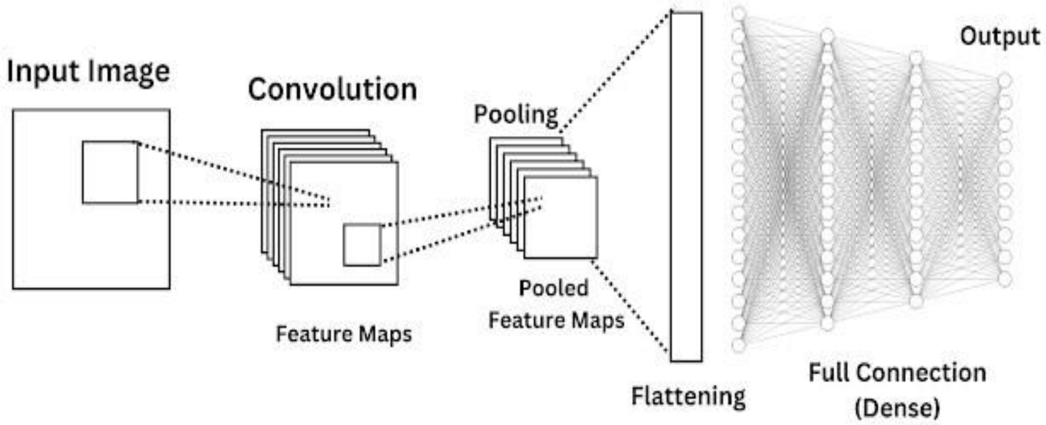
CNN for COVID-19 detection

Convolutional Neural Network?

- A Convolutional Neural Network (CNN) is a specialized type of deep neural network designed for processing and analyzing visual data, such as images and videos. It has revolutionized the field of computer vision by significantly improving the accuracy and efficiency of image recognition tasks.
- The architecture of Convolutional Neural Networks consists of several layers, Including Convolution, Activation (ReLU) Layer, Pooling, Flattening layer, and Fully connected layer.



Reference: https://www.pycodemates.com/2023/06/introduction-to-convolutional-neural-networks.html

OUTPUT-

```
Epoch 16/30
96/96 [==============================] - 40s 409ms/step - loss: 0.2360 - accuracy: 0.9016
Epoch 17/30
96/96 [==============================] - 42s 440ms/step - loss: 0.2581 - accuracy: 0.8909
Epoch 18/30
96/96 [==============================] - 36s 379ms/step - loss: 0.2333 - accuracy: 0.9033
Epoch 19/30
96/96 [==============================] - 41s 428ms/step - loss: 0.2325 - accuracy: 0.8993
Epoch 20/30
96/96 [==============================] - 38s 392ms/step - loss: 0.2085 - accuracy: 0.9143
Epoch 21/30
96/96 [==============================] - 42s 433ms/step - loss: 0.2180 - accuracy: 0.9130
Epoch 22/30
96/96 [==============================] - 45s 468ms/step - loss: 0.2004 - accuracy: 0.9215
Epoch 23/30
96/96 [==============================] - 40s 414ms/step - loss: 0.2115 - accuracy: 0.9156
Epoch 24/30
96/96 [==============================] - 37s 388ms/step - loss: 0.1928 - accuracy: 0.9208
Epoch 25/30
96/96 [==============================] - 37s 381ms/step - loss: 0.1744 - accuracy: 0.9319
Epoch 26/30
96/96 [==============================] - 43s 449ms/step - loss: 0.1864 - accuracy: 0.9231
Epoch 27/30
96/96 [==============================] - 42s 438ms/step - loss: 0.1615 - accuracy: 0.9371
Epoch 28/30
96/96 [==============================] - 39s 405ms/step - loss: 0.1752 - accuracy: 0.9316
Epoch 29/30
96/96 [==============================] - 33s 344ms/step - loss: 0.1779 - accuracy: 0.9300
Epoch 30/30
96/96 [==============================] - 32s 328ms/step - loss: 0.1669 - accuracy: 0.9407
67/67 [==============================] - 17s 258ms/step - loss: 0.1929 - accuracy: 0.9266
Test loss: 0.19287675619125366
Test accuracy: 0.9265536665916443
```

Code Explanation-

1- **Importing Libraries:**
   - The code imports necessary libraries: ImageDataGenerator from Keras for data augmentation and preprocessing, Sequential for building the model, and the required layers (Conv2D, MaxPooling2D, Flatten, Dense) for constructing the CNN.

2- **Dataset Paths and Image Dimensions:**
   - These variables hold the paths to the directories containing the training and testing images (train_data_dir and test_data_dir, respectively). The image_width and image_height variables define the dimensions to which the

input images will be resized. num_channels represents the number of color channels (3 for RGB images).

3- **Batch Size and Number of Classes:**

- The batch_size is set to 32, meaning that during training, the model will process 32 images together in each batch. The num_classes variable is set to 2, indicating that the dataset contains images from two classes. This should be updated with the correct number of classes in your specific dataset.

4- **Data Augmentation and Preprocessing:**

- Two instances of ImageDataGenerator are created: train_datagen and test_datagen. The train_datagen is used for data augmentation during training and includes transformations like rescaling, shear, zoom, and horizontal flip to augment the dataset and increase its diversity. The test_datagen is only used to rescale the test images since data augmentation is not applied during testing.

5- **Loading and Preprocessing Data:**

- The training and testing datasets are loaded and preprocessed using the flow_from_directory method of the data generators. The method reads images from the specified directories, applies the transformations defined in the corresponding ImageDataGenerator, resizes the images to the target size, and generates batches of preprocessed images along with their one-hot encoded categorical labels.
- I added the necessary steps to load and preprocess the testing dataset using the test_datagen and the flow_from_directory method for testing data. Then, after training the model, we evaluate its performance on the testing dataset using the evaluate method with the test_generator. This allows you to assess the model's generalization performance on data it has not seen during training.

6- **Building the CNN Model:**

- Input Layer:
  - The model starts with an input layer of Conv2D(32, (3, 3), activation='relu', input_shape=(image_width, image_height, num_channels)).
  - This layer is a 2D convolutional layer with 32 filters (also known as kernels or feature detectors) of size 3x3 each.
  - The activation function used is 'relu' (Rectified Linear Unit), which introduces non-linearity to the model.
  - The input_shape parameter is set to (image_width, image_height, num_channels), representing the dimensions of the input images.
- MaxPooling2D Layer:

    - After each Conv2D layer, there is a MaxPooling2D((2, 2)) layer.
    - MaxPooling2D is a pooling layer that reduces the spatial dimensions of the feature maps while retaining the most important features.
    - The (2, 2) parameter specifies the size of the pooling window, which is a 2x2 window.
    - This operation downsamples the feature maps, reducing their spatial size by half in both width and height.
- Additional Convolutional Layers:
    - The model includes four more Conv2D layers with increasing filter sizes: 64, 128, 128, and 256.
    - Each Conv2D layer applies another set of filters to capture higher-level features as the information passes through the network.
    - The activation function 'relu' is used for each of these convolutional layers.
- Flatten Layer:
    - After the last MaxPooling2D layer, there is a Flatten() layer.
    - The Flatten layer converts the 3D feature maps into a 1D vector, preparing the data for the fully connected (Dense) layers.
- Dense Layers:
    - The model continues with three Dense layers: Dense(256, activation='relu'), Dense(128, activation='relu'), and Dense(num_classes, activation='softmax').
    - Dense layers are fully connected layers, where each neuron is connected to every neuron in the previous layer.
    - The 256 and 128 refer to the number of neurons in these layers. The number of neurons in Dense layers is a hyperparameter and can be adjusted based on the complexity of the task.
    - The activation function used for the Dense layers is 'relu', except for the last Dense layer.
    - The last Dense layer has num_classes neurons, which is equal to the number of classes in the dataset (binary or multi-class).
    - The activation function used for the last Dense layer is 'softmax', which converts the output values into probabilities, allowing the model to make predictions across all classes.

7- Compiling the Model:

- The model is compiled with the Adam optimizer, categorical cross-entropy loss (since this is a multi-class classification problem), and accuracy as the metric to be monitored during training.

8- Training the Model:

- The model is trained using the fit method with the training data generator for 30 epochs. During training, the model iteratively updates its weights based on the training data to minimize the loss and improve accuracy.

9- Evaluating the Model:

- After training is complete, the model is evaluated on the testing dataset using the evaluate method. The test_generator is used to provide batches of test images and their corresponding labels. The evaluate method calculates the test loss and accuracy of the model on the unseen test data. The calculated values are then printed to the console.

Conclusion-

Overall, this code demonstrates the complete process of building, training, and evaluating a basic CNN for image classification using Keras. The data augmentation techniques used during training help improve the model's performance by increasing the diversity of the training data. The model is compiled with an optimizer, loss function, and metrics for evaluation, and then trained on the training dataset for a specified number of epochs. Finally, the trained model is evaluated on the test dataset to assess its performance on unseen data.

Aviral Asthana
Vellore Institute of Technology- Bhopal
https://www.linkedin.com/in/aviral-asthana-4393b4234/