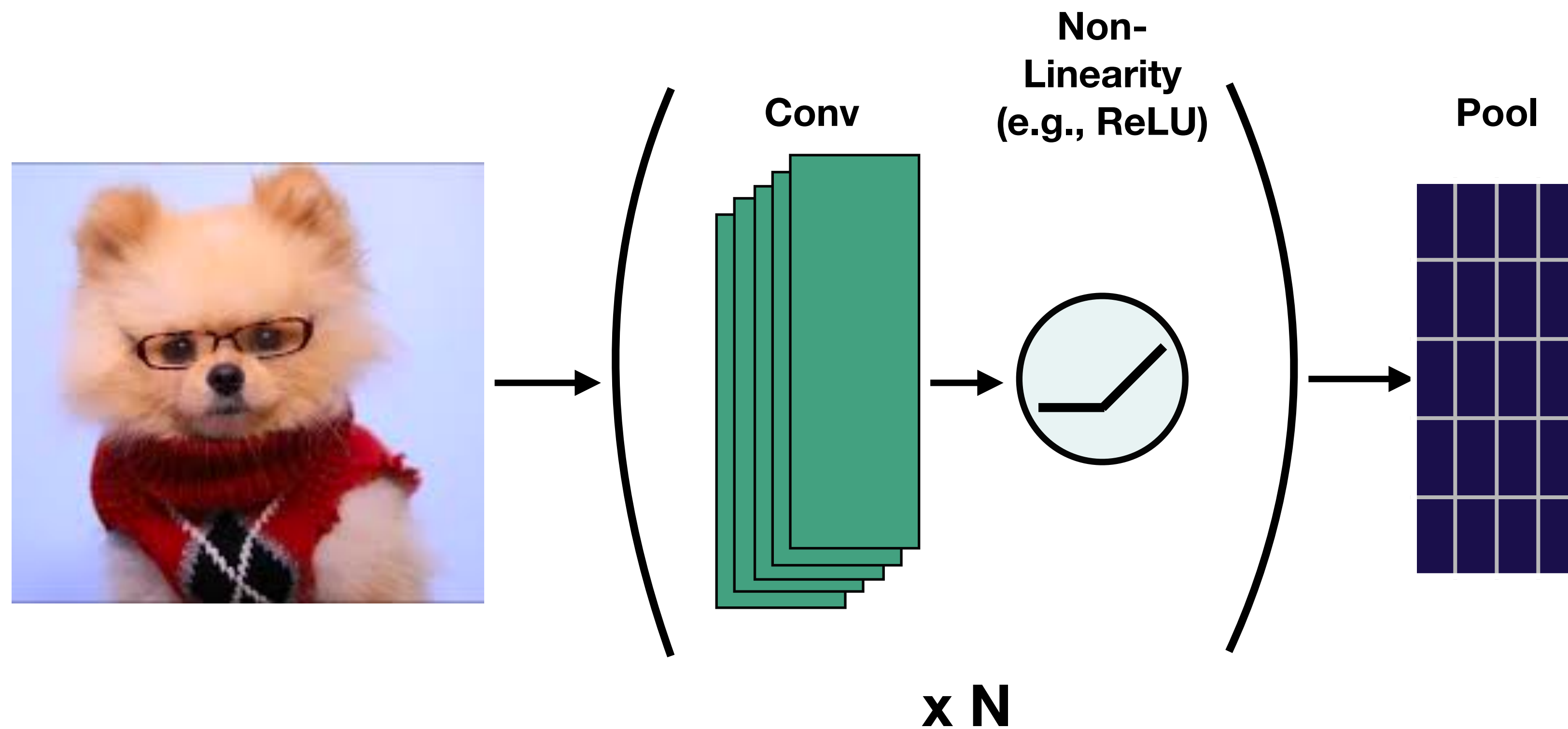
# Most standard: LeNet(-like) architectures



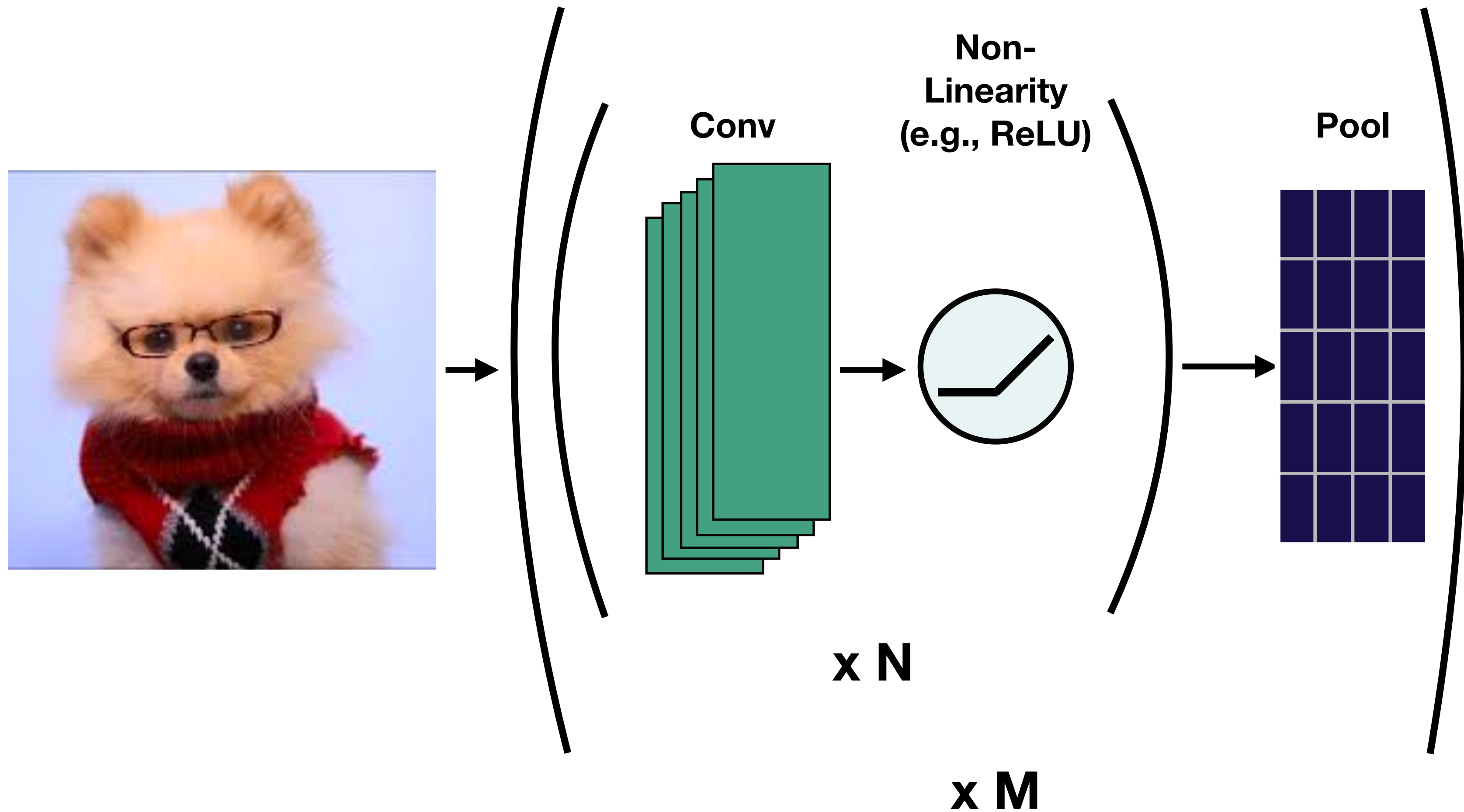
# Most standard: LeNet(-like) architectures



# Most standard: LeNet(-like) architectures

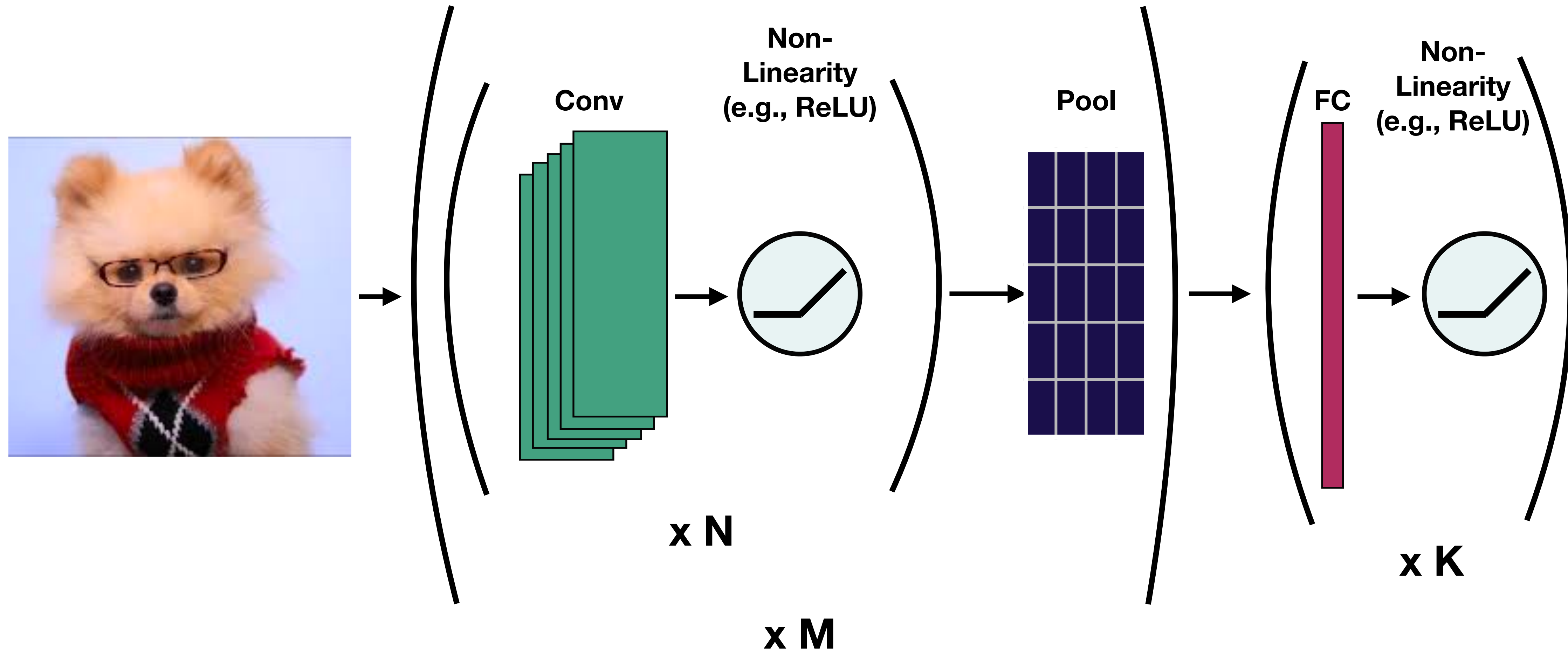


# Most standard: LeNet(-like) architectures



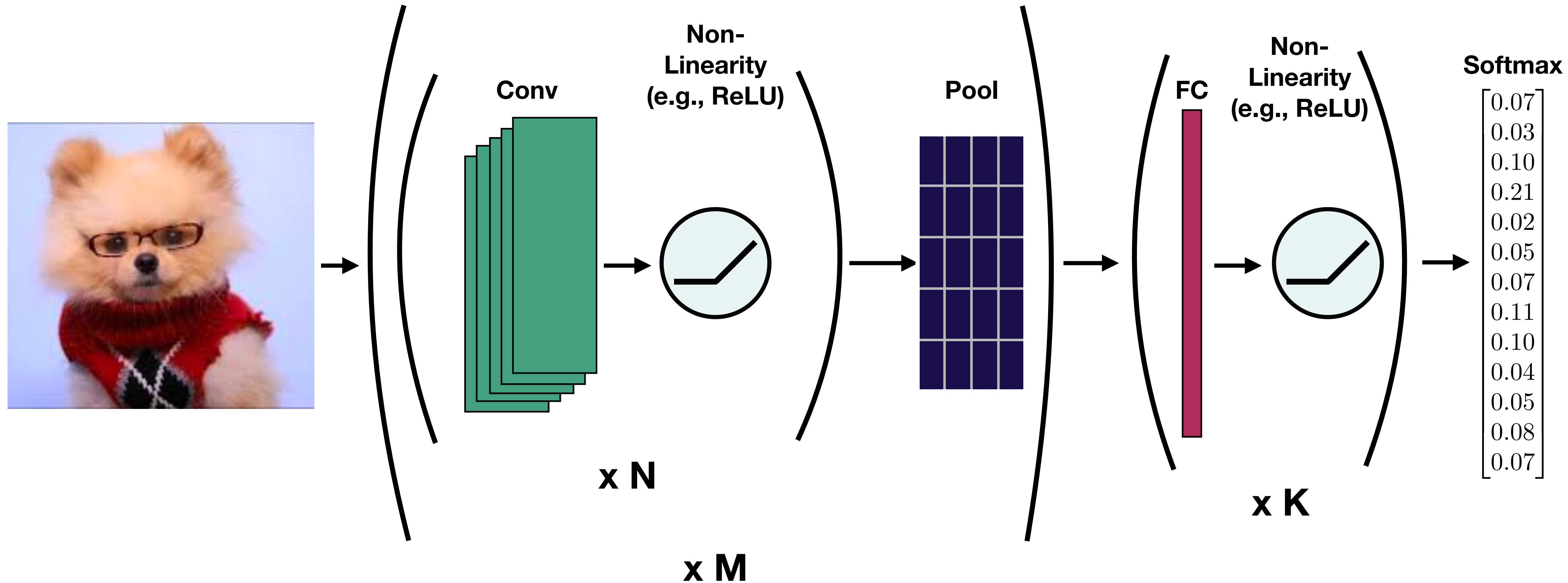


# Most standard: LeNet(-like) architectures

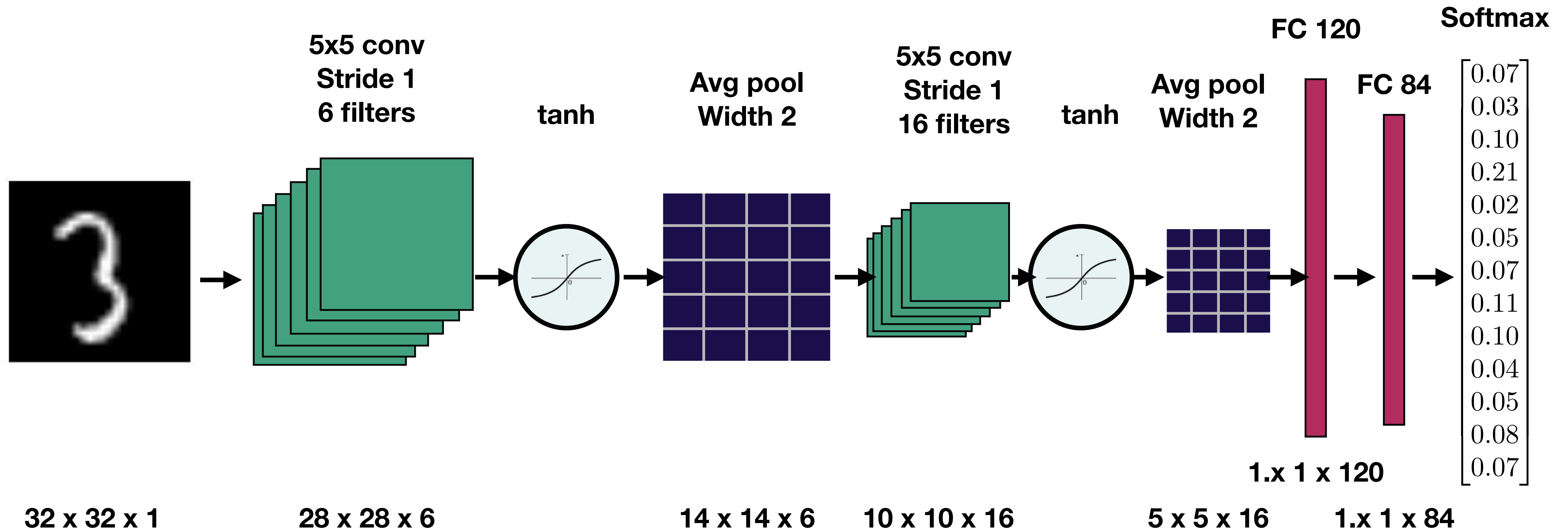




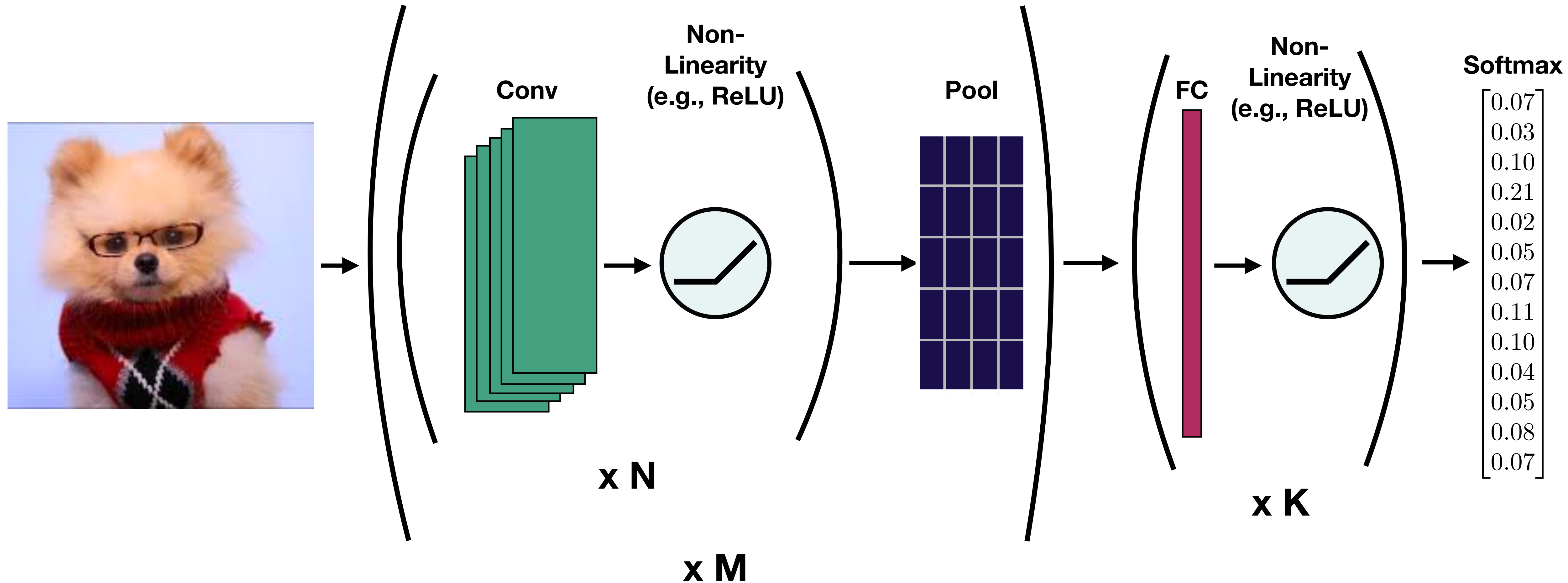
# Most standard: LeNet(-like) architectures



# Classic Convnet Architecture: LeNet



# Most standard: LeNet(-like) architectures



**More modern convnet architectures in the Vision Applications lecture!**

# Questions?

## FSDL (2021) · 课程资料包 @ShowMeAI



视频  
中英双语字幕



课件  
一键打包下载



笔记  
官方笔记翻译



代码  
作业项目解析



视频 · B 站 [ 扫码或点击链接 ]  
<https://www.bilibili.com/video/BV1iL411t7jE>



课件 & 代码 · 博客 [ 扫码或点击链接 ]  
<http://blog.showmeai.tech/berkeley-fsdl>

深度学习  
神经网络  
循环神经网络

可解释性  
计算机视觉  
数据管理

持续集成

迁移学习  
Transformer  
部署模型

卷积神经网络  
深度神经网络调试  
监控模型  
测试

Awesome AI Courses Notes Cheatsheets 是 [ShowMeAI](#) 资料库的分支系列，覆盖最具知名度的 **TOP50+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n
# Awesome AI Courses Notes Cheatsheets · 持续更新中			
知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094



### 微信公众号

资料下载方式 2：扫码点击**底部菜单栏**  
称为 **AI 内容创作者**？回复 [ 添砖加瓦 ]