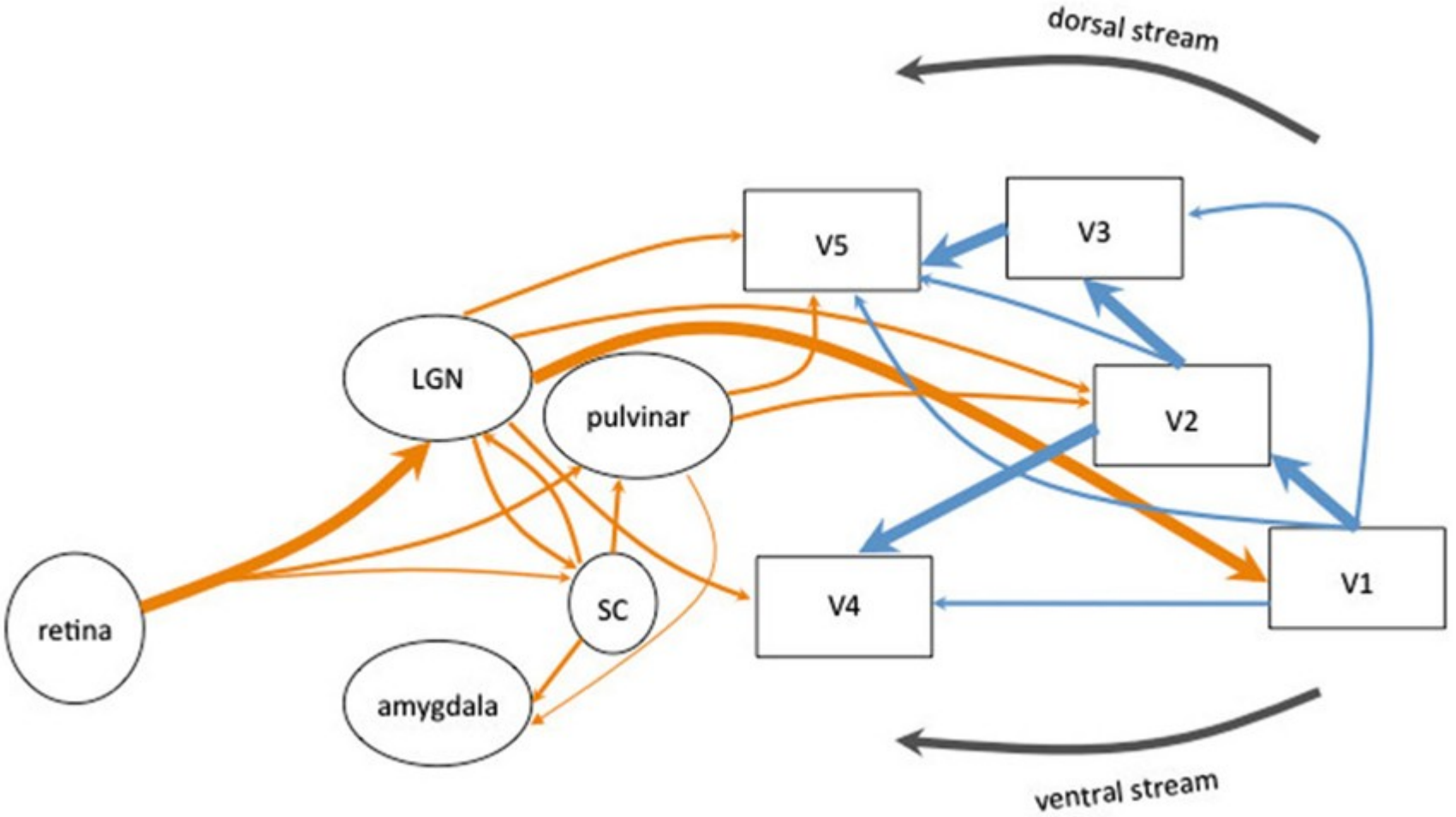
Convolutional Neural Networks

Nathan Sprague

Human Visual System



Urbanski, Marika, Olivier A. Coubard, and Clémence Bourlon. "Visualizing the blind brain: brain imaging of visual field defects from early recovery to rehabilitation techniques." Neurovision: Neural bases of binocular vision and coordination and their implications in visual training programs (2014).

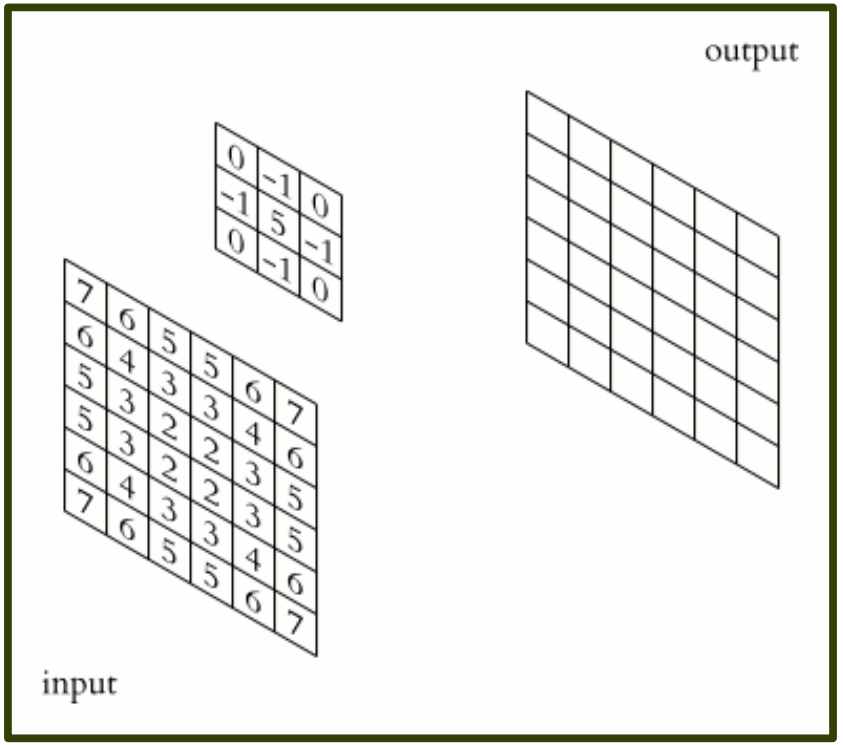
Convolutional Neural Networks

- Convolutional neural networks use the same trick of learning layers of localized features...
- CNN's were being used by Yann Lecun at Bell Labs around 1990

Convolutions

Grayscale Image

1 convolutional filter

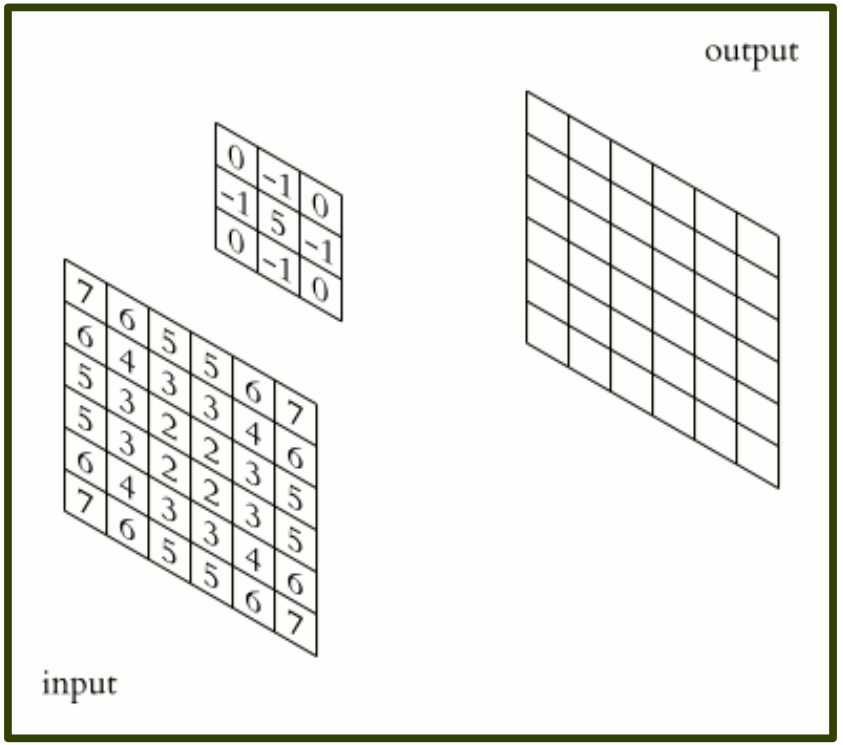


http://upload.wikimedia.org/wikipedia/commons/4/4f/3D_Convolution_Animation.gif
By Michael Plotke [CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0)

Convolutions

Grayscale Image

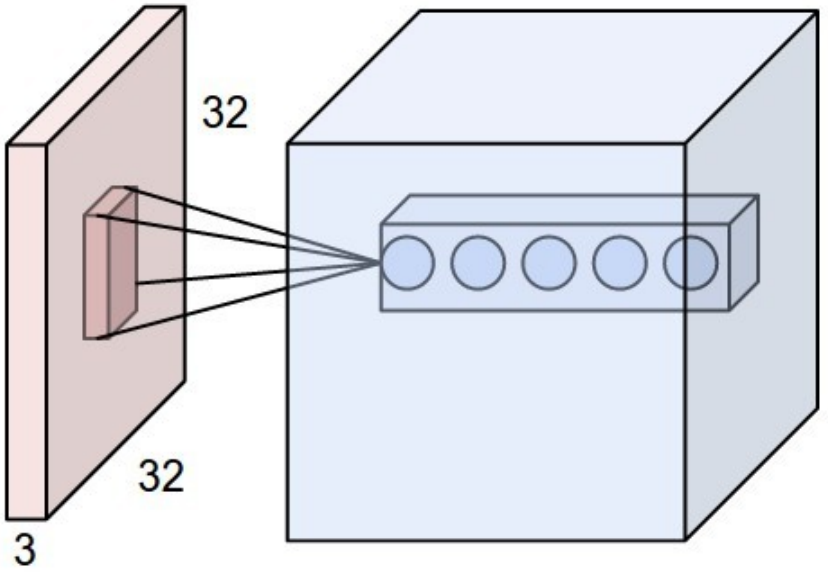
1 convolutional filter



http://upload.wikimedia.org/wikipedia/commons/4/4f/3D_Convolution_Animation.gif
By Michael Plotke [CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0)

Color Image

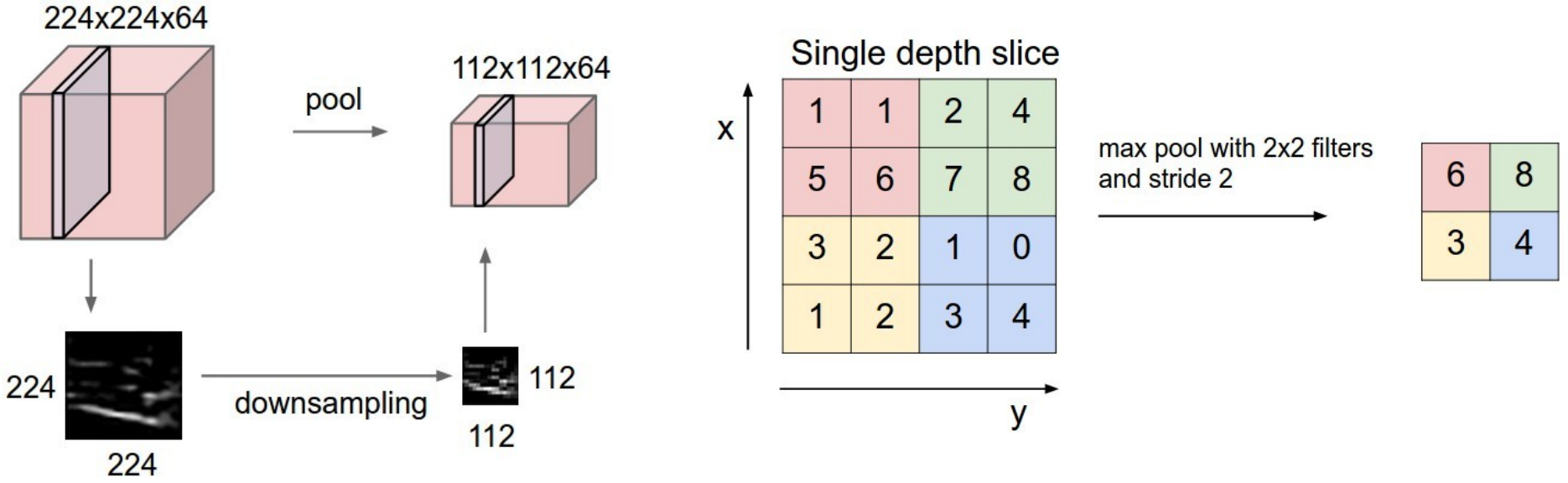
5 convolutional filters



http://cs231n.github.io/convolutional-networks/

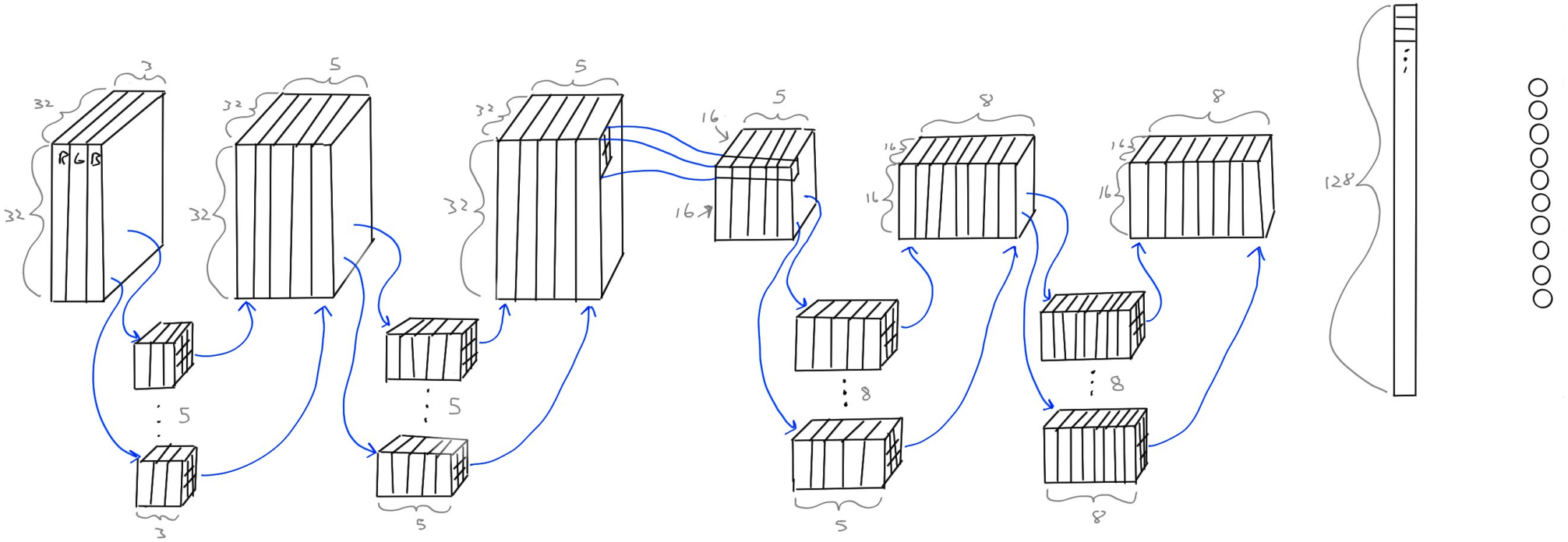
Pooling Layers

- Pooling layers down-sample the filter outputs to
  - Reduce dimensionality and computational requirements
  - Increase the spatial extent of subsequent filters



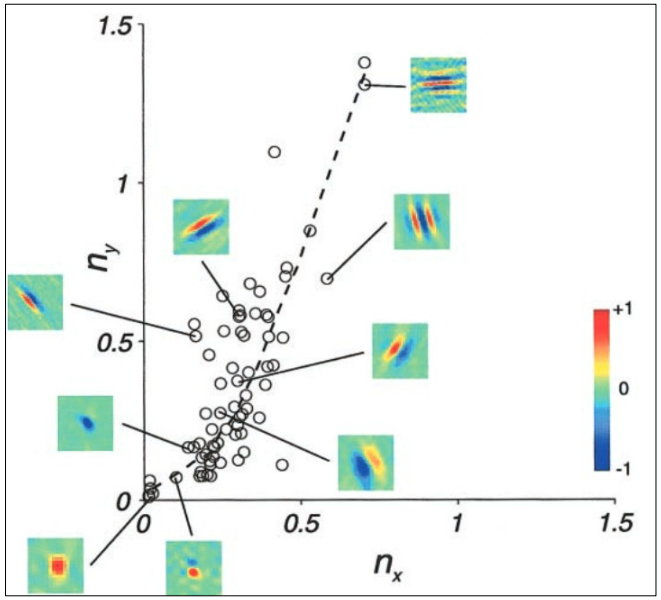
Complete Network

- A "traditional" CNN is composed of convolutional layers, each followed by non-linearities, followed by pooling layers, with one or more dense (non-convolutional) layer at the end:



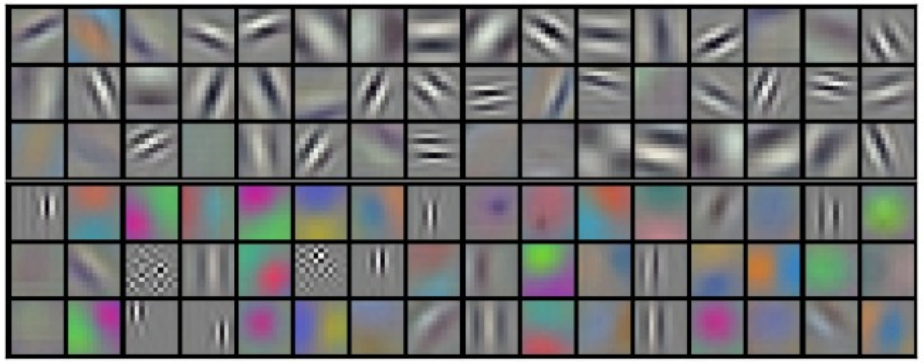
Learned Features

Macaque V1 Receptive Fields



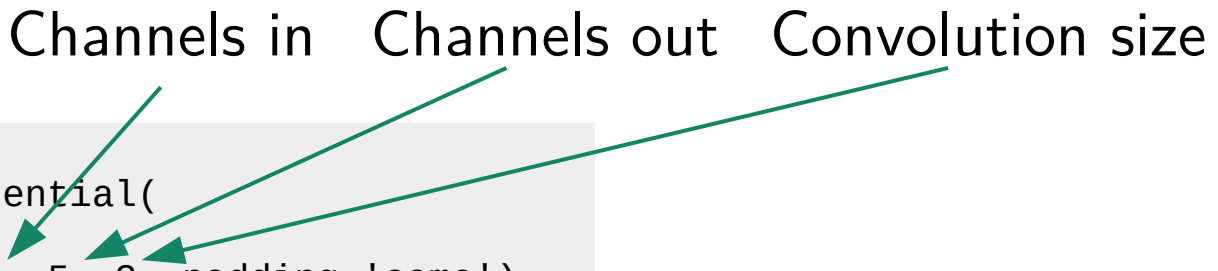
Spatial Structure and Symmetry of Simple-Cell Receptive Fields in Macaque Primary Visual Cortex
Dario L. Ringach Journal of Neurophysiology 2002 88:1, 455-463

Alexnet First Convolutional Layer



Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems 25 (2012).

Complete Network (in PyTorch)



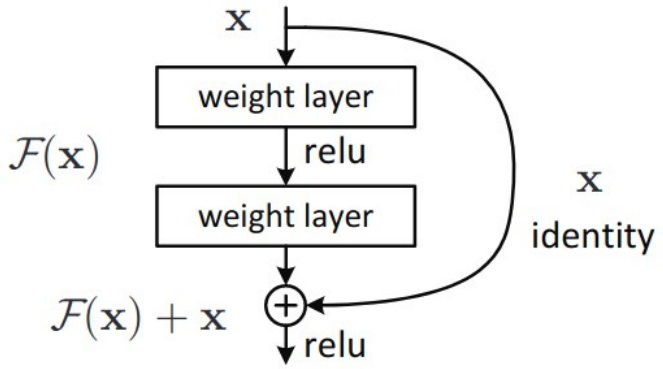

```
model = nn.Sequential(

    nn.Conv2d(3, 5, 3, padding='same'),
    nn.ReLU(),

    nn.Conv2d(5, 5, 3, padding='same'),
    nn.ReLU(),

    nn.MaxPool2d(2, 2),

    nn.Conv2d(5, 8, 3, padding='same'),
    nn.ReLU(),

    nn.Conv2d(8, 8, 3, padding='same'),
    nn.ReLU(),

    nn.Flatten(),
    nn.Linear(16 * 16 * 8, 128),
    nn.ReLU(),

    nn.Linear(128, 10)
)
```

Complete Network (alternate version)

```
class ConvNet(nn.Module):

    def __init__(self):
        super().__init__()
        self.relu = nn.ReLU()
        self.conv1 = nn.Conv2d(3, 5, 3, padding='same')
        self.conv2 = nn.Conv2d(5, 5, 3, padding='same')
        self.pool = nn.MaxPool2d(2, 2)
        self.conv3 = nn.Conv2d(5, 8, 3, padding='same')
        self.conv4 = nn.Conv2d(8, 8, 3, padding='same')
        self.flatten = nn.Flatten()
        self.fc = nn.Linear(16 * 16 * 8, 128)
        self.out = nn.Linear(128, 10)

    def forward(self, x):
        x = self.relu(self.conv1(x))
        x = self.relu(self.conv2(x))
        x = self.pool(x)
        x = self.relu(self.conv3(x))
        x = self.relu(self.conv4(x))
        x = self.flatten(x)
        x = self.relu(self.fc(x))
        x = self.out(x)
        return x

model = ConvNet()
```

Residual Networks

- How deep can we make these networks? Simply stacking more convolutional layers eventually degrades performance.
- One solution is to introduce “skip connections”:



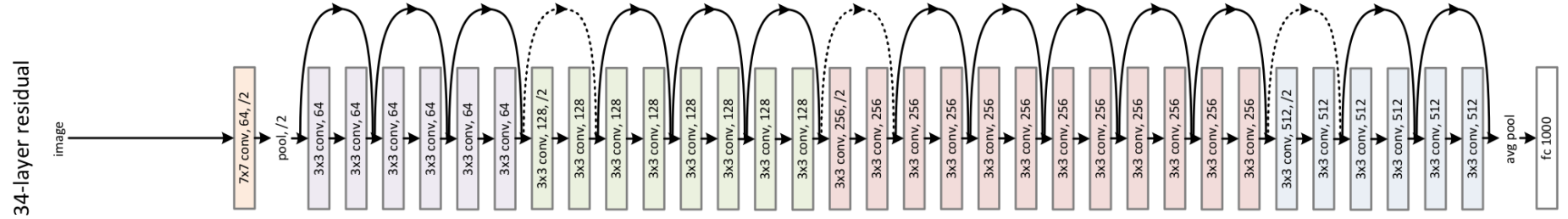
- “Residual learning”

He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

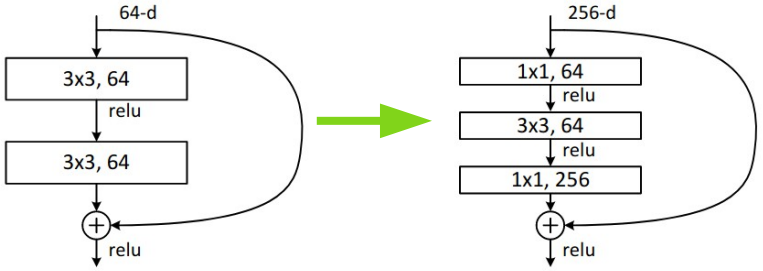
> 200,000 citations!

Residual Networks

- ResNet-34:



- Get ResNet-50 by introducing "bottleneck" blocks:



- The 1x1 convolutions can be used to increase or decrease the number of channels