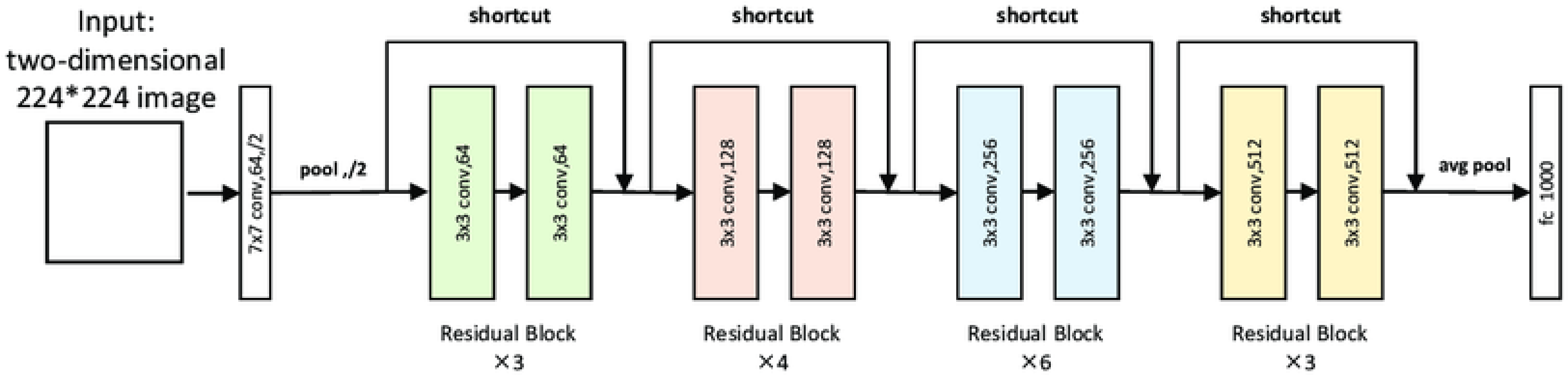
EinOps Whoopsydoopsy

```python
# Matrix multiplication (2D × 2D)
c = einsum('ij,jk->ik', a, b)

# Batch matrix multiplication (3D × 3D)
c = einsum('bij,bjk->bik', a, b)

# Dot product
c = einsum('i,i->', a, b)

# Outer product
c = einsum('i,j->ij', a, b)
```

$$M_{ij} = \cancel{\sum_k} A_{ik} B_{kj} = A_{ik} B_{kj}$$

Implementing your own Neural Network

Explanation of building blocks



nn.Module

- The base class for all neural network components
- Allows you to organize layers and their parameters modularly
- Automatically tracks all parameters for optimization
- Must implement forward() method defining computation

nn.Parameter

- Special tensor that gets automatically tracked
- Used for learnable weights and biases
- Will be updated during backpropagation
- Example:

```
self.weight = nn.Parameter(torch.randn(3, 4))
```

Buffers:

-tracked by the model, part of statedict

-not updated by the optimizer

-Example:

```
self.register_buffer('running_mean', torch.zeros(hidden_dim))
```

```
self.running_mean = 0.9 * self.running_mean + 0.1 * batch_mean
```

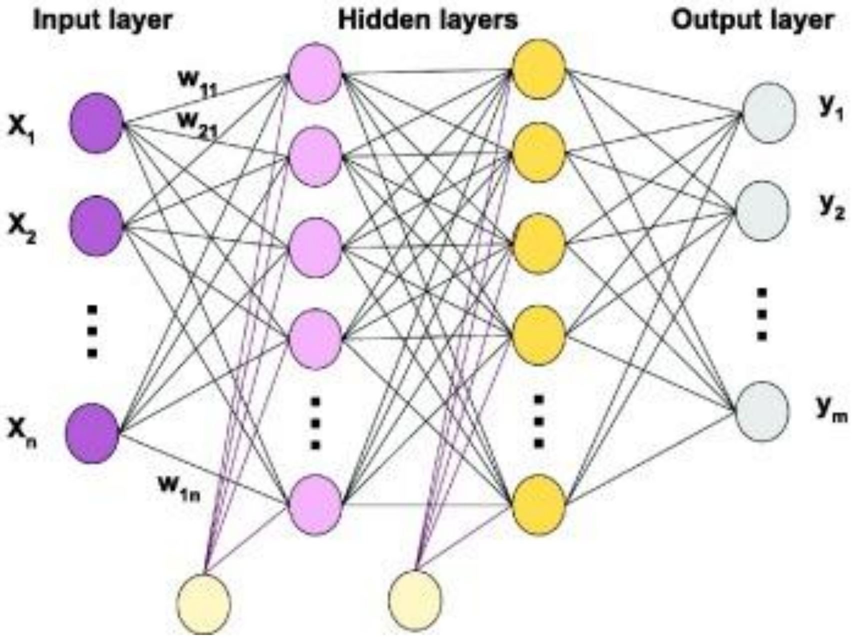
nn.Sequential

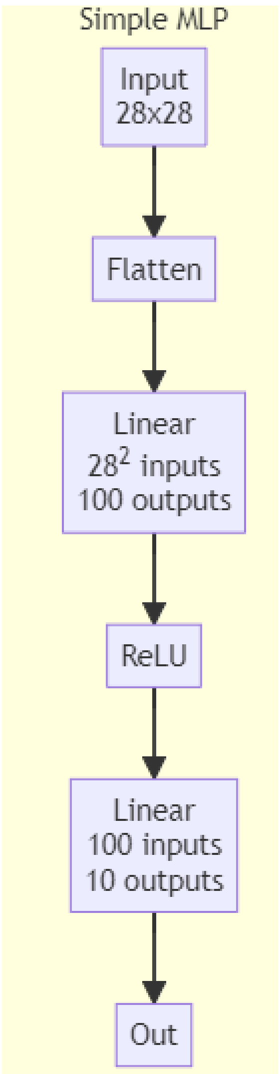
- Container to stack modules in sequence
- Automatically chains forward() calls
- Useful for linear architectures
- Example:

```
nn.Sequential(
    nn.Linear(10, 20),
    nn.ReLU(),
    nn.Linear(20, 30)
)
```

Multyilayer Perceptron

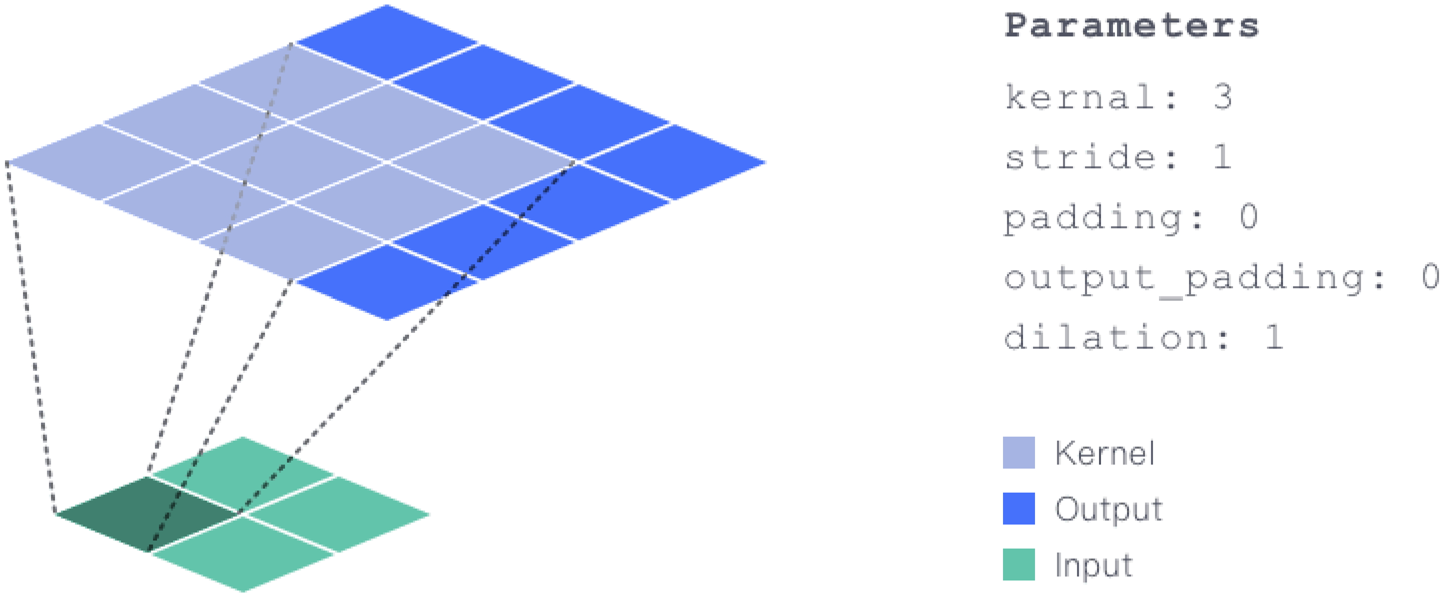
- The classic simplest neural network
- A stack of fully connected layers with activation functions
- No sense of space in the neural network structure
- Works with any kind of data
- Also, still part of Transformer architecture





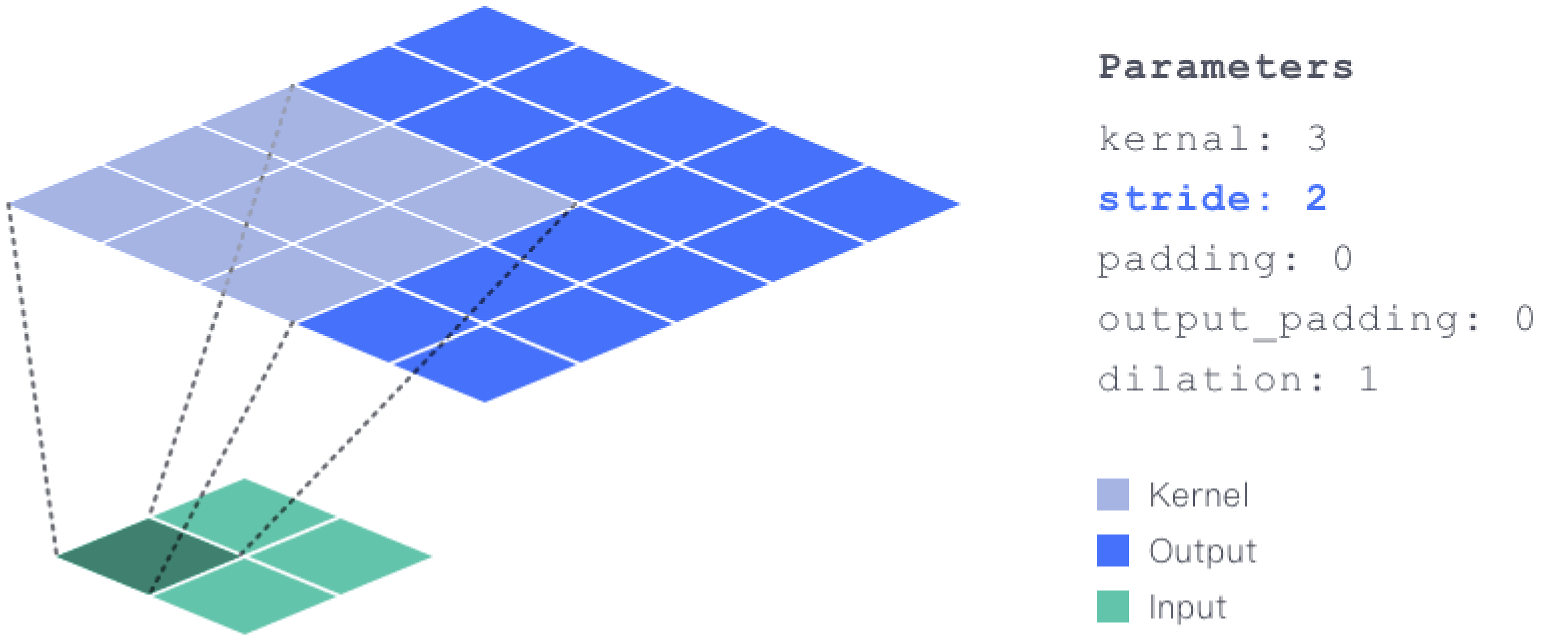
Convolution

- You can make use of the “local” structure of the image
- An “edge detector” (for example)acts the same on parts of the image



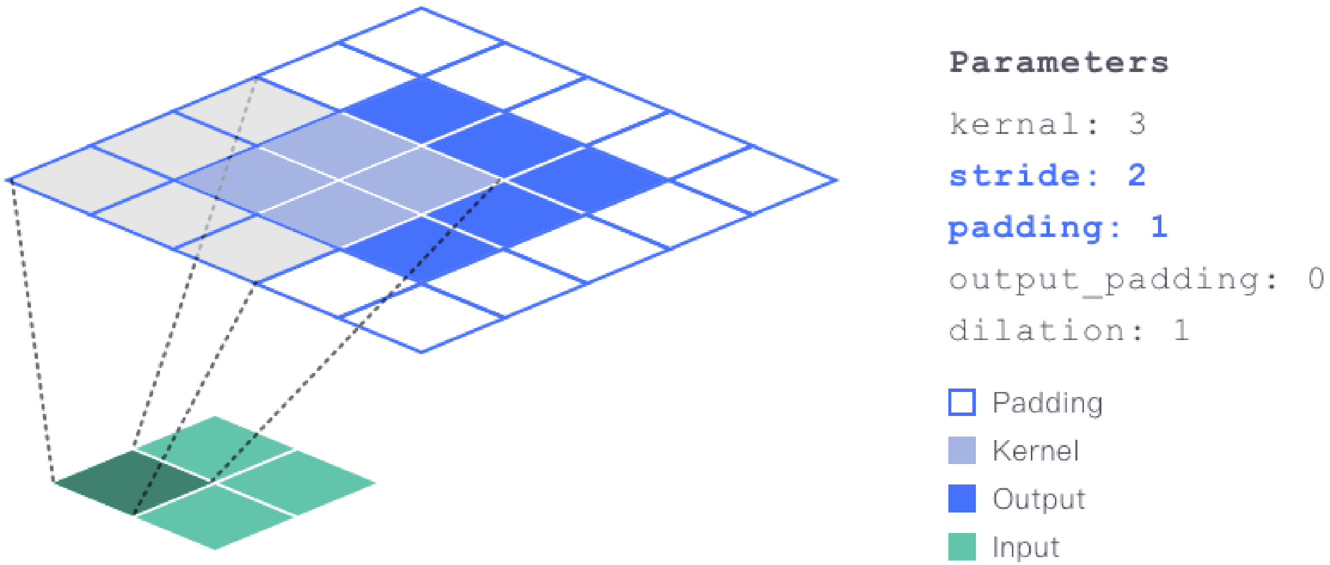
Convolution - stride

- You can give the image, as it passes through the network, lower resolution, by skipping steps
- Here, no pixel gets completely lost



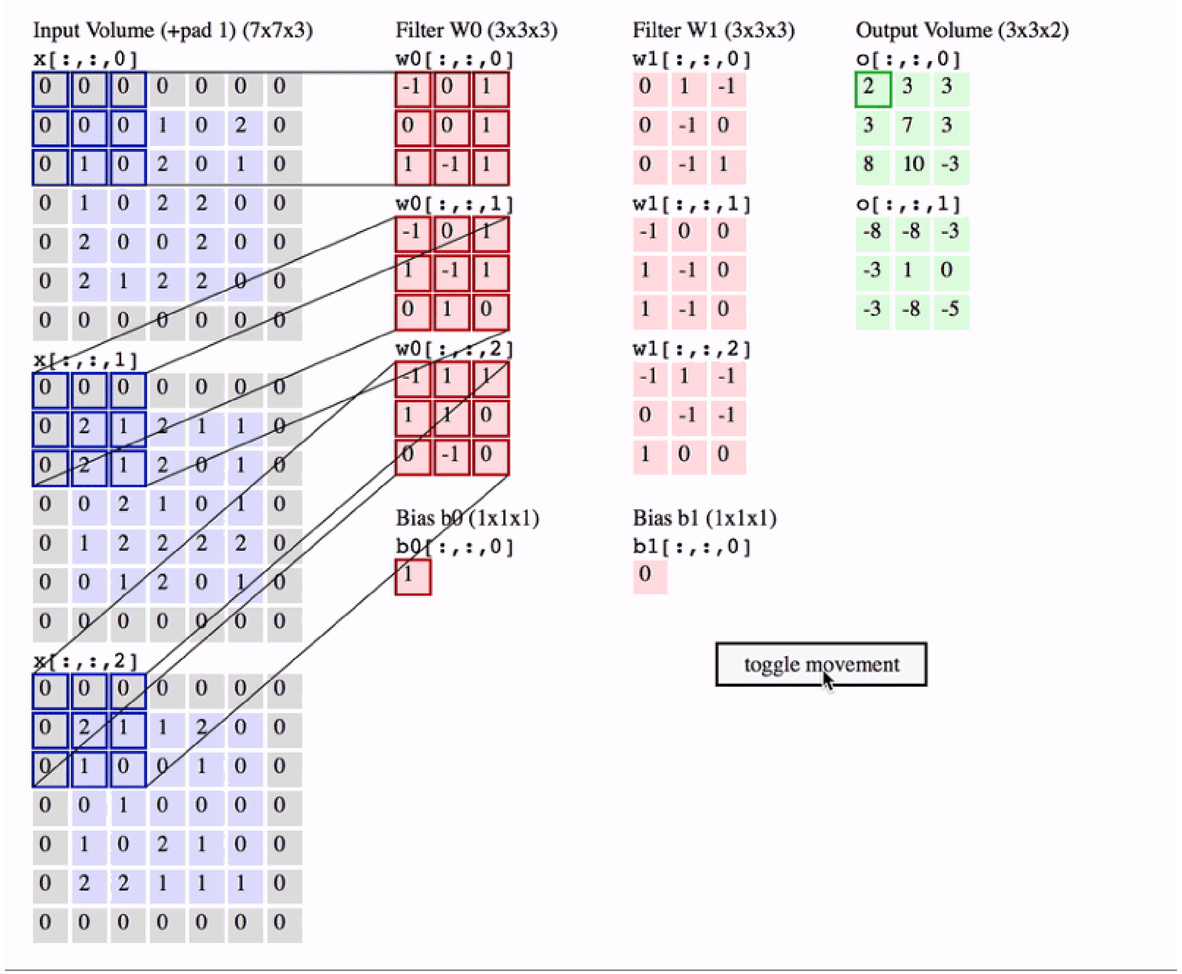
Convolution - padding

- Pixels on the border of the image get viewed only once, image gets smaller
- Image gets smaller without stride
- Add “empty” pixels outside the image to cancel this effect



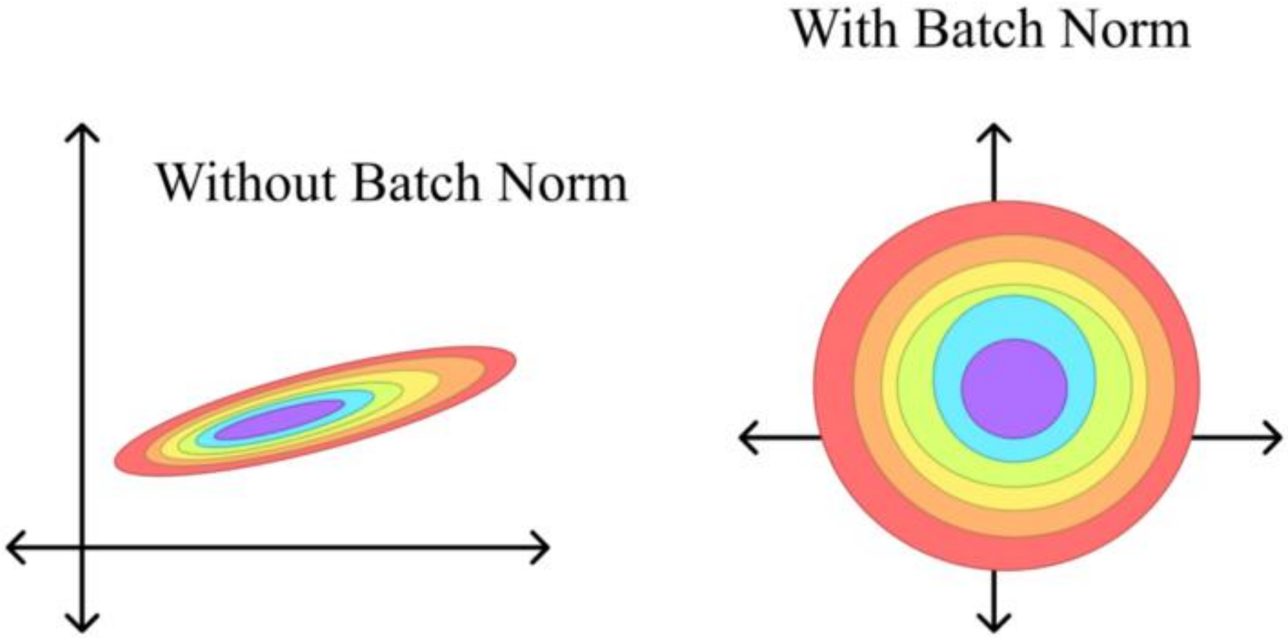
Convolution - channels

- Typically, you have more then one convolution kernel active at once
- This leads to an additional output dimension per picture: channel
- Example: Colour or different features like different edges/objects



Batch Norm

- The value of the activation is less important then its relative size
- Example: brightness in an image / across images
- Normalize distribution



Batch Norm– how this gets calculated

Learnable parameters

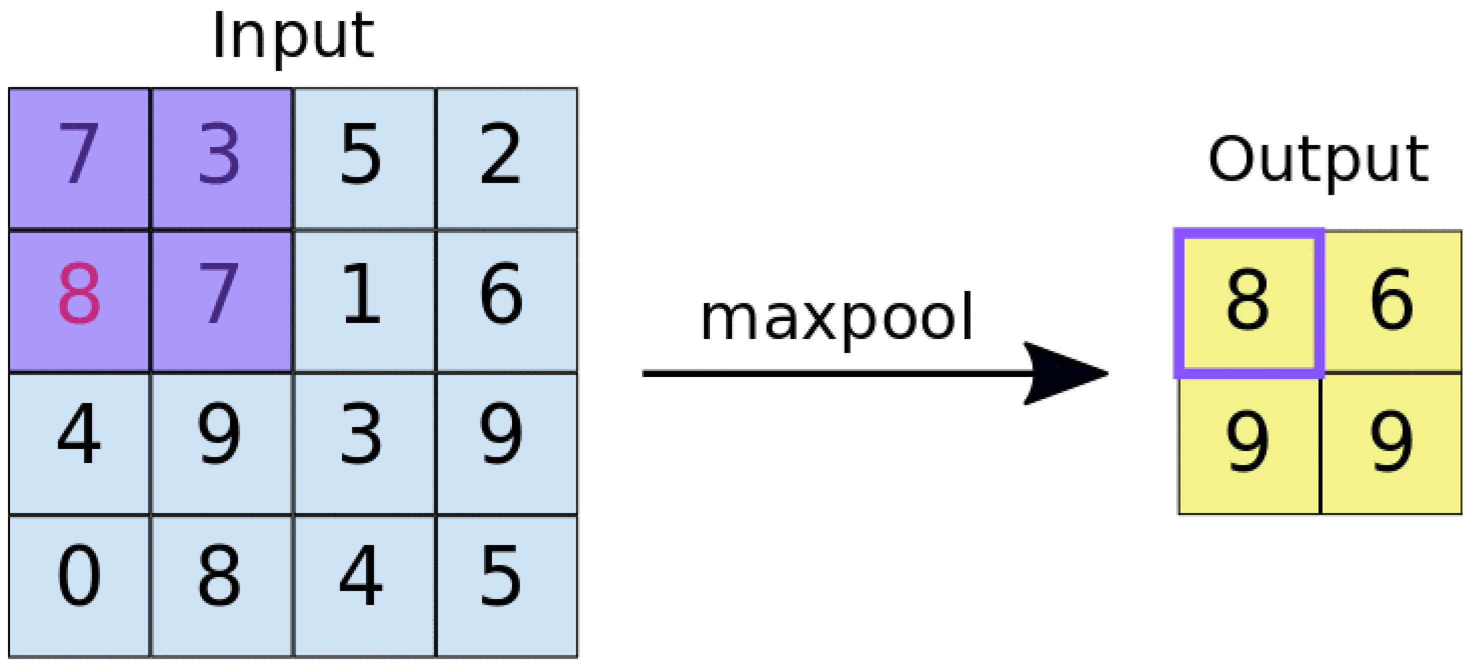
$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} * \gamma + \beta$$

In Training: Updated as a rolling average
In Deployment: Frozen

Activations: For taking activation statistics, all activations from the same channel are taken as draws from the same distribution (independent of location in picture)

MaxPooling (2d)

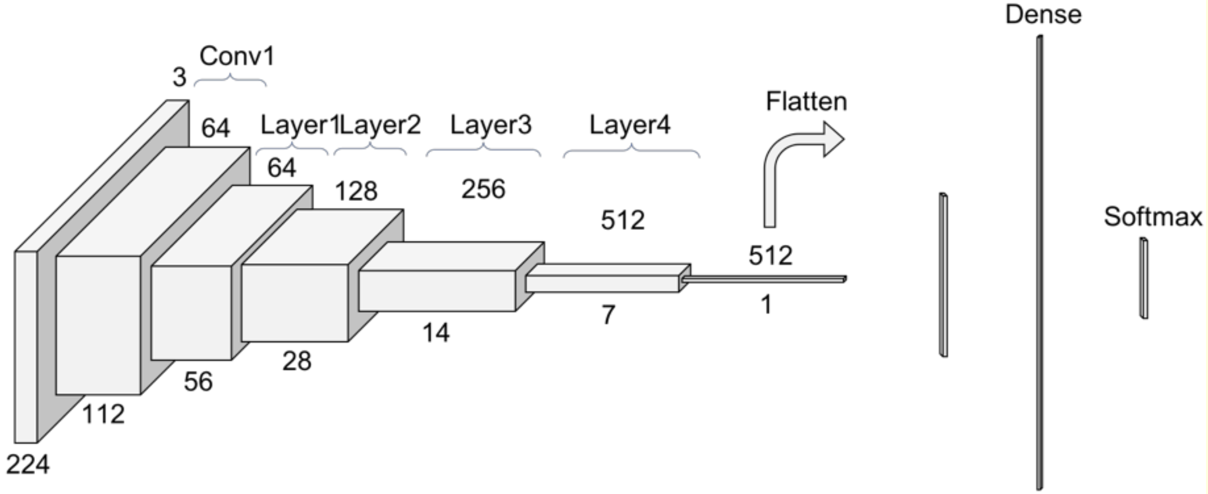
- Reduce the size of the resulting Image
- Taking the maximum of each kernel window
- Example: Object detection in a part of the image



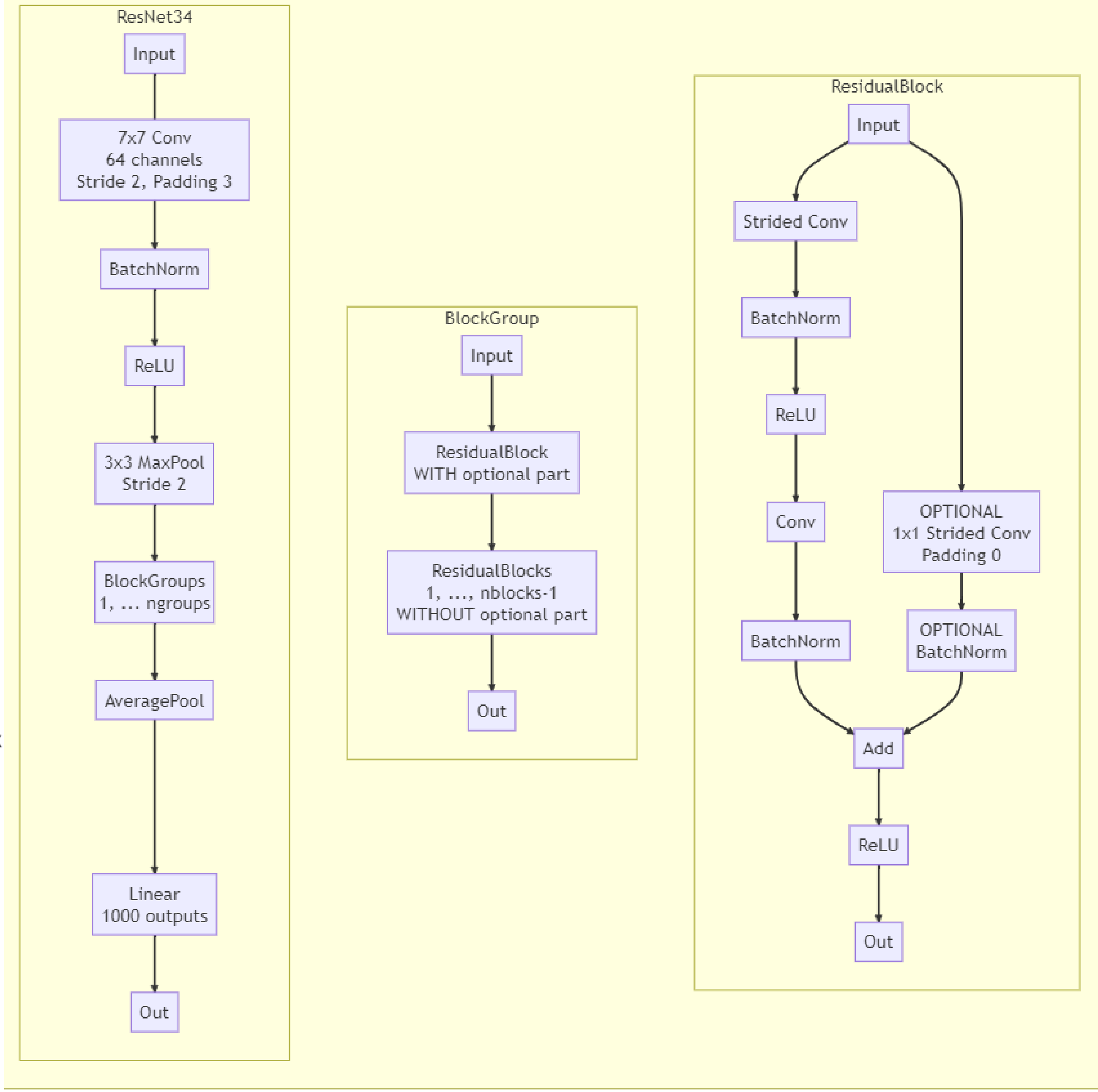
AveragePool

- Reduce resulting image to a single number
- Example: classifying an image with a single lable
- Average over space dimensions

ResNet 34



Conv1
3
64
Layer1
Layer2
Layer3
Layer4
64
128
256
512
Flatten
512
1
7
14
28
56
112
224
Dense
Softmax



ResNet34
Input
7x7 Conv
64 channels
Stride 2, Padding 3
BatchNorm
ReLU
3x3 MaxPool
Stride 2
BlockGroups
1, ... ngroups
AveragePool
Linear
1000 outputs
Out
BlockGroup
Input
ResidualBlock
WITH optional part
ResidualBlocks
1, ..., nblocks-1
WITHOUT optional part
Out
ResidualBlock
Input
Strided Conv
BatchNorm
ReLU
Conv
BatchNorm
OPTIONAL
1x1 Strided Conv
Padding 0
OPTIONAL
BatchNorm
Add
ReLU
Out