

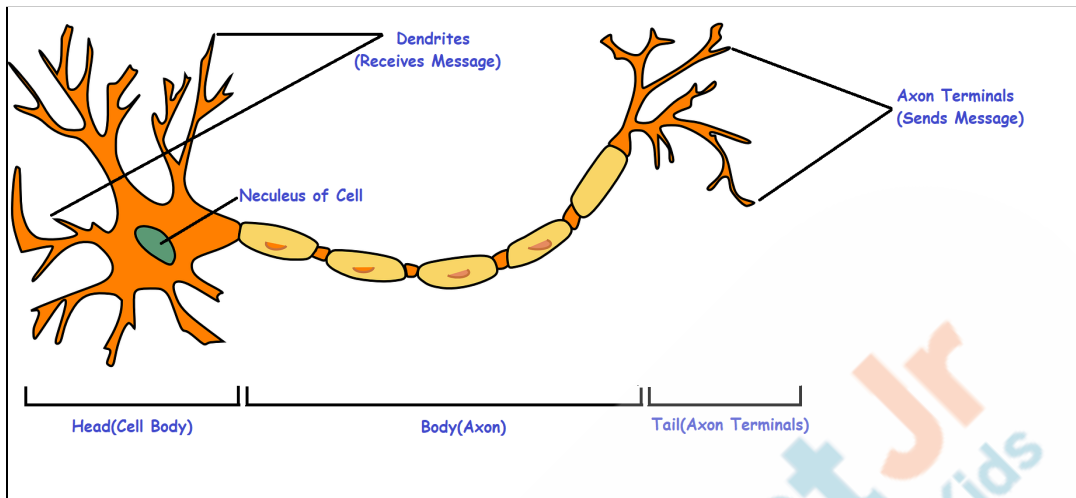


Topic	DEFINE A CNN MODEL	
Class Description	The student will learn to define a Convolutional Neural Network(CNN) for training the model which can classify images.	
Class	PRO C112	
Class time	45 mins	
Goal	<ul style="list-style-type: none"> Understand the Convolutional Neural Network(CNN). Define a CNN using Keras to train the model for image classification. 	
Resources Required	<ul style="list-style-type: none"> Teacher Resources: <ul style="list-style-type: none"> Laptop with internet connectivity Webcam Earphones with mic Notebook and pen Smartphone Student Resources: <ul style="list-style-type: none"> Laptop with internet connectivity Webcam Earphones with mic Notebook and pen 	
Class structure	Warm-Up Teacher-led Activity 1 Student-led Activity 1 Wrap-Up	10 mins 20 mins 10 mins 05 mins
Credit	Teachable Machines by Google. Tensorflow by Google Brain team. Keras by Google Engineer Francois Chollet.	
WARM-UP SESSION - 10 mins		

<div>  <p>Teacher Starts Slideshow</p> <p>Slide 1 to 3</p> <p>Refer to speaker notes and follow the instructions on each slide.</p> </div>	
Teacher Action	Student Action
<p>Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today?</p> <p>Following are the WARM-UP session deliverables:</p> <ul style="list-style-type: none"> Greet the student. Revision of previous class activities. Quizzes. 	<p>ESR: Hi, thanks! Yes I am excited about it!</p> <p>Click on the slide show tab and present the slides</p>
<p>WARM-UP QUIZ Click on In-Class Quiz</p>	
<div>  <p>Continue WARM-UP Session</p> <p>Slide 4 to 17</p> </div>	
<p>Following are the session deliverables:</p> <ul style="list-style-type: none"> Appreciate the student. Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students. 	
Teacher Action	Student Action
<p>Now, once we have the dataset ready for training, we need to define the model for training.</p> <p>We are going to use the Convolutional Neural Network(CNN) to define an image classification model.</p> <p>Well, this sounds like some pretty heavy terms to</p>	

<p>understand, right?</p> <p>Although there are multiple types of neural networks available, in today's class we are going to break down the Convolutional Neural Network(CNN) process and understand the concepts behind how computers/machines are trained using CNN.</p> <p>Before we can actually do that, let's understand the human brain neural network, which led to the idea of neural networks in machine learning.</p> <p>Do you know what a brain cell looks like?</p> <p>Our brain is made up of billions(1 billion = 1,000,000,000) of neurons, these are brain nerve cells which are the messengers of the information from the brain to the other parts of the body.</p> <p><i>Note: We are not going to understand the complete biological structure of the neurons in this class.</i></p> <p>For now, we can understand that, the neuron contains:</p> <p>Dendrites(structured like the branches of a tree) which receive messages.</p> <p>Axon helps in passing away the messages from the cell body.</p> <p>Axon Terminal is the intersection point between two neurons from where the message is sent.</p>	<p>ESR: Yes.</p> <p>ESR: Varied.</p>
---	--



One neuron sends messages to another in the form of electrical and chemical signals.

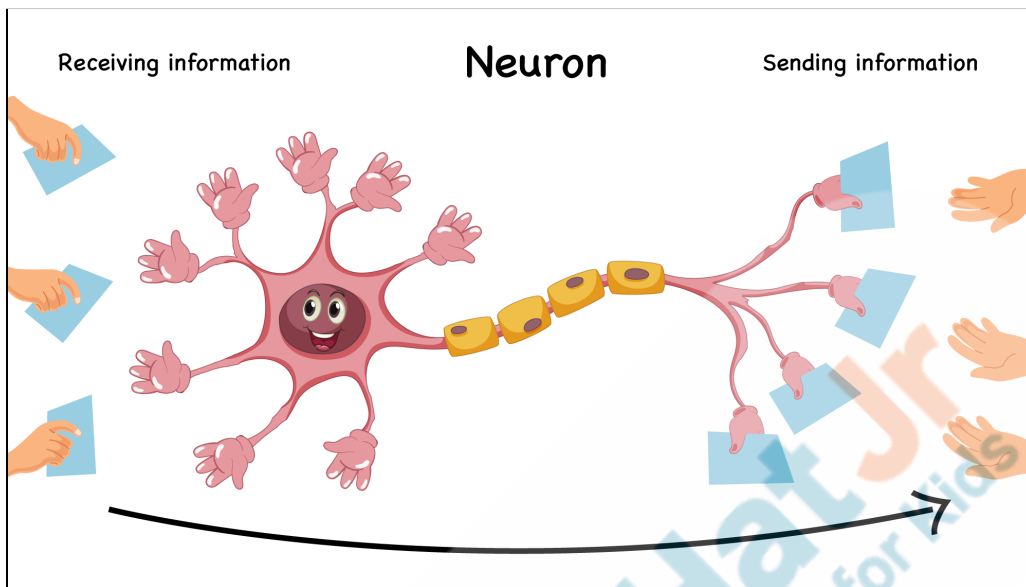
The head of the neurons receives the message from the other neurons and then the tail of the neurons sends the message to other neurons. This way, the message is transferred to all the nerves present in different parts of the body!

For example, when we feel pain due to some injury, the sensory neurons in the brain receive the information of the pain, and then it sends the information to the part where it's hurting.

Well now that you know the information about the pain is also transferred through the brain cells, if somehow this information is not passed then we won't even feel the pain no matter how big the injury is!

Isn't this incredible how the brain works?

ESR: Yes.

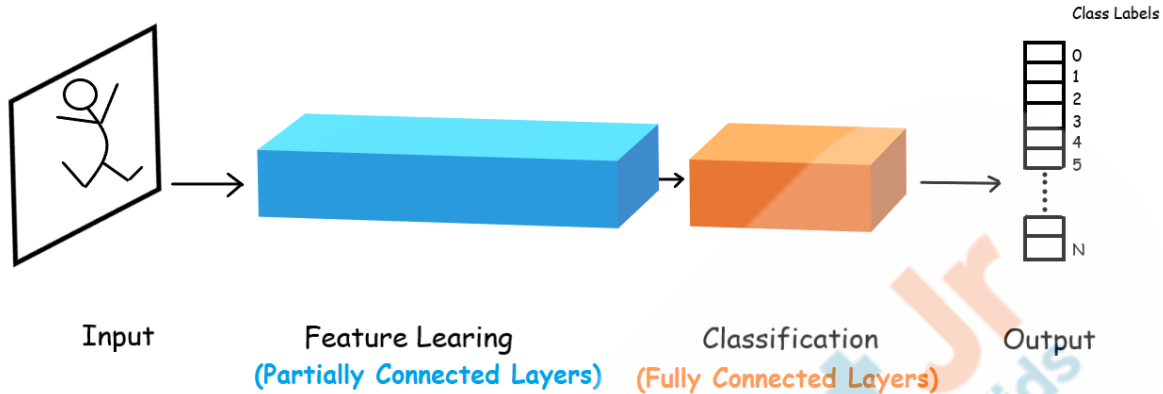


Now that we know how the brain is a web of interconnected neurons which work together to help our body to function, let's understand what neural networks are.

Neural Networks are interconnected layers of **nodes** (like neurons in the human brain) which take **input data** and pass it through **multiple layers of neural network** to train the machine which can classify the input according to its **class** (category).

<h2 style="text-align: center;">Neural Network</h2>	
<p>In today's class, we will deep dive into a specific type of neural network called Convolutional Neural Network!</p> <p>Are you excited?</p> <p>Let's get started.</p>	<p>ESR: Yes.</p>
<p>Teacher Ends Slideshow </p>	
<p>TEACHER-LED ACTIVITY - 20 mins</p>	
<p>Teacher Initiates Screen Share</p>	
<p><u>ACTIVITY</u></p> <ul style="list-style-type: none"> • Understanding Convolutional Neural Network. • Define a Convolutional Neural Network To Class Chest X-Ray Image with Pneumothorax disease. 	

Teacher Action	Student Action
<p>Understanding Convolutional Neural Network Architecture:</p> <p>Convolutional Neural Network(CNN) is a type of artificial neural network which is generally used for image classification or other computer vision tasks.</p> <p>The CNN is made of multiple layers connected to each other through nodes like any neural network as we discussed.</p> <p>All the layers of CNN can be categorized into:</p> <p>Feature Learning:</p> <ul style="list-style-type: none"> Feature learning helps the machine to learn multiple features of the image. These layers take the input image and break it down into multiple images to extract the features of the image one by one. <p>Classification:</p> <ul style="list-style-type: none"> Classification helps to map the extracted features with each class in which the image belongs. <p>We will talk about these layers in detail.</p> <p>We can say that, CNN processes the images by extracting features from the image, a set of features will be extracted for each class(category), which will be used to identify the unseen image.</p>	



Now that we know CNN is made of Feature Learning Layers and Classification Layers, let's start understanding the Feature Learning Layers and Classification Layers architecture in detail.

Before we understand that, remember, images are represented with pixel values in the 2D array format. The pixel value ranges from [0, 255].

Note: Help the student to recollect how images are pixel values arrays(that we covered in earlier classes).

The 0 pixel value represents the black and 255 represents white color.

Feature Learning Layers and Classification Layers have sub-layers which perform mathematical computations using images as arrays, to perform the image classification, but for now let's understand the architecture of these layers before we can actually use these in the program.

Feature Learning Architecture:

Feature Learning process will have one or more:

Convolutional Layer:

- This is the main building block of CNN, where the main feature extraction occurs. This layer needs:
 - An **input image** as an array of pixel values.
 - A **feature detector**(also known as **filter/kernel**) to extract the features.

Feature extraction:


The feature detector breaks down the one input image into multiple images containing one or the other portion like edges, lines, curves or any other shapes, called **features**.

This process of extracting features is known as **convolution**.

The convolution is a mathematical computation between two arrays, the **image array** and the **filter array** which gives a new image array.

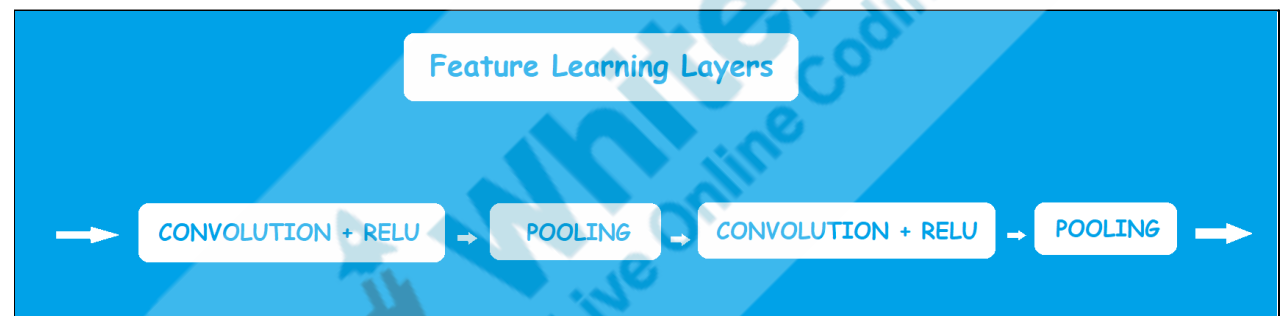
Note: Refer **ADDITIONAL ACTIVITIES** to understand the [mathematical computation in convolutional layer](#).

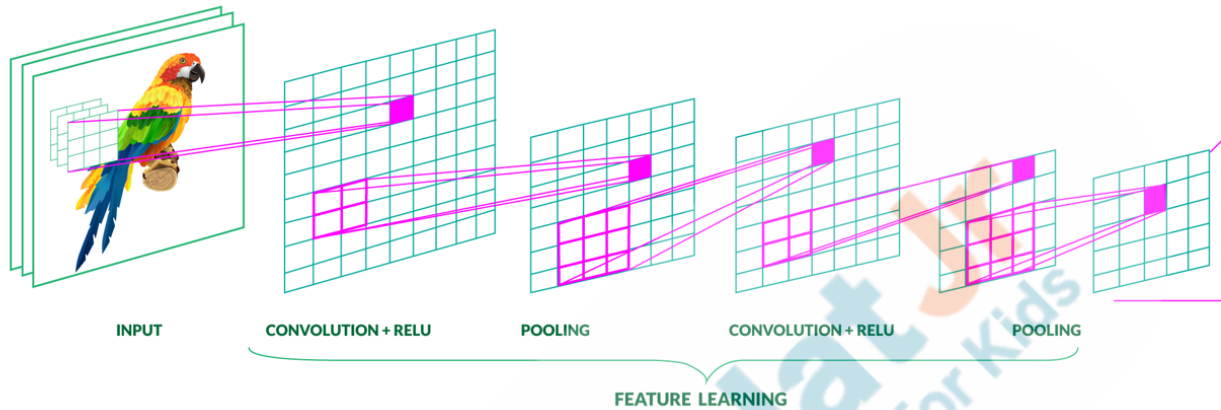
Visually, we can understand that the feature detector/filter moves over the image to

<p>extract features(Visualization: Open Teacher Activity 1).</p>	
<div data-bbox="175 422 1398 842">  </div>	
<ul style="list-style-type: none"> After each convolution operation, a CNN uses the ReLU activation method that helps (which we will understand in a while) to improve the process of feature extraction. <p>Pooling Layer:</p> <ul style="list-style-type: none"> This layer is used to reduce the dimension(width and height) of the image before feeding it to the next layer. This helps to keep the most important features and discard the rest. Also, it speeds up the model training process as size/dimension of image is reduced in this layer. <p><u>The whole Feature Learning process can have multiple Convolutional Layers and Pooling Layers:</u></p> <p>For example, if we want to train the machine to classify the image into 4 or more different categories of birds, like parrot, swan, pigeon, peacock, etc.</p>	

- The first layer starts by detecting simple features in the parrot image like edges and corners and reduces dimension using a pooling layer.
- The second layer takes input from the first layer and can detect the horizontal and vertical lines in the image and reduces dimension using a pooling layer.
- As we move on with layers, we detect complex features in the image like objects like eyes, feathers, beaks etc, taking input from the previous layers.

This way, all the features of an image are extracted layer by layer.



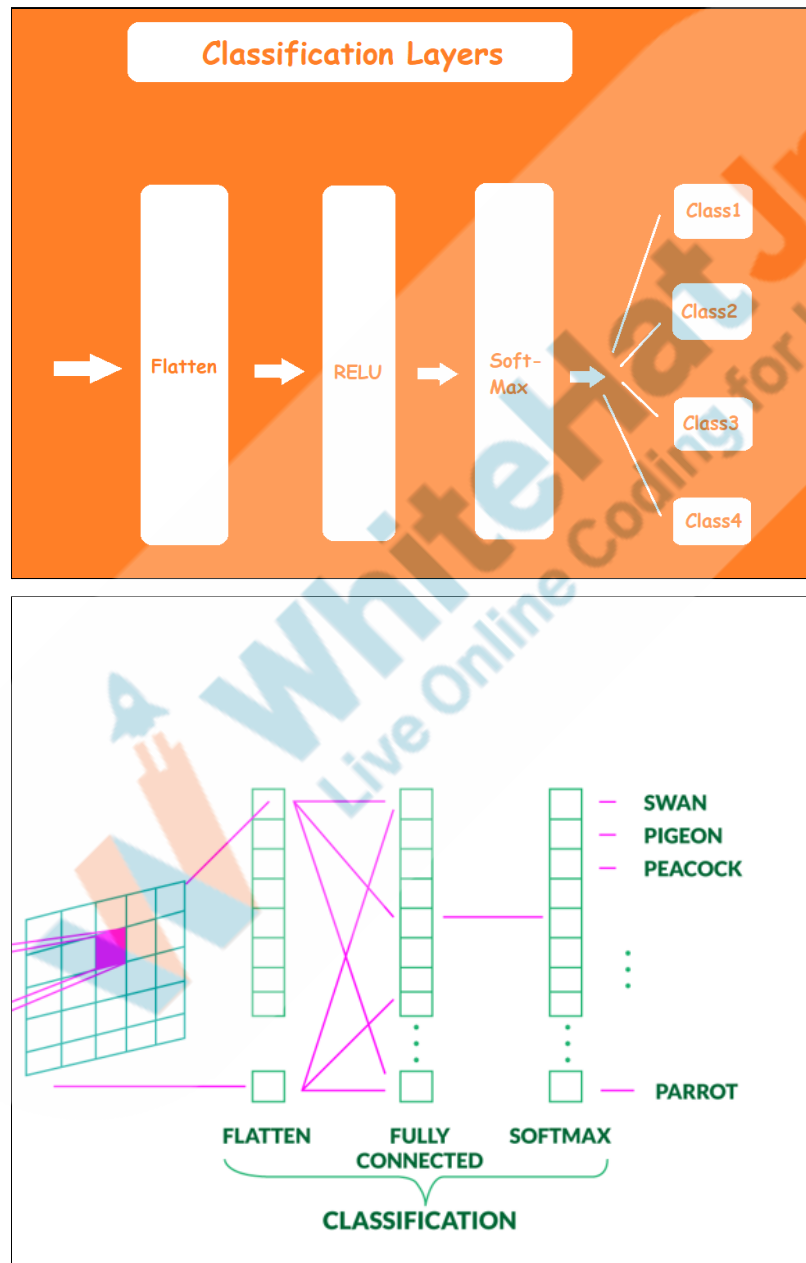


Once the features are extracted, we still need to be able to classify them, which is done by the second part of a convolutional network, the **classifier**.

Classification:

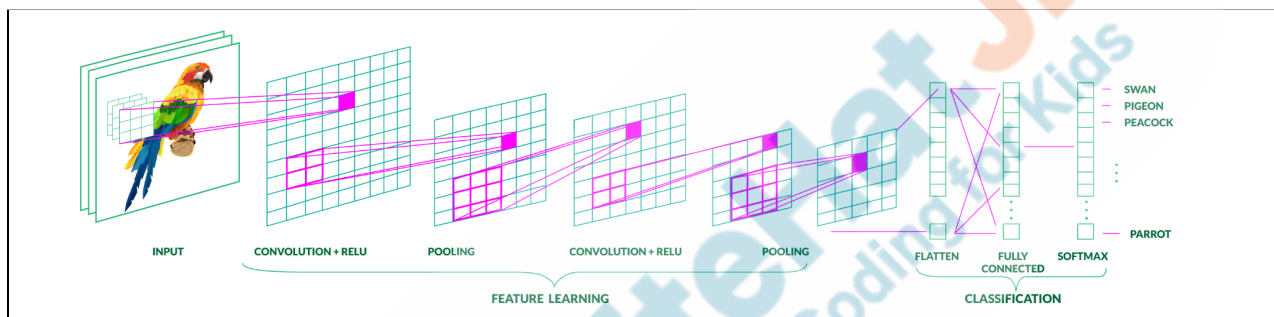
- This part consists of fully connected layers that take high level abstracted features from our input, rather than the raw input pixels.
- The output from the Feature Learning Layers (Convolutional Layers + Pooling Layers) is first converted into a 1D array, known as the **Flatten Layer**.
- Then **Dense Layers**, also called Fully Connected Layers, with **ReLU** and **Softmax** activation methods are used (which we will understand in a while writing the program) to classify the image into different classes.

We can teach the machine to identify each category of the bird (like **parrot**, **swan**, **pigeon**, **peacock**, etc.) by passing it through classification layers after taking input from layers of feature learning process!



CNN Architecture:

- **Feature Learning layers:**
 - Convolutional layers
 - Activation layer
 - Pooling layer
- **Classification layers:**
 - Flatten
 - Dropout layer
 - Dense layer(ReLU & Softmax)



Now that we know the basic architecture of a CNN, let's start writing the program to build(or define) a CNN model using Keras library.

Open [Teacher Activity 2](#) for the boilerplate code and run all the cells.

In the previous class, we preprocessed the image dataset containing Chest X-Ray images with and without Pneumothorax disease infection.

Note: Help the student to recollect the **ImageDataGenerator Class(Keras)** and **Data Augmentation techniques**.

We used the **ImageDataGenerator** class to apply data augmentation techniques(like rotation, width shift etc.). The **ImageDataGenerator** class also converts each image into an array of pixels and assigns labels to each image.

Remember that for this the images are to be stored in a specific subdirectories structure with a master directory of image containing **Training**, **Validation** and **Testing** subdirectories, each having “**infected**” and “**uninfected**” subdirectories with respective images.

Training and **Validation** images are used during the training process, **Training** are images on which the model is trained and **Validation** images are used validating the training process output of each layer while the model is under training. These images are thus already known to the model.

Testing images are the unseen images(the image which are not used while training the model) which will be used to test the working of the model, whether the model is able to predict the correct class to which the image belongs!

In the previous class, we only added a few of the data augmentation to the training dataset. We can add a few data augmentation for the training dataset, this will provide more variety of images for training the model as it is difficult to detect features in the X-Ray disease image:

Rescale: Scaling the image pixel values from [0, 255] to [0, 1].

Height Shift: Shifting the image along the height of the image.

Zoom: Magnifying the image in or out.

Horizontal Flip: Randomly flipping images horizontally.

Vertical Flip: Randomly flipping images horizontally.

The teacher explains the boilerplate code.

```
# Random Data Augmentation(Rescale, Rotation, Flips, Zoom, Shifts) using ImageDataGenerator
training_data_generator = ImageDataGenerator(
    rescale = 1.0/255,
    rotation_range=40,
    width_shift_range=0.3,
    height_shift_range=0.3,
    zoom_range=0.3,
    horizontal_flip=True,
    vertical_flip=True,
    fill_mode='nearest')
```

```
# Image Directory
training_image_directory = "/content/PRO-M3-Pneumothorax-Image-Dataset/training_dataset"

# Generate Preprocessed Augmented Data
training_augmented_images = training_data_generator.flow_from_directory(
    training_image_directory,
    target_size=(180,180))
```

Found 200 images belonging to 2 classes.

Note: This is a part of the boilerplate code.

Since we are going to use validation image dataset during the training, we should also rescale the image from [0, 255] range to [0,1] using the **ImageDataGenerator** class in the same way.

We can apply other data augmentation techniques on validation dataset too, but for now we will only rescale the pixel values of the images.

The teacher explains the boilerplate code.

```
# Random Data Augmentation(Rescale) using ImageDataGenerator
validation_data_generator = ImageDataGenerator(rescale = 1.0/255)

# Image Directory
validation_image_directory = "/content/PRO-M3-Pneumothorax-Image-Dataset/validation_dataset"

# Generate Preprocessed Augmented Data
validation_augmented_images = validation_data_generator.flow_from_directory(
    validation_image_directory,
    target_size=(180,180))

Found 200 images belonging to 2 classes.
```

Note: *This is a part of the boilerplate code.*

Now that we have image data in the correct format, we need to first define the CNN Model.

For this, we will first need:

Sequential Class of the Model API(Keras): This class is used to define multiple layers(like convolution, pooling, activation & classification layer) of the model one after the other.

```
import tensorflow as tf

model = tf.keras.models.Sequential( [ ] )
```

[Teacher Activity 3](#)

Reference: [Model Class: The Sequential class](#)

```
model = tf.keras.models.Sequential([  
  
    # 1st Convolution & Pooling layer  
  
    # 2nd Convolution & Pooling layer  
  
    # 3rd Convolution & Pooling layer  
  
    # 4th Convolution & Pooling layer  
  
    # Flatten the results to feed into a Dense Layer  
  
    # Classification layer  
  
])
```

Now we will define the **1st Convolution Layer** for feature extraction.

For this we will need:

Conv2D Class: This class is used to define convolution filters that will extract features from the images.

There are 1D, 2D and 3D Conv Layers depending on the input data array. Our images data are passed as a 2D array pixel values to the Convolutional Layer, hence the Conv2D class.

This class takes many parameters, but for now in this model we will be using 4 parameters:

- **filters:** This is the number of filters(also called **kernels** or **feature detectors**) to be used. Filters are a small array of the image data which are randomly chosen by the convolutional layers.

- **kernel_size**: This is the size of each filter(also called as **kernels** or **feature detectors**) to be used.
- **activation**: The activation functions help Convolution layers to extract complex features. There are many activation functions that can be applied, we will be using ReLU(short for Rectified Linear Unit) Activation Function which helps to keep only the positive values in the 2D array.
- **input_shape**: This is the shape of the image on which the model will be trained. We will keep the input shape as 180x180x3

Teacher Activity 4

Reference: [Convolution Layer: Conv2D layer](#)

Teacher Activity 9

Reference: [Activation Layer : ReLU layer](#)

Let's define the **1st Conv2D** layer with the 4 parameters:

- Taking 64 filters(we can start by taking any number filter, while keeping in mind that starting with very few filters will require more CNN layers to extract features and taking way too many filters takes time for that particular layer) each of size 3x3.
- The activation function is 'relu'.
- And input shape as the 180x180 with 3 channels for RGB.

```
model = tf.keras.models.Sequential([

    # 1st Convolution & Pooling layer
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', input_shape=(180, 180, 3)),

    # 2nd Convolution & Pooling layer

    # 3rd Convolution & Pooling layer

    # 4th Convolution & Pooling layer

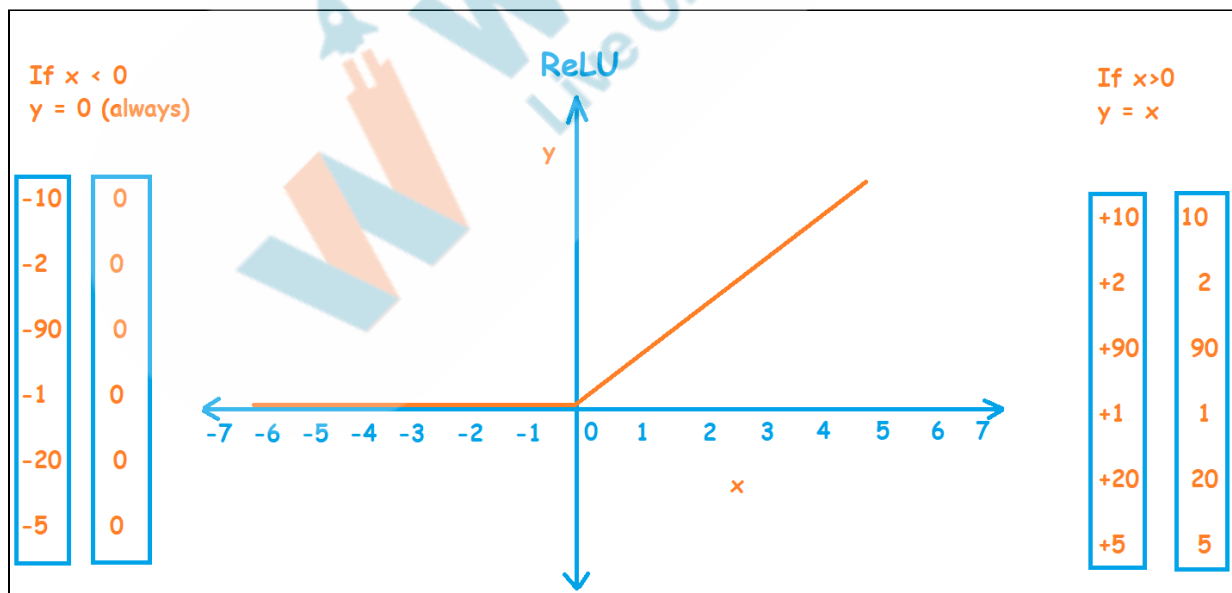
    # Flatten the results to feed into a Dense Layer

    # Classification layer

])
```

(Optional) Mathematically, ReLU is defined as a function, $y = f(x)$ such that it gives x for all values of $x > 0$ and 0 for all values of $x < 0$.

Note: Only explain the mathematical part of the ReLU function if the student is curious to understand what a ReLU function does!



Now that we have defined the 1st Conv2D layer, we can reduce the dimensions(width and height) of the image using **Max Pooling technique**, in this technique:

1. First, there is an input array(for example 4x4) and pool size(for example 2x2). Pool size is always smaller than the input array size.
2. Then the maximum value is taken from the sub-array of size equal to pool size.
3. The result after applying the Max Pooling will be the new array of size equal to the half of the size of the original input array.

Since our input array is 4x4, after max pooling the new array will be 2x2, hence reducing the dimension of the array.

MaxPooling2D Example

Input Array Size: 4x4 & Pool Size: 2x2

6	5	10	20
0	1	8	3
4	5	0	0
6	7	5	1

Maximum out of 6, 5, 0 & 1
is 6

6	5	10	20
0	1	8	3
4	5	0	0
6	7	5	1

Maximum out of 10, 20, 8, 3
is 20

6	5	10	20
0	1	8	3
4	5	0	0
6	7	5	1

Maximum out of 4, 5, 6, 7
is 7

6	5	10	20
0	1	8	3
4	5	0	0
6	7	5	1

Maximum out of 0, 0, 5, 1
is 5

6	20
7	5

We can reduce the dimensions of the image using:

MaxPooling2D Class: This class reduces the dimension(width and height) of the image after the Conv2D layer. Reducing the image dimension helps to extract the feature at a much faster rate.

The dimensions are reduced by taking the maximum value for a given pool size.

[Teacher Activity 5](#)

Reference: [Pooling Layer : MaxPooling2D layer](#)

```
model = tf.keras.models.Sequential([
    # 1st Convolution & Pooling layer
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', input_shape=(180, 180, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),

    # 2nd Convolution & Pooling layer

    # 3rd Convolution & Pooling layer

    # 4th Convolution & Pooling layer

    # Flatten the results to feed into a Dense Layer

    # Classification layer

])
```

Once we have defined the 1st layer of the CNN Feature Learning process, we can now define the next layer for the Feature Learning process.

We will define 3 more Cov2D layers followed by MaxPooling2D layers:

- In the **2nd Conv2D layer**, we will keep the number of filters 64 with each of 3x3 size and activation function as 'relu'. Note that we need to provide the input shape only in the first layer.
- In the **3rd & 4th Conv2D layers**, we can increase the number of filters 128 with each of 3x3 size and activation function as 'relu'. Increasing the number filters in inner layers can help us extract features with more accuracy.

```
model = tf.keras.models.Sequential([

    # 1st Convolution & Pooling layer
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', input_shape=(180, 180, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),

    # 2nd Convolution & Pooling layer
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    # 3rd Convolution & Pooling layer
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    # 4th Convolution & Pooling layer
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    # Flatten the results to feed into a Dense Layer

    # Classification layer

])
```

Once we have the output from all the layers of the Feature Learning, we need first convert the output array returned

from the Feature Learning Layers to 1D array and remove extra node in the CNN the using:

Flatten Layer: This layer converts the output array returned from the Feature Learning Layers to a 1D array.

For this, we will use **Flatten Class**.

Dropout Layer: Dropout is a technique where a few of the output from the previous layer (randomly selected for dropout) is ignored before passing it to the next layer.

For this, we will use **Dropout Class** with a decimal value between 0 and 1 as the parameter, which denotes the percentage of the image data to be dropped. For example, 0.5 which is 50% (0.5×100).

Teacher Activity 6

Reference: [Reshaping layers : Flatten layer](#)

Teacher Activity 7

Reference: [Regularization layers : Dropout layer](#)

Then as the final layer we can use **Dense Layer Class**, for this we will need 2 parameters:

- **units:** The number of filter units.
- **activation:** The activation method.

Dense Class (with ReLU activation): In this, we will use 512 units and 'relu' activation.

Dense Class (with Softmax activation): In this, we will use 2 units and 'softmax' activation:

- The last layer of the CNN must have filters units equal to the number of the class we have. Since we are training our model for 2 classes, we will use 2 units in this layer.

***Note:** Make sure the number of filter units matches the number of classes chosen for the classification training process.*

- The **softmax** function produces values in the range of 0–1 for each class therefore it is used as the final layer in classification models.

***Note 1:** Generally it is recommended to use **sigmoid** function for **2 classes**(called binary class classification) and **softmax** function for **more than 2 classes**(multi-class classification), but both functions give similar results.*

***Note 2:** Students might get projects based on 2 classes or more than 2 class classification in after class projects.*

Teacher Activity 8

Reference: Core Layer : Dense layer

Teacher Activity 9

Reference: Activation Layer : ReLU layer

Teacher Activity 10

Reference: Activation Layer : Softmax layer

```
model = tf.keras.models.Sequential([

    # 1st Convolution & Pooling layer
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', input_shape=(180, 180, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),

    # 2nd Convolution & Pooling layer
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    # 3rd Convolution & Pooling layer
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    # 4th Convolution & Pooling layer
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    # Flatten the results to feed into a Dense Layer
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.5),

    # Classification Layer
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(2, activation='softmax')

])
```

Now that we have defined the CNN model, we can check the output shape of the image array returned after each layer using the **summary()** method.

We can see the output of the 4 convolutional layers (conv2d, conv2d_1, conv2d_2, conv2d_3) with max pooling layers(max_pooling2d, max_pooling2d_1, max_pooling2d_2, max_pooling2d_3) reducing dimension of the output image from the convolutional layer to half. The first layer conv2d(None, 178, 178, 64) becomes (None, 89, 89, 64) after max_pooling2d.

```
model.summary()
```

```
Model: "sequential"
```

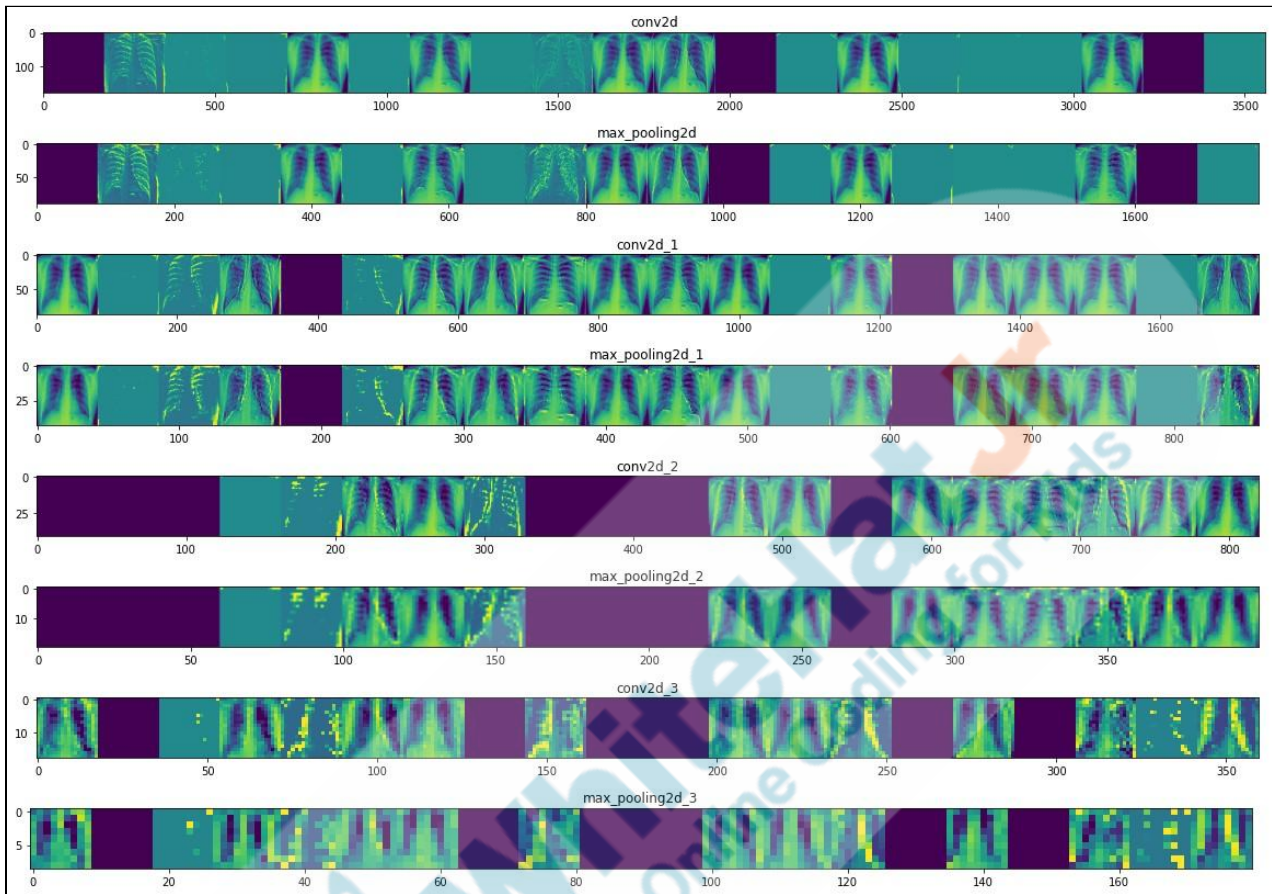
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 178, 178, 64)	1792
max_pooling2d (MaxPooling2D)	(None, 89, 89, 64)	0
conv2d_1 (Conv2D)	(None, 87, 87, 64)	36928
max_pooling2d_1 (MaxPooling2D)	(None, 43, 43, 64)	0
conv2d_2 (Conv2D)	(None, 41, 41, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 20, 20, 128)	0
conv2d_3 (Conv2D)	(None, 18, 18, 128)	147584
max_pooling2d_3 (MaxPooling2D)	(None, 9, 9, 128)	0
flatten (Flatten)	(None, 10368)	0
dropout (Dropout)	(None, 10368)	0
dense (Dense)	(None, 512)	5308928
dense_1 (Dense)	(None, 2)	1026
Total params: 5,570,114		
Trainable params: 5,570,114		
Non-trainable params: 0		

Open [Teacher Activity 13](#) to explain the visuals which shows how the features are extracted layer after layer in 4 convolutional layers (conv2d, conv2d_1, conv2d_2, conv2d_3) with max pooling layers(max_pooling2d, max_pooling2d_1, max_pooling2d_2, max_pooling2d_3)

Note: This visual is created only for explanation purposes.

- At the start of a convolutional network, the filter(feature detector/kernel) detects simple patterns, like horizontal lines, vertical lines, and corners, simple shapes. You can see there more X-Ray images with only lines in layer 1.
- In later layers of the network, filters(feature detector/kernel) are complex that detect shapes, objects, and other complex structures, which is done by using the previously generated feature and their detected simple features to build more complex ones. You can see more X-Ray images with more features after layer 1. At the last layer the CNN has extracted features to send it for classification layers.

Note: As we go deeper in the layers, the feature becomes increasingly complex, hence less visually interpretable. They begin to encode higher-level concepts such as single borders, corners and angles. Higher presentations carry increasingly less information about the visual contents of the image, and increasingly more information related to the class of the image, this why the outputs of the Dense Layer will not be shown for visual explanation.



That was interesting! We learned to define a CNN model.
Now you define a CNN model too.

Are you excited?

ESR: Yes.


Teacher Stops Screen Share

Teacher Starts Slideshow

Slide 18 to 19

Refer to speaker notes and follow the instructions on each slide.

We have one more class challenge for you.

Can you solve it? Let's try. I will guide you through it.	
<div style="text-align: center;">  Teacher Ends Slideshow </div>	
STUDENT-LED ACTIVITY - 10 mins	
<ul style="list-style-type: none"> • Ask the student to press the ESC key to come back to the panel. • Guide the student to start Screen Share. • The teacher gets into Fullscreen. 	
<div style="text-align: center;"> ACTIVITY </div> <ul style="list-style-type: none"> • Define a Convolutional Neural Network To Class Chest X-Ray Image with Pneumothorax disease. 	
Teacher Action	Student Action
<p><i>Guide the student to open the colab notebook and make a copy for the boilerplate code using Student Activity 1.</i></p> <p>Run all the cells to load the dataset and update the code cells output.</p>	
<p><i>Guide the student to define the Convolutional Neural Network using Keras:</i></p> <p>Note: Follow the teacher's activities to explain the code.</p> <ol style="list-style-type: none"> 1. Feature Learning Layers: <ol style="list-style-type: none"> a. Conv2D with ReLU Activation b. MaxPooling2D 2. Classification: <ol style="list-style-type: none"> a. Flatten the output from Feature Learning Layer b. Dropout Layer c. Dense Layer with ReLU Activation 	

d. Dense Layer with Softmax Activation

```
model = tf.keras.models.Sequential([
    # 1st Convolution & Pooling layer
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', input_shape=(180, 180, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),

    # 2nd Convolution & Pooling layer
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    # 3rd Convolution & Pooling layer
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    # 4th Convolution & Pooling layer
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    # Flatten the results to feed into a Dense Layer
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.5),

    # Classification Layer
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(2, activation='softmax')
])
```

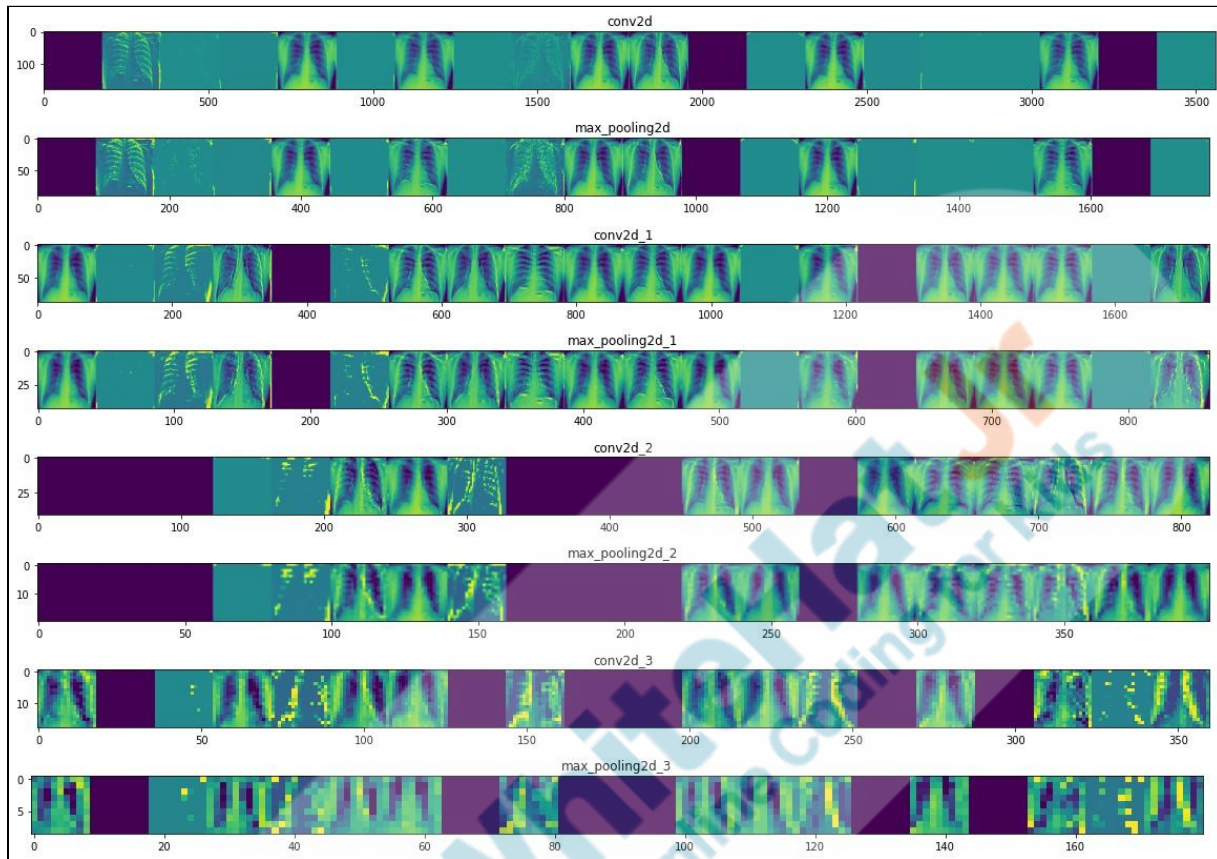
Guide the student to check the summary of CNN model:

```
model.summary()
```

```
model.summary()
```

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 178, 178, 64)	1792
max_pooling2d (MaxPooling2D)	(None, 89, 89, 64)	0
conv2d_1 (Conv2D)	(None, 87, 87, 64)	36928
max_pooling2d_1 (MaxPooling2D)	(None, 43, 43, 64)	0
conv2d_2 (Conv2D)	(None, 41, 41, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 20, 20, 128)	0
conv2d_3 (Conv2D)	(None, 18, 18, 128)	147584
max_pooling2d_3 (MaxPooling2D)	(None, 9, 9, 128)	0
flatten (Flatten)	(None, 10368)	0
dropout (Dropout)	(None, 10368)	0
dense (Dense)	(None, 512)	5308928
dense_1 (Dense)	(None, 2)	1026
Total params: 5,570,114		
Trainable params: 5,570,114		
Non-trainable params: 0		



With this, our Convolutional Neural Network is ready to be trained for image classification.

You did amazing work today!

Teacher Guides Student to Stop Screen Share

WRAP-UP SESSION - 05 mins

Teacher Starts Slideshow
Slide 20 to 25



Activity Details:

© 2021 - WhiteHat Education Technology Private Limited.

Note: This document is the original copyright of WhiteHat Education Technology Private Limited.

Please don't share, download or copy this file without permission.

Following are the WRAP-UP session deliverables:

- Appreciate the student.
- Revise the current class activities.
- Discuss the quizzes.

WRAP-UP QUIZ

Click on In-Class Quiz



Continue WRAP-UP Session
 Slide 26 to 31

Activity Details:

Following are the session deliverables:

- Explain the facts and trivia
- Next class challenge
- Project for the day
- Additional Activity (Optional)

FEEDBACK

- **Appreciate the student for his/her efforts in the class.**
- **Ask the student to make notes for the reflection journal along with the code they wrote in today's class.**

Teacher Action


You get Hats off for your excellent work!

Student Action

Make sure you have given at least 2 Hats Off during the class for:

Creatively Solved Activities  +10

Great Question  +10

		<div>Strong Concentration </div>
<div>PROJECT OVERVIEW DISCUSSION</div> <div>Refer the document below in Activity Links Sections</div>		
<div>Teacher Clicks</div>		<div>✕ End Class</div>
<div>ADDITIONAL ACTIVITIES</div>		
<div><i>Encourage the student to understand the mathematical computations of the Conv2D layer, which happens in the backend using Conv2D class.</i></div> <div>Teacher Activity 11</div> <div>Reference: Conv2D Computations Example</div> <div><i>Note that this example is shown with only 1 filter.</i></div> <div>We need:</div> <div><ul style="list-style-type: none">• An input image array• A filter(feature detector/kernel) array. This array is randomly chosen at the time of computation.</div> <div>The output will be the new image array.</div>		<div><i>The student uses the markdown editor to write her/his reflections in the reflective journal.</i></div>

Conv2D Feature Extraction Computation Example

Input Image(5x5)

0	0	0	0	0
0	1	1	1	0
0	1	0	0	0
0	1	1	1	0
0	0	0	0	0

Filter1(3x3)

1	0	1
0	0	1
1	0	1

Conv2D Layer Output

?	?	?
?	?	?
?	?	?

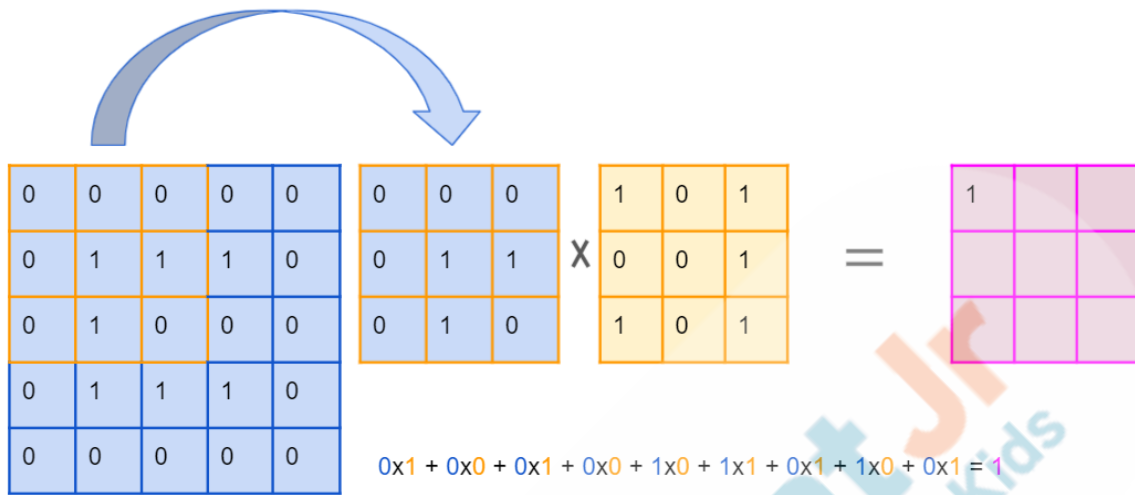
Note that the computation process can be explained using [Teacher Activity 11](#)

Since the **input image** array and the **filter**(feature detector/kernel) array are 2D, they are also called **matrices**.

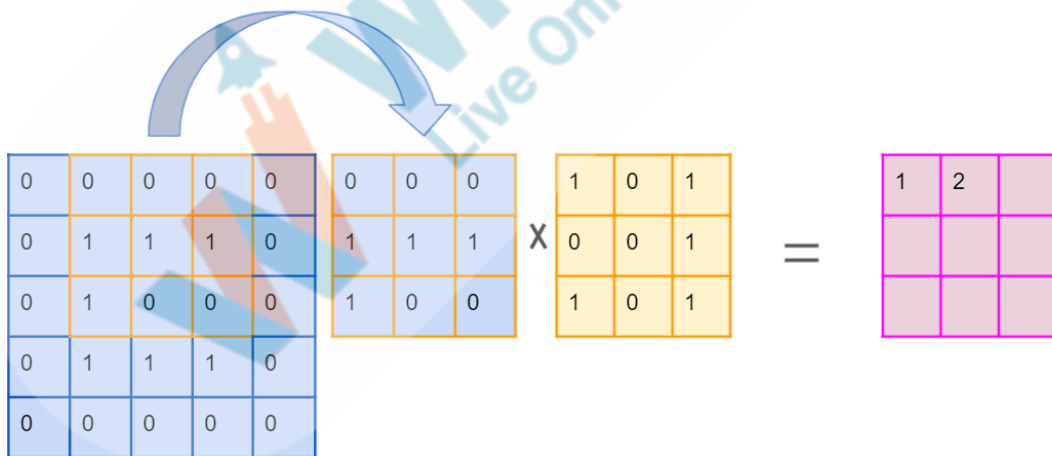
A portion of the input image array matrix, called a sub-array(size same as the size of the filter) is taken, starting from the top left.

This sub-array is multiplied with the filter array. We can multiply one array matrix to another, by multiplying 1st element to 1st element of both the arrays(2nd element to 2nd element of both arrays and so on).

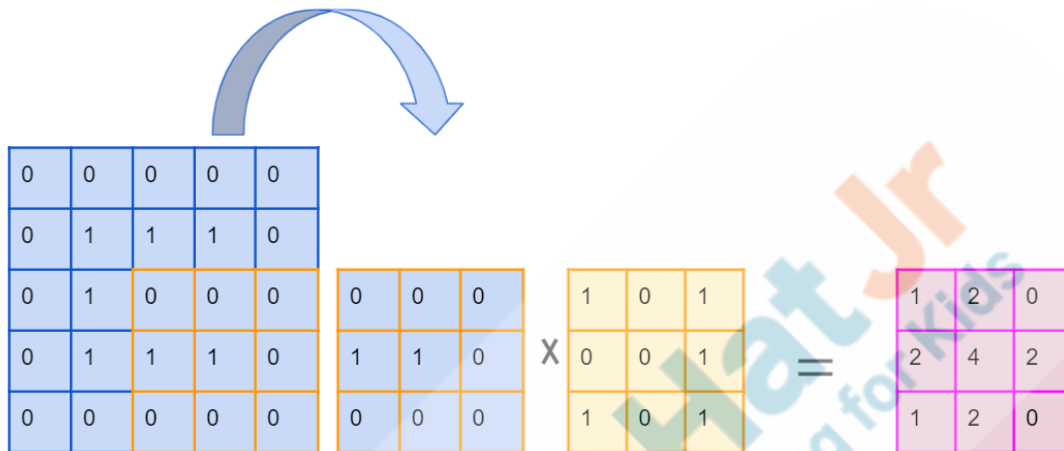
After multiplying, the result is added, which gives the value of the 1st element of the new image array.



Then we shift towards right by one column, repeat the above steps to get the value of the 2nd element of the new array.



Once the whole row is finished, we shift downwards by one row, repeat the above steps to get the value of all elements of the new array one by one.



This is the final output after convolution with 1 filter.

Conv2D Feature Extraction Computation Example

Input Image(5x5)

0	0	0	0	0
0	1	1	1	0
0	1	0	0	0
0	1	1	1	0
0	0	0	0	0

Filter1(3x3)

1	0	1
0	0	1
1	0	1

Conv2D Layer Output(3x3)

1	2	0
2	4	2
1	2	0

<p>The whole process is repeated with different filters, to get different output, which all together is the output of the 1st Conv2D layer.</p> <p>These outputs from the 1st Conv2D layer are given to the 2nd Conv2D layer and convolutions are performed.</p> <p>This repeated for all the layers of the CNN model.</p>	
--	--

ACTIVITY LINKS		
Activity Name	Description	Link
Teacher Activity 1	Feature Learning Conv2D Visualization	https://s3-whjr-curriculum-uploads.whjr.online/3917c089-1d8f-4f32-b5c4-401b6abe8d47.gif
Teacher Activity 2	Teacher Boilerplate Code	https://colab.research.google.com/drive/1SV9oF-xxiccW0A3cInZM4JOluftmw1rW?usp=sharing
Teacher Activity 3	Sequential Class API Reference	Model Class: The Sequential class
Teacher Activity 4	Conv2D Layer API Reference	Convolution Layer: Conv2D layer
Teacher Activity 5	MaxPooling2D API Reference	Pooling Layer : MaxPooling2D layer
Teacher Activity 6	Flatten Layer API	Reshaping layers : Flatten layer
Teacher Activity 7	Dropout Layer API	Regularization layers : Dropout layer
Teacher Activity 8	Dense Layer API Reference	Core Layer : Dense layer
Teacher Activity 9	Activation ReLU API Reference	Activation Layer : ReLU layer

Teacher Activity 10	Activation Softmax API Reference	Activation Layer : Softmax layer
Teacher Activity 11	Feature Learning Conv2D Computation	Conv2D Computations Example
Teacher Activity 12	Reference Code	https://colab.research.google.com/drive/1sMtqEuDduPTLix3szxDjTlfxIsXerWgO?usp=sharing
Teacher Activity 13	Convolutional Layer Intermediate Output Visualization	https://s3-whjr-curriculum-uploads.whjr.online/b941c8bd-c137-449f-ae8f-7233735a7845.jpg
Student Activity 1	Student Boilerplate Code	https://colab.research.google.com/drive/1SV9oF-xxiccW0A3cInZM4JOluftmw1rW?usp=sharing
Teacher Reference 1	Project Document	https://s3-whjr-curriculum-uploads.whjr.online/a8ee7298-438c-4229-851c-84692d9f7504.pdf
Teacher Reference 2	Project Solution	https://colab.research.google.com/drive/1JK2soeyrbR3ur6_rTKNMLvtv2E9uYh9T?usp=sharing
Teacher Reference 3	Visual Aid Link	https://s3-whjr-curriculum-uploads.whjr.online/03766132-552d-444f-9537-16da5d8d89d9.html
Teacher Reference 4	In Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/89f6f53b-cad6-435e-b666-aea1bfc83504.pdf