

# LECTURE 8. CNN

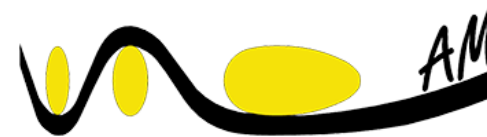
(Convolutional Neural Network)

## Part II. Building the CNN Classifier

MANU 465

Ahmad Mohammadpanah

PhD, PEng



[AIntelligentManufacturing.com](http://AIntelligentManufacturing.com)

## Extra Image Pre-Processing, Image Augmentation

In Part I, we became familiar with some basic image preprocessing, such as Scaling, Resizing, Feature Mapping via Convolution, Max Pooling, and Flattening. In order to increase the number of Images, we may apply:

### 1. Shearing:

Image will be distorted along an axis. It's usually used to augment images so that computers can see how humans see things from different angles.

### 2. Zooming

The zoom augmentation either randomly zooms in on the image or zooms out of the image.

### 3. Flipping

The vertical and horizontal flip augmentation means they will reverse the pixels rows or column-wise respectively.



## Implement in Python:

```
from keras.preprocessing.image import ImageDataGenerator
```

```
train_datagen = ImageDataGenerator(rescale = 1./255,  
                                   shear_range = 0.2,  
                                   zoom_range = 0.2,  
                                   horizontal_flip = True)
```

```
training_set = train_datagen.flow_from_directory('Dataset/Training_Set',  
                                                  target_size = (64, 64),  
                                                  batch_size = 32,  
                                                  class_mode = 'binary')
```

```
test_set = test_datagen.flow_from_directory('Dataset/Test_Set',  
                                             target_size = (64, 64),  
                                             batch_size = 32,  
                                             class_mode = 'binary')
```

For multiple classes, just change: `class_mode = 'categorical'`

In Part I, we learned how to extract features, using different Kernels



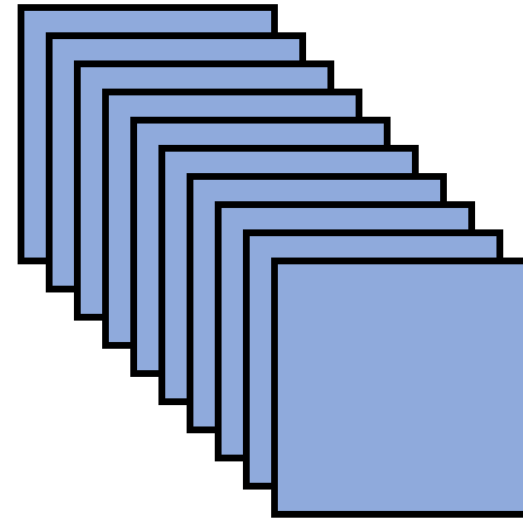
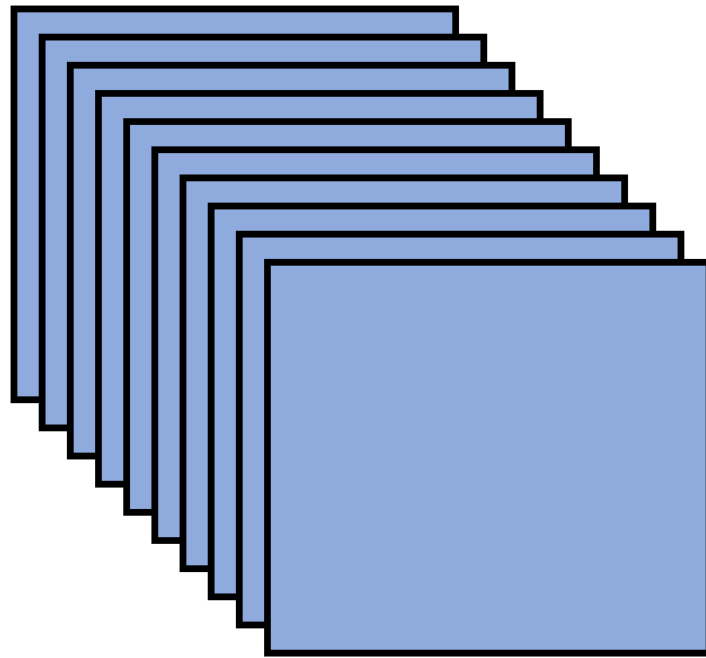
0	0	0	0	0			
0	0	-1	0	0			
0	-1	0	0	0			
0	0	0	-2	-1	0	0	
0	0	0	-1	1	1	0	
0	0	0	0	0	0	0	
	0	0	0	0	-1	0	0
		0	-1	5	-1	0	
		0	0	-1	0	0	
		0	0	0	0	0	

Implement in Python:

```
conv=keras.layers.Conv2D(filters=10, kernel_size=3, strides=1, padding="same", activation='relu')
```

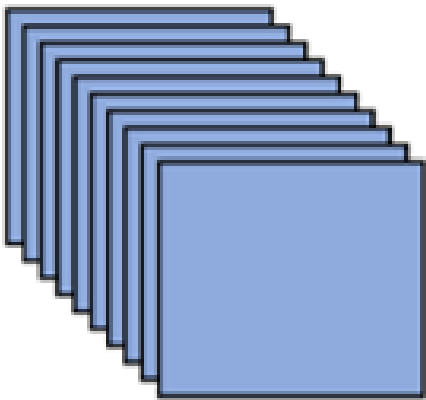
Python decides which Kernel to use, by optimizing the kernel enteries through the backpropagation process.

# Putting All Together





we repeat the same process for all the images, for example

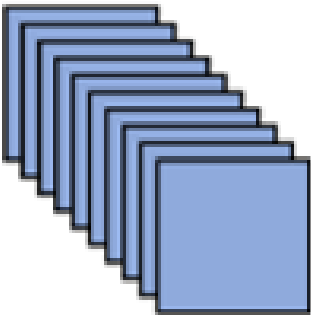
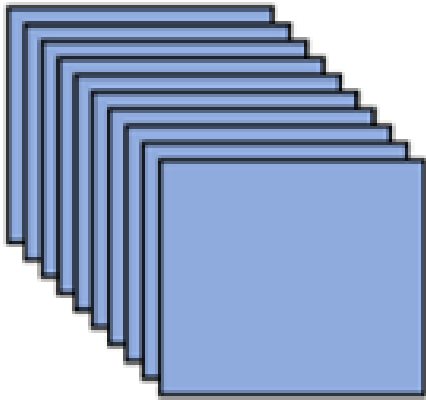


.

.

.

.

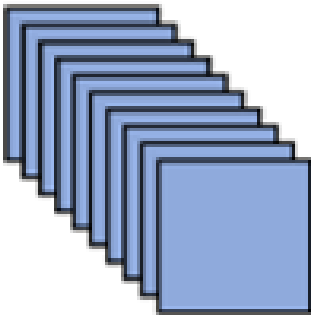
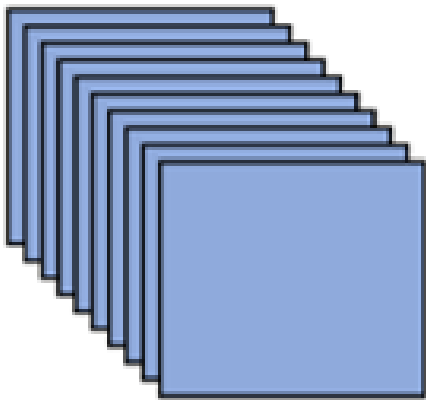


.

.

.

.

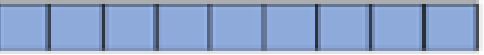


.

.





.

.



# Neural Network Classification for Images

Each image is an input for  $X$  with a label ( $y$ ). Now, we can build an ANN model (similar to ANN classification).

Images	Input (X)	Label (y)
1		Cat
2		Dog
3		cat
.		
.		
.		
10000		Cat

# Implement in Python

(See, examples 16 & 17 on Canvas)

Please note, the following differences, between Binary and Multiple Classifications:

## 1. Output Layer:

```
Model.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
```

For multiple classes, just change: activation='softmax'

## 2. Compiling:

For binary classification:

```
Model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

For Multiclass classification:

```
Model.compile(loss='categorical_crossentropy', optimizer='nadam', metrics=['accuracy'])
```



# Examples

1. Please check examples on Canvas, 16 & 17 for details.
  - Cats & Dogs
  - Metal Casting Defect
2. Tutorial 6. CNN with Multiple Classes (*Leaves Identification*)
3. Assignment 6 (*Shoes & Personality*)

# An Industry Project (2021, North BC Sawmill, by Ahmad)

## Log Debarker

