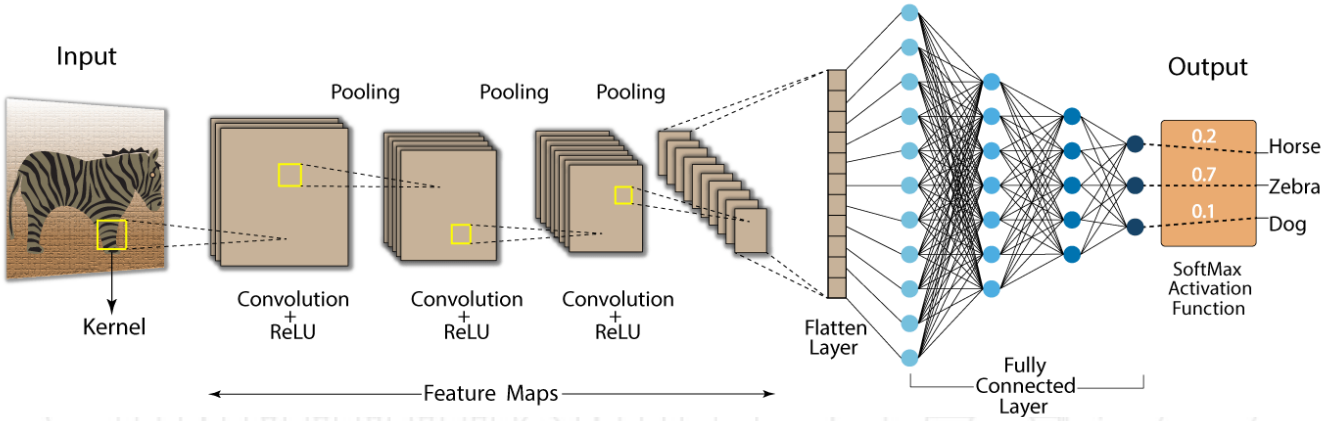
Module 5

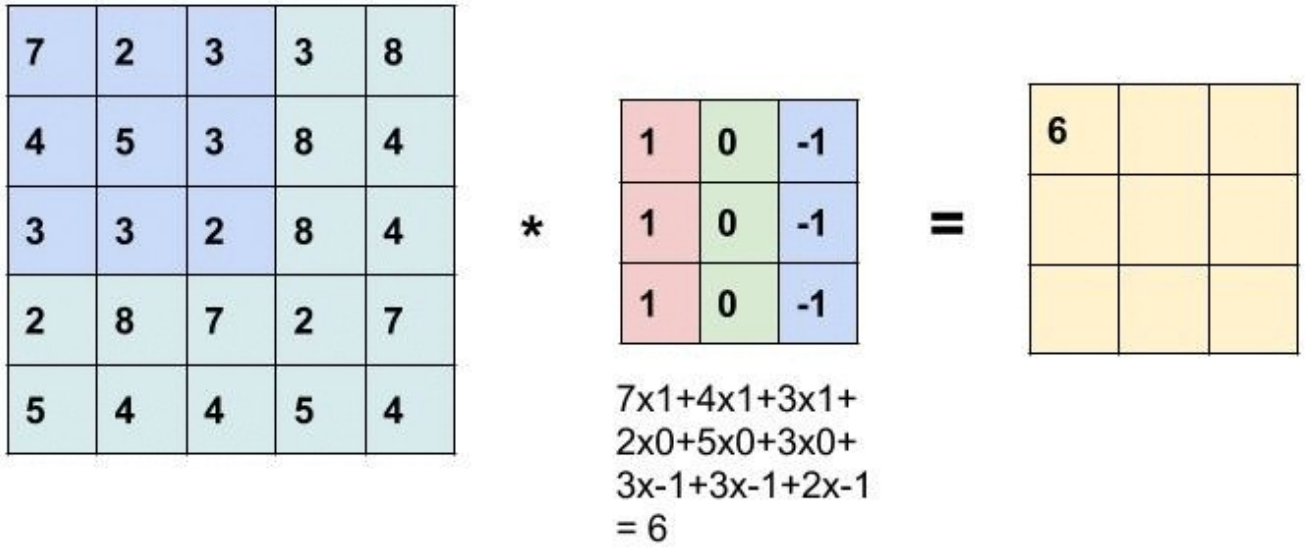
Convolutional Neural Networks

DL: CNNs Pre-Trained Models



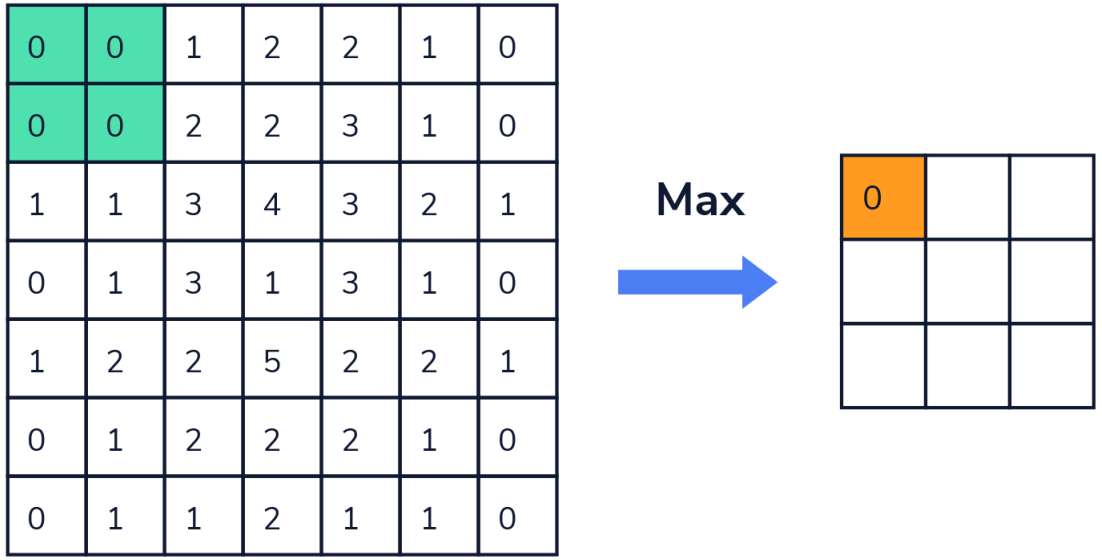
Review: Convolutions

- Below is a (3×3) kernel (convolution) applied to (5×5) image
- Stride is $(1,1)$



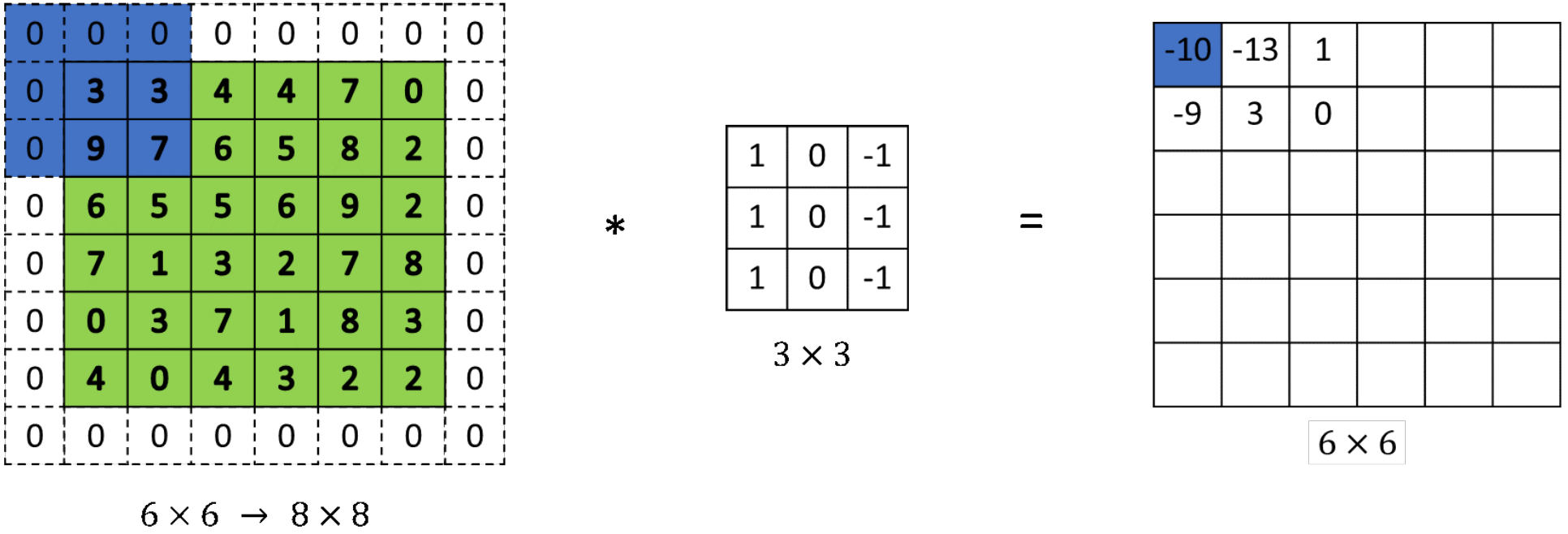
Review: Pooling

- A pooling filter is used to downsample the images
- min, max and average pooling are the most used.

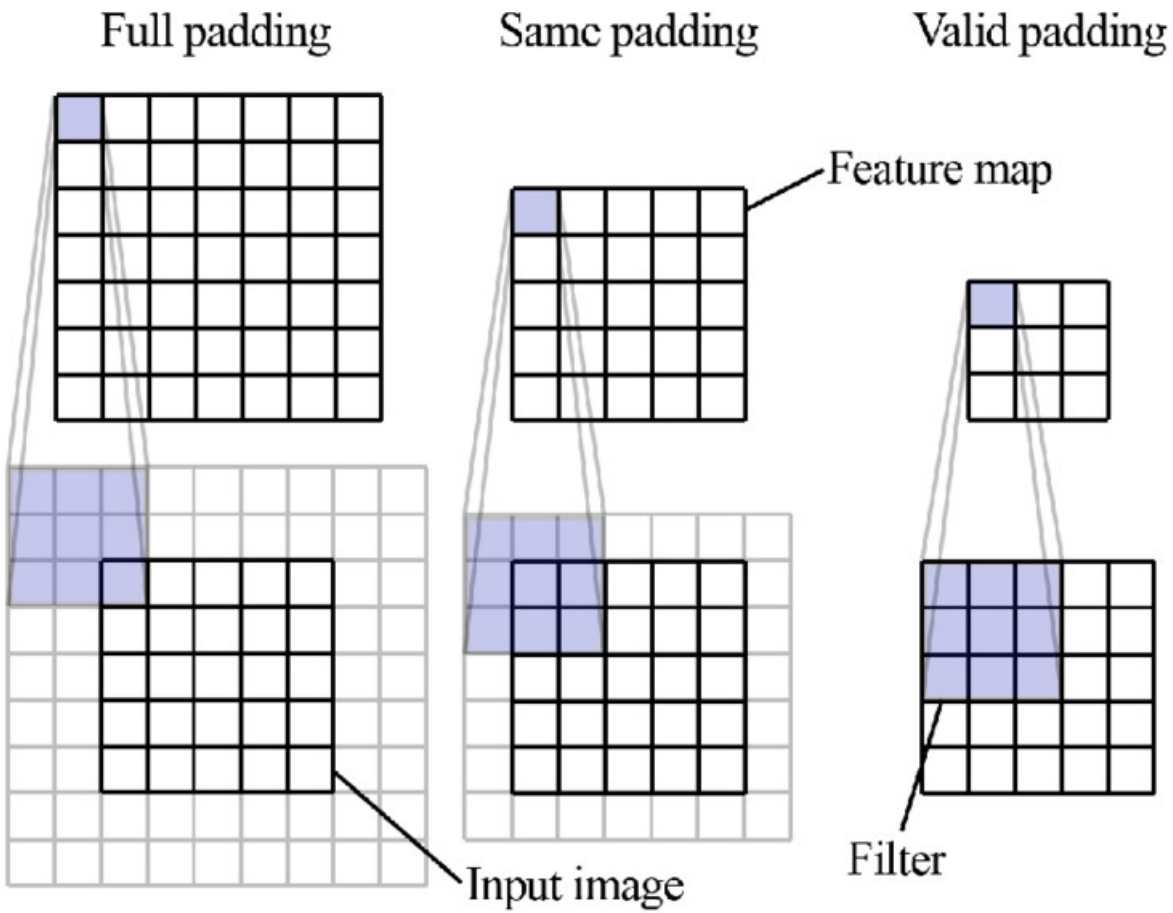


Padding

- Used to preserve the spatial dimensions of the input image
- Example: adding an extra layer of padding, e.g., padding = 1



Types of Padding



Output Size with Padding and Stride

```scss
output_size = (input_size - filter_size + 2*padding) / stride + 1
            = (32 - 5 + 2*2) / 1 + 1
            = 32
```

Copy code

0	0	0	0	0	0	0	0
0	3	3	4	4	7	0	0
0	9	7	6	5	8	2	0
0	6	5	5	6	9	2	0
0	7	1	3	2	7	8	0
0	0	3	7	1	8	3	0
0	4	0	4	3	2	2	0
0	0	0	0	0	0	0	0

$6 \times 6 \rightarrow 8 \times 8$

*

1	0	-1
1	0	-1
1	0	-1

3×3

=

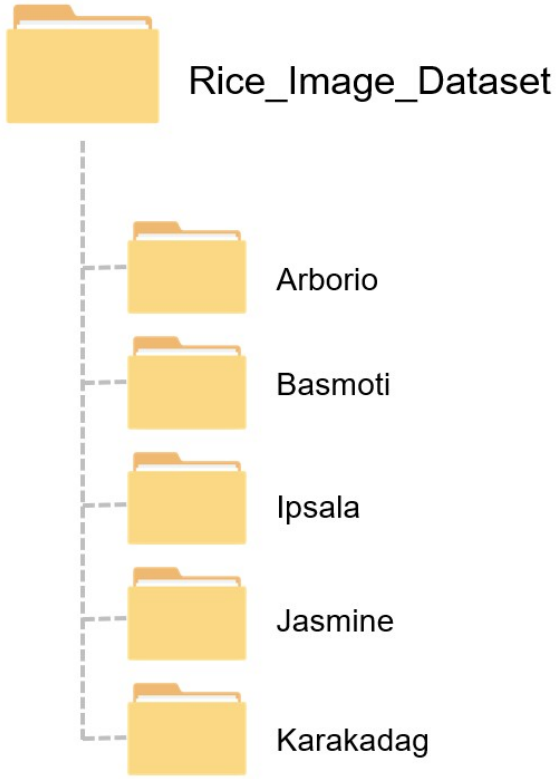
-10	-13	1			
-9	3	0			

6×6

Folder Structure for CNN Training

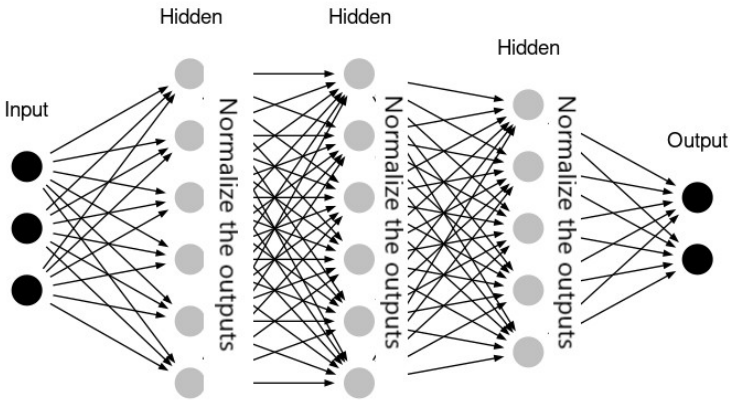
- A typical folder structure for a CNN-based image classification project:

```
cnn_project/
├── data/
│   ├── train/
│   │   ├── class_1/
│   │   └── class_2/
│   └── val/
│       ├── class_1/
│       └── class_2/
├── models/
│   └── saved_model.h5
├── notebooks/
│   └── exploratory_analysis.ipynb
```



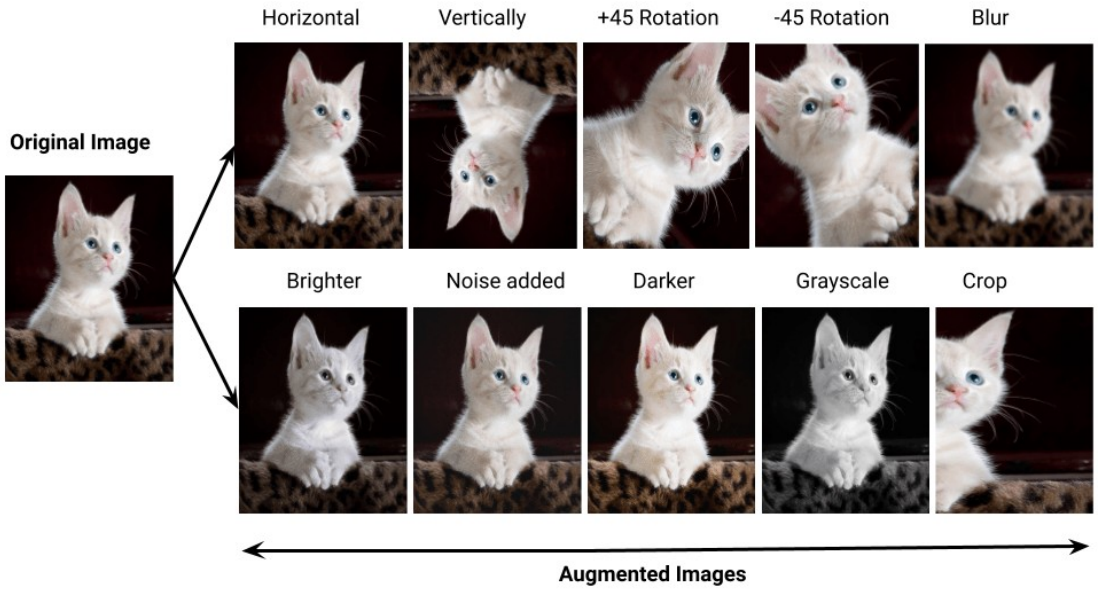
Batch Normalization

- Batch normalization is a technique used to improve the training of deep neural networks.
  - Normalizes the input of each layer to have zero mean and unit variance.
  - Helps reduce internal covariate shift.
  - Speeds up training and allows for higher learning rates.
  - Acts as a form of regularization, sometimes reducing the need for dropout.
  - Typically applied after linear transformation (e.g., convolution) and before the activation functio



Data Augmentation

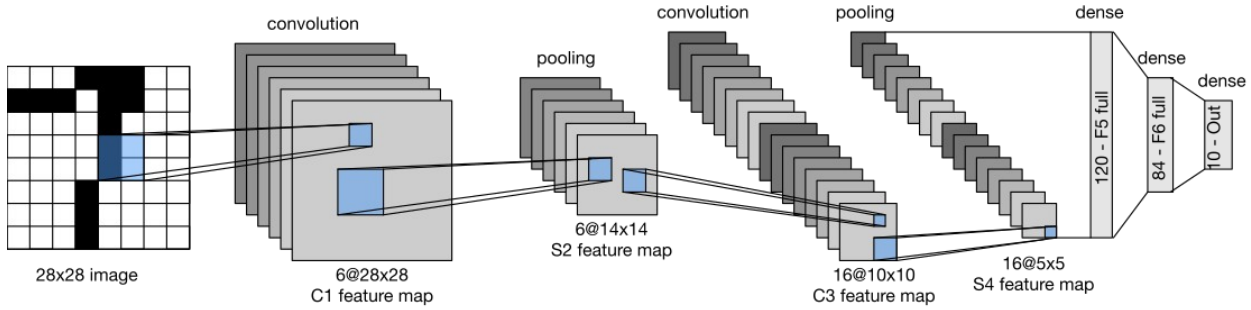
- Data augmentation is a technique used to increase the size and diversity of training datasets.
  - Image rotation, flipping (horizontal/vertical)
  - Cropping and resizing
  - Brightness and contrast adjustments
  - Adding noise
  - Translation and scaling
  - Cutout and mixup methods



Python

Examples of CNN Architectures

- LeNet-5 is a foundational CNN architecture designed for handwritten digit recognition (e.g., MNIST).



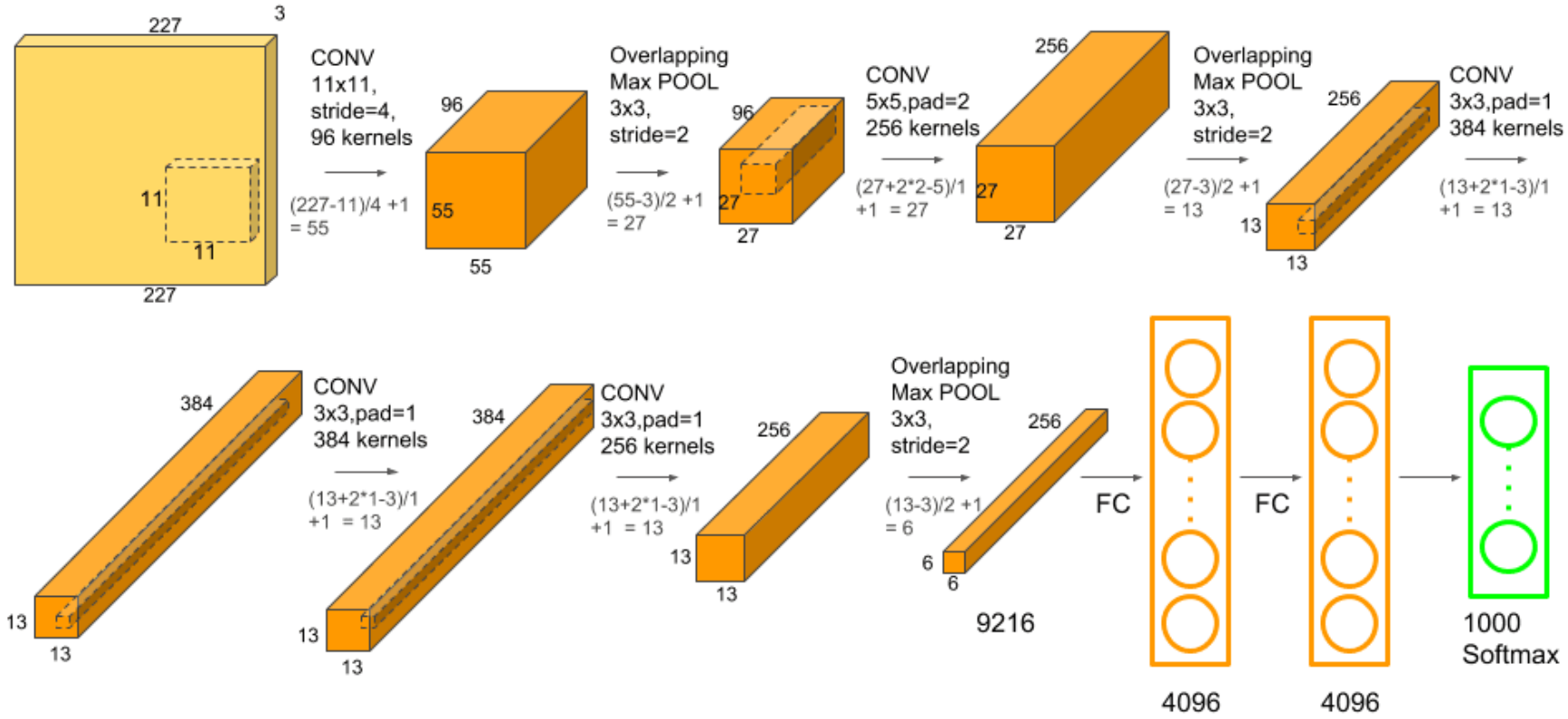
Architecture Overview:

1. Input: 32x32 grayscale image
2. C1 - Convolutional layer: 6 filters of size 5x5 → Output: 28x28x6
3. S2 - Downsampling (Avg Pooling): 2x2 → Output: 14x14x6
4. C3 - Convolutional layer: 16 filters → Output: 10x10x16
5. S4 - Downsampling (Avg Pooling): 2x2 → Output: 5x5x16
6. C5 - Fully connected convolution: Output: 120
7. F6 - Fully connected: Output: 84
8. Output Layer: Fully connected with 10 units (for classification)

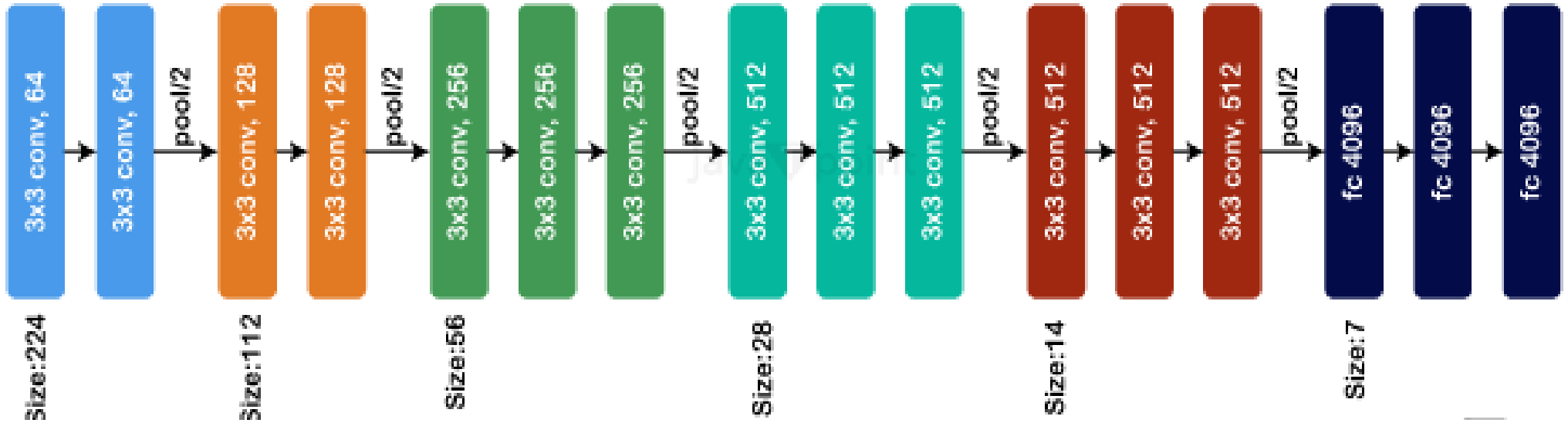
Examples of CNN Architectures

- AlexNet (2012): Introduced ReLU activation, dropout, and GPU training. Won ImageNet 2012.
- VGGNet (2014): Uses very small (3x3) filters and a deep architecture (16-19 layers).
- GoogLeNet (Inception) (2014): Introduced Inception modules to reduce parameters while maintaining depth.
- ResNet (2015): Introduced residual connections to enable very deep networks (e.g., 50,101 layers).
- DenseNet (2016): Each layer receives input from all previous layers (dense connectivity).
- MobileNet (2017): Lightweight CNN for mobile and embedded vision applications.

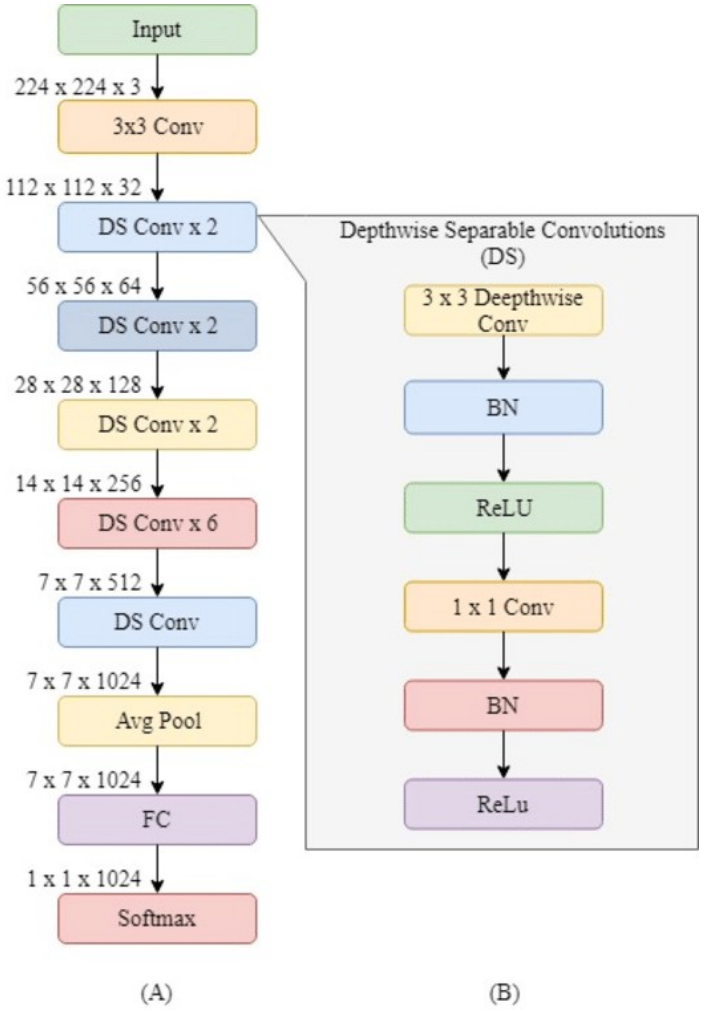
AlexNet


227
3
227
11
11
CONV
11x11,
stride=4,
96 kernels
(227-11)/4 +1
= 55
96
55
55
Overlapping
Max POOL
3x3,
stride=2
(55-3)/2 +1
= 27
96
27
27
CONV
5x5,pad=2
256 kernels
(27+2*2-5)/1
+1 = 27
256
27
27
Overlapping
Max POOL
3x3,
stride=2
(27-3)/2 +1
= 13
256
13
13
CONV
3x3,pad=1
384 kernels
(13+2*1-3)/1
+1 = 13
384
13
13
CONV
3x3,pad=1
384 kernels
(13+2*1-3)/1
+1 = 13
384
13
13
CONV
3x3,pad=1
256 kernels
(13+2*1-3)/1
+1 = 13
256
13
13
Overlapping
Max POOL
3x3,
stride=2
(13-3)/2 +1
= 6
256
6
6
9216
FC
4096
FC
4096
1000
Softmax

VGGNet


3x3 conv, 64
3x3 conv, 64
pool/2
3x3 conv, 128
3x3 conv, 128
pool/2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool/2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool/2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool/2
fc 4096
fc 4096
fc 4096
Size:224
Size:112
Size:56
Size:28
Size:14
Size:7

VGGNet


Input
224 x 224 x 3
3x3 Conv
112 x 112 x 32
DS Conv x 2
56 x 56 x 64
DS Conv x 2
28 x 28 x 128
DS Conv x 2
14 x 14 x 256
DS Conv x 6
7 x 7 x 512
DS Conv
7 x 7 x 1024
Avg Pool
7 x 7 x 1024
FC
1 x 1 x 1024
Softmax
(A)
Depthwise Separable Convolutions (DS)
3 x 3 Deepthwise Conv
BN
ReLU
1 x 1 Conv
BN
ReLu
(B)

Python