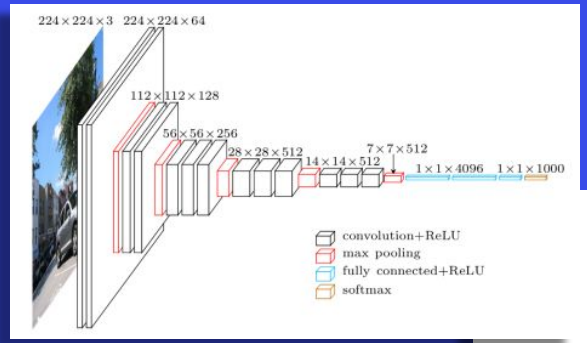
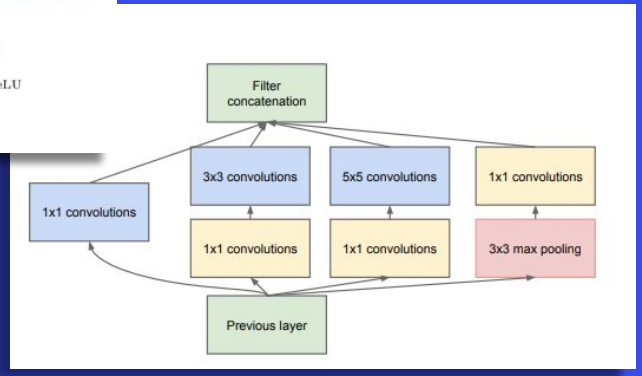
Introduction to Image Classification using Convolutional Neural Networks (CNNs)

“Convolutional Neural Networks (CNNs) have revolutionized **image classification**, enabling machines to **identify** and **categorize** visual data with unprecedented accuracy. By learning **features** directly from images, CNNs power applications in fields from **healthcare** to **autonomous driving**. This presentation introduces the **fundamentals of CNNs** and their role in transforming how we interact with technology.





Yaqoob Abdulhakeem Omotolani

Brief Agenda

- **Overview of CNNs**
- **Image classification use case**
- **CNN architecture and key concepts (Convolution, Pooling, Activation Functions, Fully Connected Layers)**
- **Lab Preview**

What is A Convolutional Neural Network (CNN)?

CNNs are a class of deep neural networks commonly used for **analyzing** visual data. CNNs excel at **detecting patterns** and visual features in images due to their architecture, making them a popular choice for image-related tasks.

Key Use Cases

Multilabel Classification

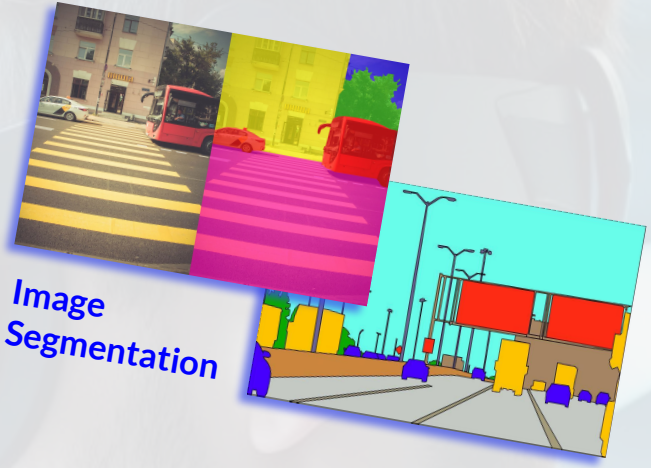
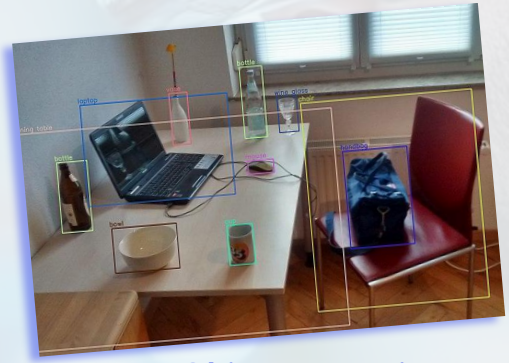


Image Segmentation

Binary Classification

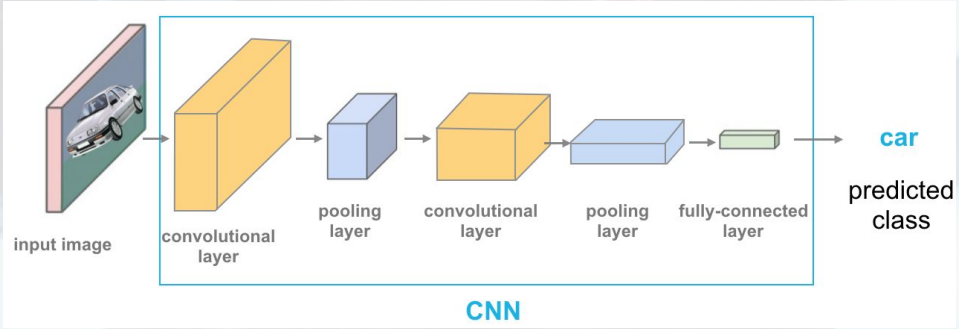


Object Detection

Face Detection

Components in a CNN Architecture

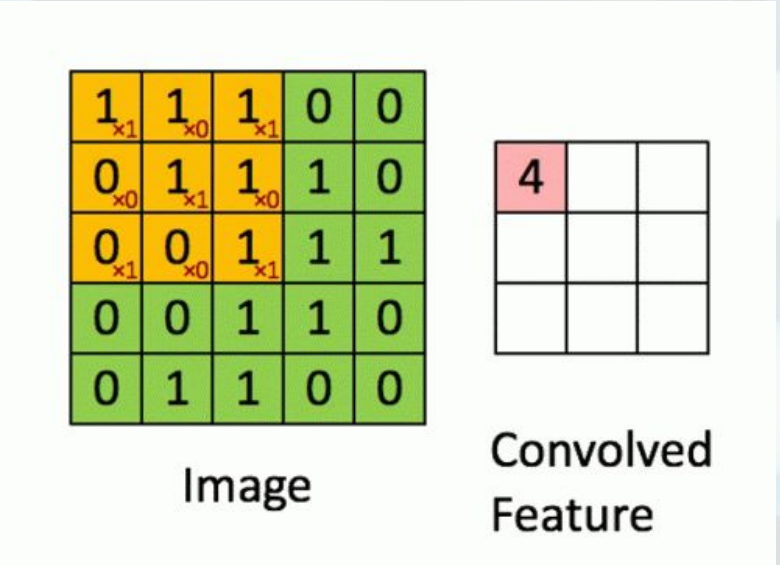
- Input Layer
- Convolutional Layer
- Max Pool (Pooling) Layer
- Flatten Layer
- Dense Layer (Fully Connected Layer) or Activation Layer



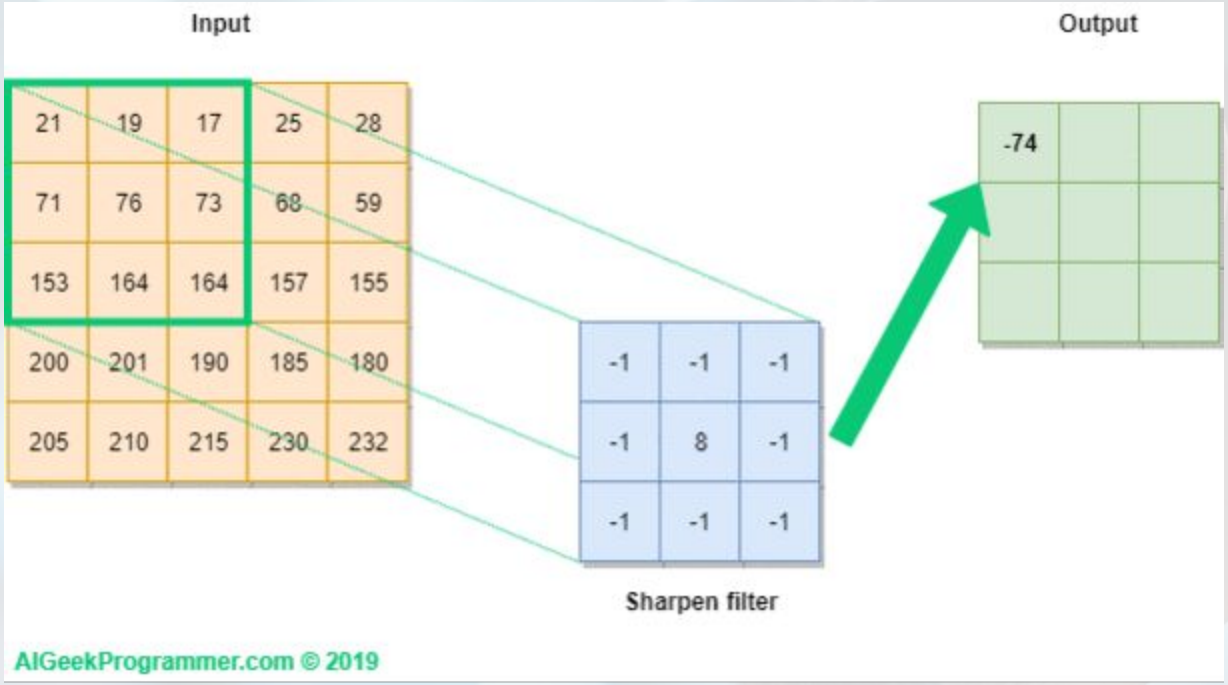
(source: Convolutional Neural Networks — Cezanne Camacho — Machine and deep learning educator.)

Convolutional Layer

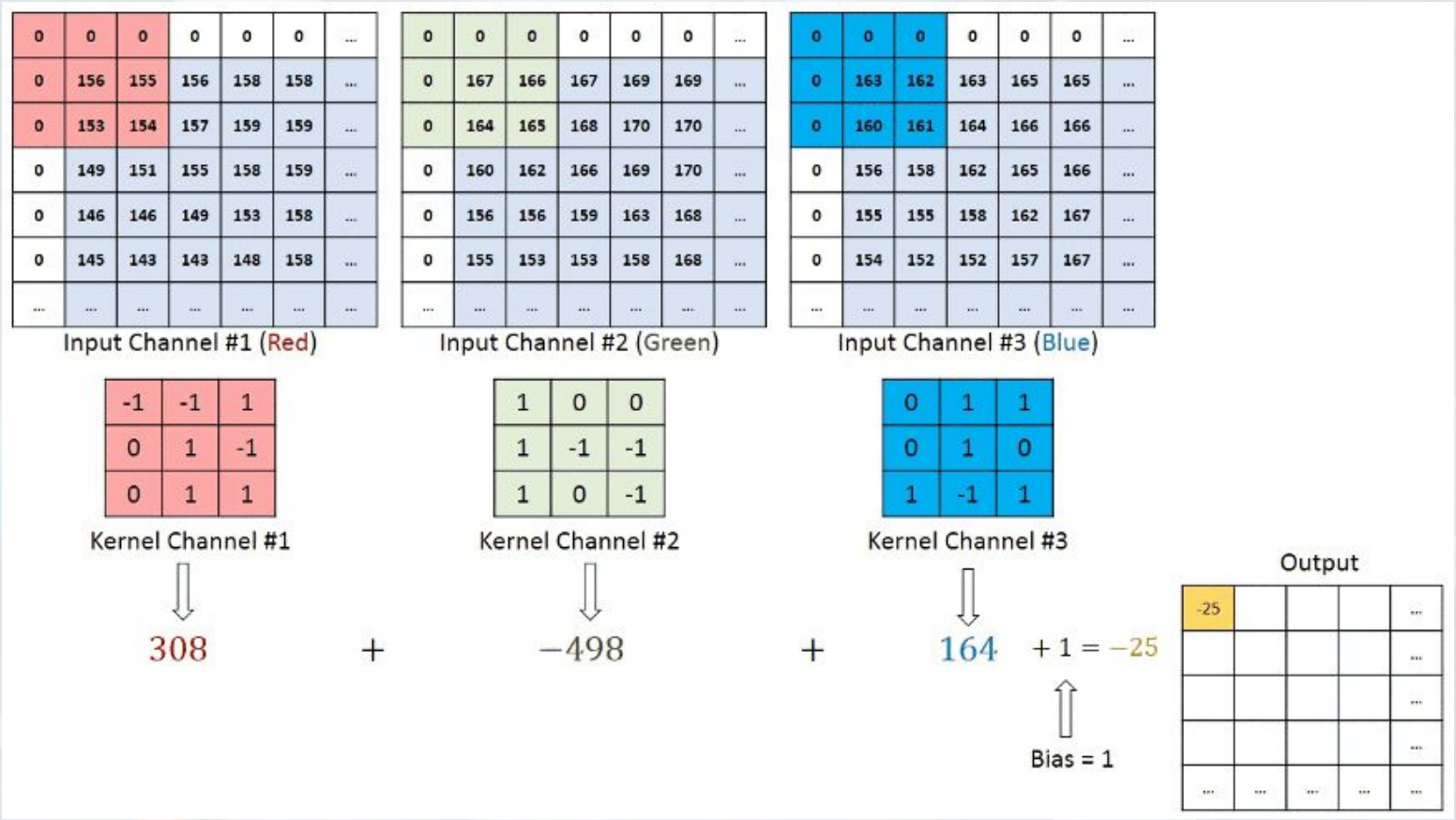
- **What it is:** The heart of a CNN, designed to detect features in an image.
- **How it works:** Small filters (kernels) move across the image, identifying patterns like edges, textures, and shapes.
- **Visual:** Show an example of a filter scanning an image to highlight these features.



Convolutional Layer (Example 1)



Convolutional Layer (Example 2)



Convolutional Layer (Hyperparameters)

1. Padding:

Padding adds a border around the image to make sure edges aren't ignored when the filter (kernel) scans the image. Zero-padding (adding zeros) is the most common method because it's simple and helps maintain the same image size for deeper networks.

2. Kernel Size:

Kernel size is the size of the window that slides over the image. Smaller kernels capture more details and allow the network to learn complex patterns, while larger kernels work faster but may miss fine details. Smaller sizes are usually better for image classification.

3. Stride:

Stride is how far the kernel moves with each step. A small stride (like 1) captures more details by analyzing every part of the image, while a larger stride skips areas, making the process faster but less detailed.

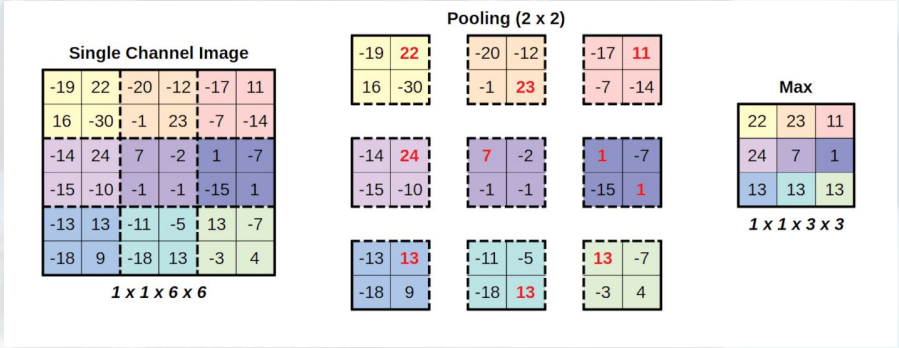
Questions,
Comments or Contributions?

Activity 1

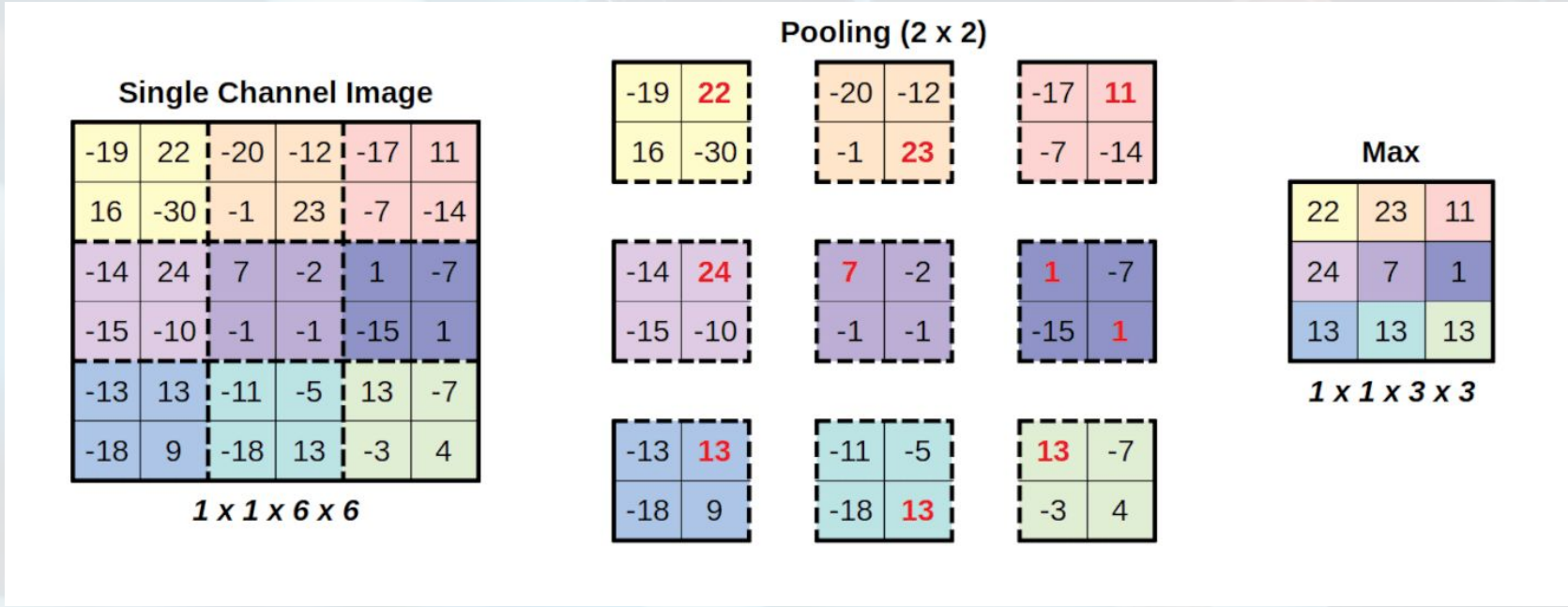
- **Deep Lizard** (In Class)

Pooling Layer

- What it is: A layer that reduces the size of the feature maps while keeping important information.
- How it works: It simplifies the data by taking the most important parts (like the highest values) from small regions of the image.
- Visual: Show an example of pooling reducing an image's size while keeping key features intact.



Max Pooling Layer



Pooling Layer (Hyperparameters)

1. Padding: Similar to convolutional layers, padding in pooling layers determines whether the size of the input will be reduced or maintained after pooling. Adding padding ensures that the pooling window can cover all parts of the image, even at the edges. However, it's less common in pooling layers compared to convolutional layers. The most typical approach is no padding.

2. Kernel Size: Kernel size, in the context of pooling layers, refers to the size of the window used to pool the values. The most common kernel size is 2x2, meaning the window reduces the image size by half. Larger kernel sizes result in more aggressive downsampling, which reduces the resolution faster, while smaller sizes retain more spatial information.

3. Stride: Stride in a pooling layer determines how far the pooling window moves across the input image. A stride of 2 is most commonly used, meaning the window skips every other element, reducing the spatial dimensions of the input. A larger stride speeds up the computation but decreases the output resolution more aggressively, while a smaller stride retains more detailed spatial information.

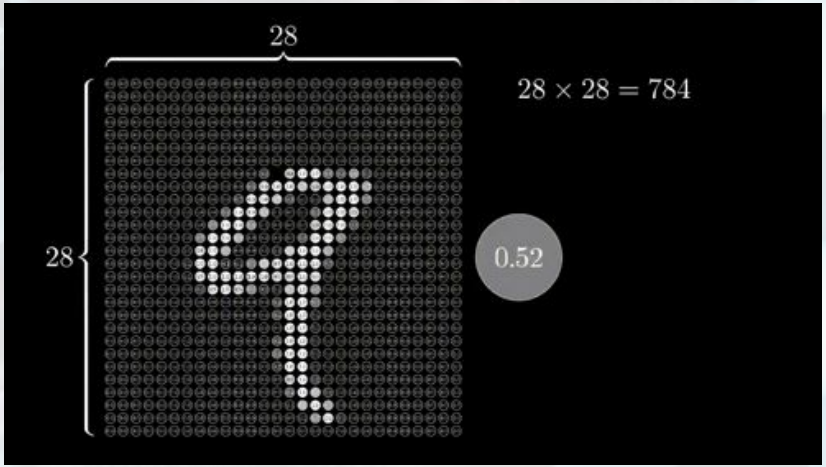
Questions,
Comments or Contributions?

Activity 2

- **Deep Lizard** (In Class)

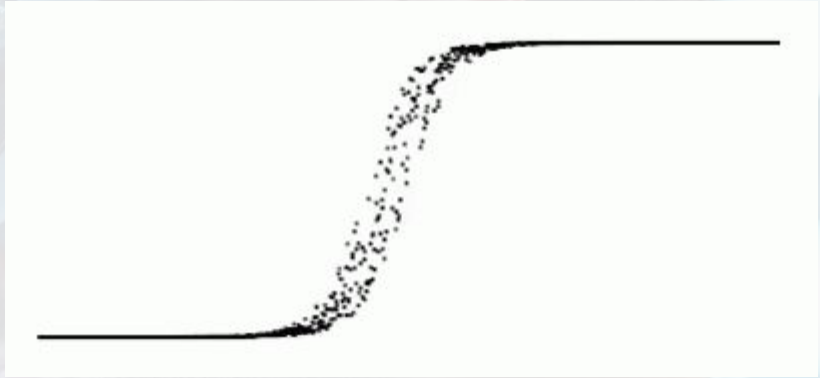
Flatten Layer

- **What it is:** A layer that transforms the 2D feature maps into a 1D vector.
- **How it works:** It takes the output from the convolutional and pooling layers and "flattens" it into a single long list of numbers, preparing it for the fully connected layers.



Activation Layer

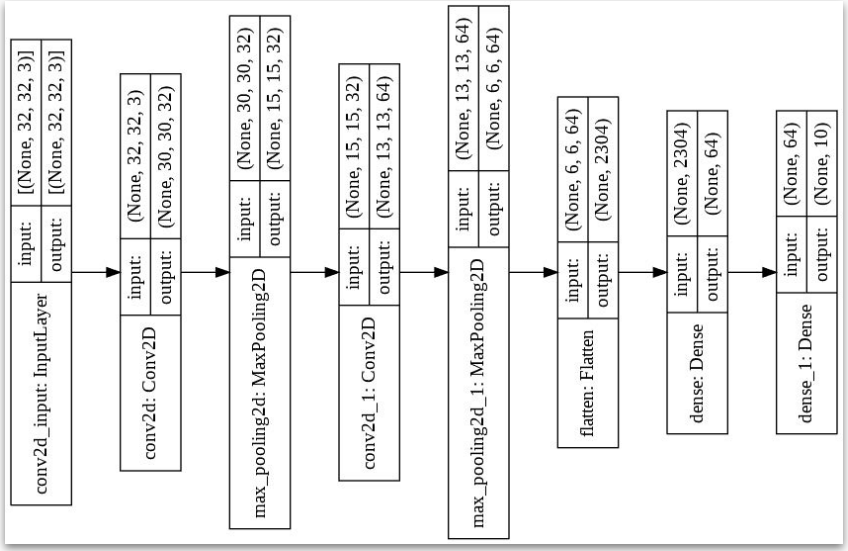
- What it is: The final layer that makes the prediction based on the learned features.
- How it works: It applies an activation function (like sigmoid for 2 and softmax for 3 classes) to generate probabilities for each class, determining the image's category.



```
model = tf.keras.Sequential(
[
tf.keras.layers.Conv2D(32, (3,3), padding='same',
activation="relu",input_shape=(32, 32, 3)),
tf.keras.layers.MaxPooling2D((2, 2), strides=2),
tf.keras.layers.Conv2D(64, (3,3), padding='same',
activation="relu"),

tf.keras.layers.MaxPooling2D((2, 2), strides=2),

tf.keras.layers.Flatten(), tf.keras.layers.Dense(100,
activation="relu"),

tf.keras.layers.Dense(10, activation="softmax")
] )
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 2304)              0
_________________________________________________________________
dense (Dense)                (None, 64)                147520
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
=================================================================
Total params: 167,562
Trainable params: 167,562
Non-trainable params: 0
_________________________________________________________________
```

Sample CNN Code



Questions,
Comments or Contributions?

Activity 3

- **CNN Explainer (In Class)**