AI Bootcamp

Image Classification With Neural Networks

Module 19 Day 2

Class Objectives

By the end of class, you will be able to:

1. Describe convolution.
2. Use **`Conv2D`** and **`MaxPooling2D`** layers in TensorFlow.
3. Create convolutional neural networks (CNNs) for image classification.
4. Augment images for training CNNs.

Instructor **Demonstration**

Introduction to Convolution

Convolution

A mathematical operation that blends or combines two sets of data to create a third set.

Consider numerical input data arranged in a 9x9 array, where each number corresponds to a pixel value.

Then consider a smaller, 3x3 array also containing numerical values. This smaller grid is a **filter** and all the values contained in it are referred to collectively as the **kernel**.

Input data

9x9

	0	1	2	3	4	5	6	7	8
0	0.92	0.63	0.21	0.67	0.88	0.68	0.06	0.89	0.23
1	0.93	0.84	0.24	0.88	0.74	0.20	0.29	0.18	0.29
2	0.67	0.51	0.29	0.82	0.31	0.68	0.48	0.65	0.82
3	0.04	0.52	0.10	0.57	0.81	0.85	0.78	0.93	0.13
4	0.75	0.58	0.15	0.62	0.52	0.97	0.50	0.90	0.99
5	0.98	0.96	0.82	0.14	0.31	0.63	0.87	0.17	0.97
6	0.73	0.55	0.63	0.56	0.92	0.42	0.98	0.38	0.78
7	0.25	0.51	0.45	0.96	0.07	0.87	0.75	0.46	0.13
8	0.70	0.82	0.31	0.18	0.56	0.53	0.27	0.35	0.87

Filter

3x3

	0	1	2
0	1	0	-1
1	2	0	-2
2	1	0	-1

Convolution Calculation 1

Calculation for the 3x3 receptive field centered on (1, 1)

	0	1	2	3	4	5	6	7	8
0	0.92	0.63	0.21	0.67	0.88	0.68	0.06	0.89	0.23
1	0.93	0.84	0.24	0.88	0.74	0.20	0.29	0.18	0.29
2	0.67	0.51	0.29	0.82	0.31	0.68	0.48	0.65	0.82
3	0.04	0.52	0.10	0.57	0.81	0.85	0.78	0.93	0.13
4	0.75	0.58	0.15	0.62	0.52	0.97	0.50	0.90	0.99
5	0.98	0.96	0.82	0.14	0.31	0.63	0.87	0.17	0.97
6	0.73	0.55	0.63	0.56	0.92	0.42	0.98	0.38	0.78
7	0.25	0.51	0.45	0.96	0.07	0.87	0.75	0.46	0.13
8	0.70	0.82	0.31	0.18	0.56	0.53	0.27	0.35	0.87

*

	0	1	2
0	1	0	-1
1	2	0	-2
2	1	0	-1

= (0.92 x 1) + (0.63 x 0) + (0.21 x -1)
= (0.93 x 2) + (0.84 x 0) + (0.24 x -2) =
(0.67 x 1) + (0.51 x 0) + (0.29 x -1)

0.92 + 0.00 + -0.21 + 1.86 + 0.00 + -0.47 + 0.67 + 0.00 + -0.29 = **2.47**

Convolution Calculation 2

Calculation for the 3x3 receptive field centered on (1, 2)

	0	1	2	3	4	5	6	7	8
0	0.92	0.63	0.21	0.67	0.88	0.68	0.06	0.89	0.23
1	0.93	0.84	0.24	0.88	0.74	0.20	0.29	0.18	0.29
2	0.67	0.51	0.29	0.82	0.31	0.68	0.48	0.65	0.82
3	0.04	0.52	0.10	0.57	0.81	0.85	0.78	0.93	0.13
4	0.75	0.58	0.15	0.62	0.52	0.97	0.50	0.90	0.99
5	0.98	0.96	0.82	0.14	0.31	0.63	0.87	0.17	0.97
6	0.73	0.55	0.63	0.56	0.92	0.42	0.98	0.38	0.78
7	0.25	0.51	0.45	0.96	0.07	0.87	0.75	0.46	0.13
8	0.70	0.82	0.31	0.18	0.56	0.53	0.27	0.35	0.87

*

	0	1	2
0	1	0	-1
1	2	0	-2
2	1	0	-1

= (0.63 x 1) + (0.21 x 0) + (0.67 x -1)
= (0.84 x 2) + (0.24 x 0) + (0.88 x -2) =
(0.51 x 1) + (0.29 x 0) + (0.82 x -1)

0.63 + 0.00 + -0.67 + 1.68 + 0.00 + -1.76 + 0.51 + 0.00 + -0.82 = **-0.43**

Output Array

	0	1	2	3	4	5	6	7	8
0	0.92	0.63	0.21	0.67	0.88	0.68	0.06	0.89	0.23
1	0.93	0.84	0.24	0.88	0.74	0.20	0.29	0.18	0.29
2	0.67	0.51	0.29	0.82	0.31	0.68	0.48	0.65	0.82
3	0.04	0.52	0.10	0.57	0.81	0.85	0.78	0.93	0.13
4	0.75	0.58	0.15	0.62	0.52	0.97	0.50	0.90	0.99
5	0.98	0.96	0.82	0.14	0.31	0.63	0.87	0.17	0.97
6	0.73	0.55	0.63	0.56	0.92	0.42	0.98	0.38	0.78
7	0.25	0.51	0.45	0.96	0.07	0.87	0.75	0.46	0.13
8	0.70	0.82	0.31	0.18	0.56	0.53	0.27	0.35	0.87

*

	0	1	2
0	1	0	-1
1	2	0	-2
2	1	0	-1

=

2.47							

	0	1	2	3	4	5	6	7	8
0	0.92	0.63	0.21	0.67	0.88	0.68	0.06	0.89	0.23
1	0.93	0.84	0.24	0.88	0.74	0.20	0.29	0.18	0.29
2	0.67	0.51	0.29	0.82	0.31	0.68	0.48	0.65	0.82
3	0.04	0.52	0.10	0.57	0.81	0.85	0.78	0.93	0.13
4	0.75	0.58	0.15	0.62	0.52	0.97	0.50	0.90	0.99
5	0.98	0.96	0.82	0.14	0.31	0.63	0.87	0.17	0.97
6	0.73	0.55	0.63	0.56	0.92	0.42	0.98	0.38	0.78
7	0.25	0.51	0.45	0.96	0.07	0.87	0.75	0.46	0.13
8	0.70	0.82	0.31	0.18	0.56	0.53	0.27	0.35	0.87

*

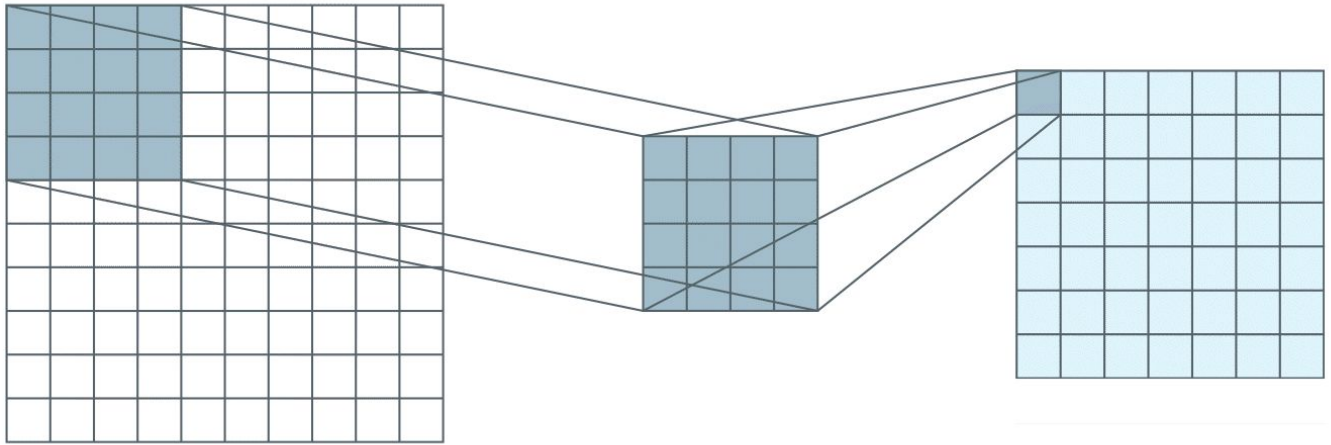
	0	1	2
0	1	0	-1
1	2	0	-2
2	1	0	-1

=

2.47	-0.43						

What do you notice about the shape of the **output array**?

Convolution



Kernels: Common kernels and their effects in CNNs

Sobel kernel
Detects horizontal edges

1	0	-1
2	0	-2
1	0	-1

Sobel kernel
Detects vertical edges

-1	-2	-1
0	0	0
1	2	1

Emboss kernel
Creates 3D embossed effect

-2	-1	0
-1	1	1
0	1	2

Box blur kernel
Blurs an image by averaging pixel values

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

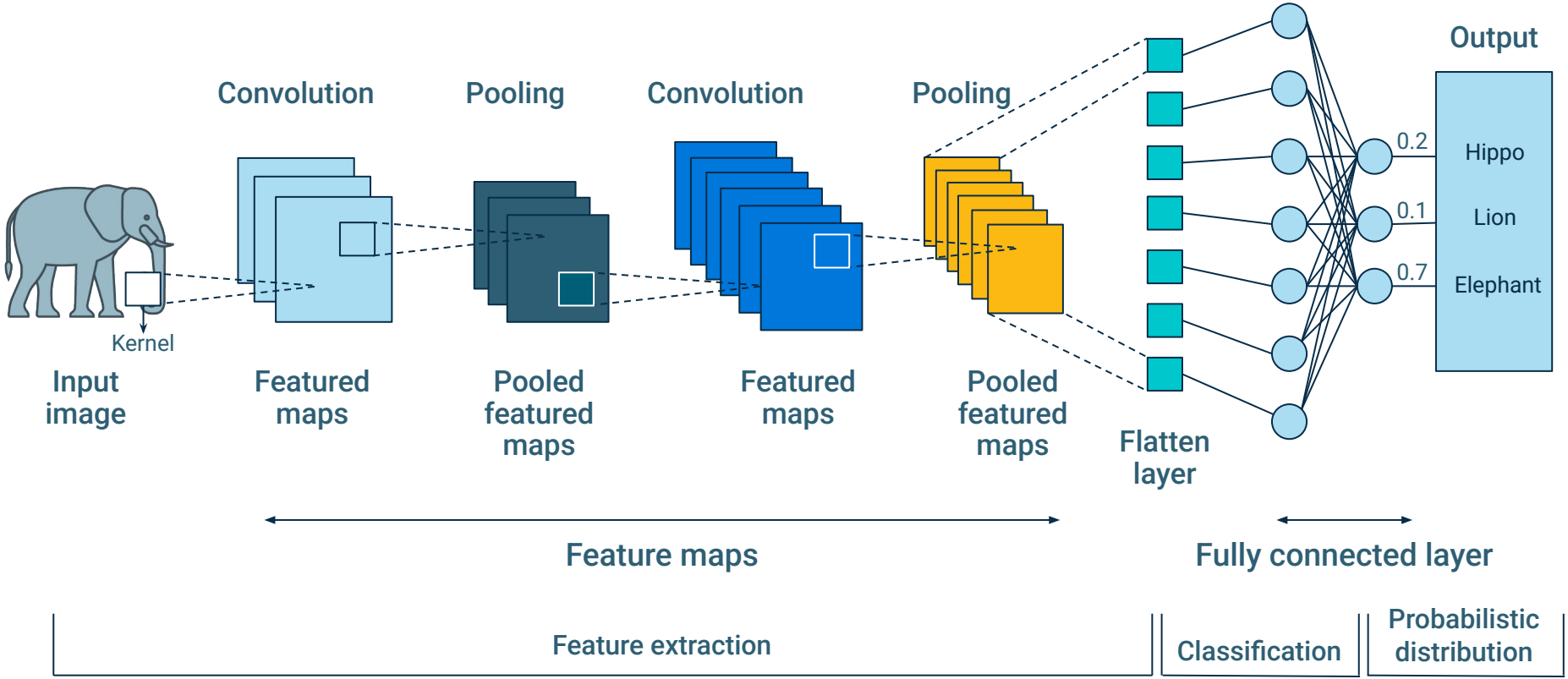
Gaussian kernel
Reduces noise and detail, effectively blurring or smoothing the image

1	2	1
2	4	2
1	2	1

Identity kernel
Makes no changes to the original image

0	0	0
0	1	0
0	0	0

CNNs


Convolution
Pooling
Convolution
Pooling
Output
0.2
Hippo
0.1
Lion
0.7
Elephant
Kernel
Input image
Featured maps
Pooled featured maps
Featured maps
Pooled featured maps
Flatten layer
Feature maps
Fully connected layer
Feature extraction
Classification
Probabilistic distribution

Instructor **Demonstration**

Building a Basic CNN

Activity:

Build a CNN

In this activity, you will use the Keras library to create, compile, fit, and validate a basic CNN binary classification model.

Suggested Time:
15 Minutes

Time’s up!

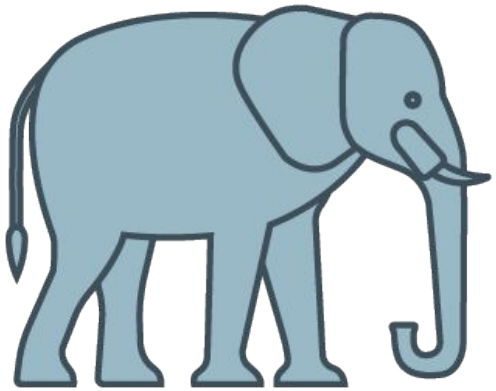
Let’s review

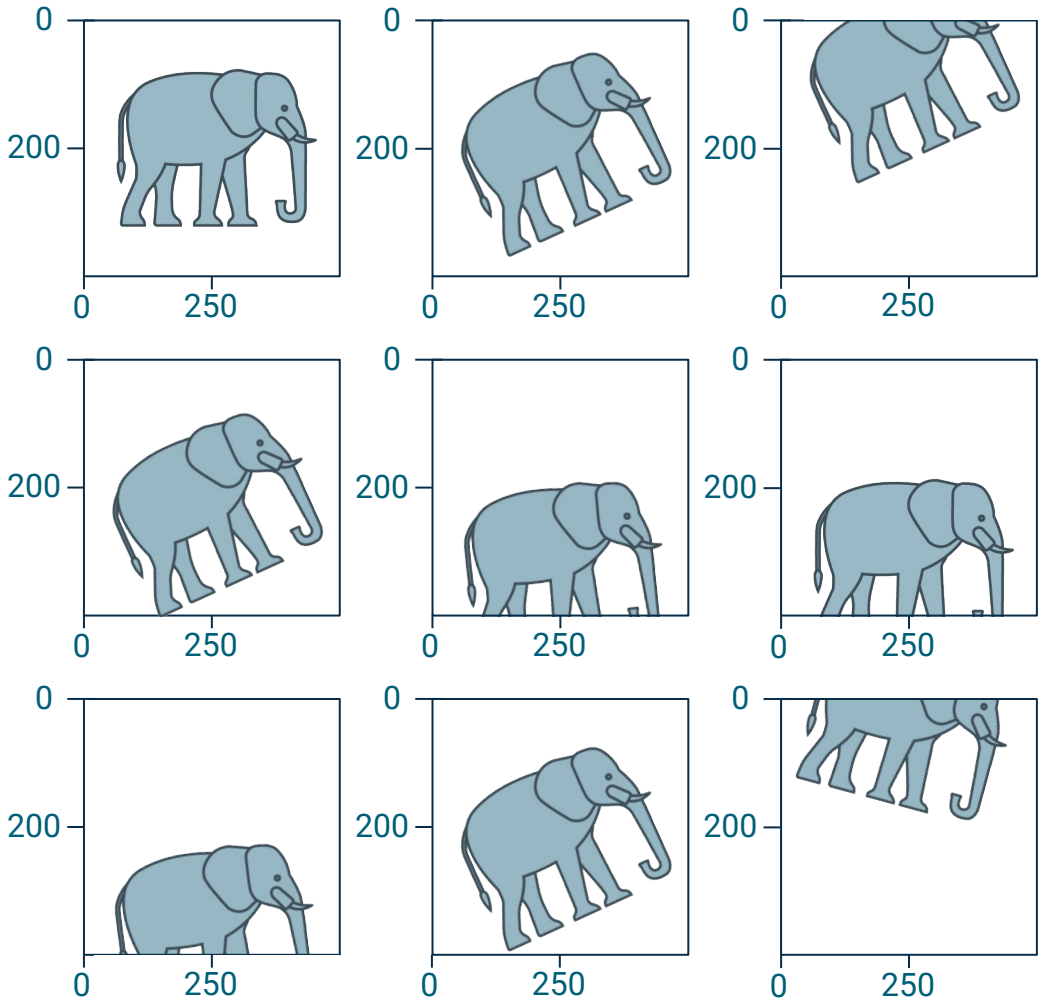
Questions?

Instructor **Demonstration**

Augmenting Data

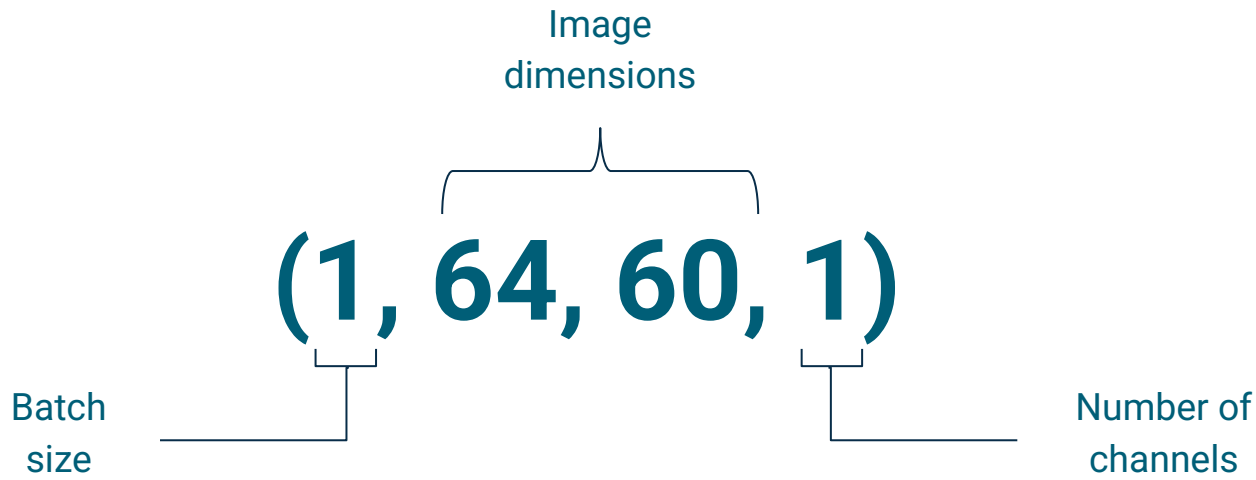
Augmentation





Augmenting Data

Working with `ImageDataGenerator`



Augmenting Data

Adding dimensions to the input data shape

```
# Add a batch size dimension
new_img = np.expand_dims(img, axis=0)
```

```
# Add a channel dimension
new_img = np.expand_dims(new_img, axis=-1)
```

Activity:

Augmenting One Image

In this activity, you will use an instance of **`ImageDataGenerator`** to augment an image from the DeFungi database and then compare it to the original image.

Suggested Time:
10 Minutes

Time’s up!

Let’s review

Questions?

Activity:

Augmenting Fungi

In this activity, you will apply the augmentation process to the entire fungi training dataset.

Suggested Time:
15 Minutes

Time’s up!

Let’s review

Questions?

Break

15 mins

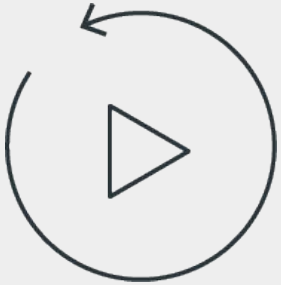
Activity:

Building CNN From Scratch

In this activity, you will use Pillow, NumPy, and Keras to code the entire process of building a CNN.

Suggested Time:
45 Minutes

Questions?



Let’s **recap**

Recap

After today's lesson, you are able to:

1. Describe convolution.
2. Use **`Conv2D`** and **`MaxPooling2D`** layers in TensorFlow.
3. Create CNNs for image classification.
4. Augment images for training CNNs.

Next

In the next lesson, you'll begin to work with **branching** CNN models.

Questions?

The End