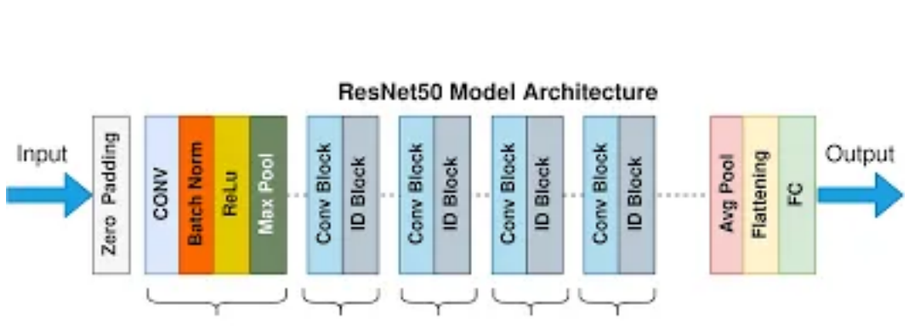
Medium Search Write Sign up Sign in

Binary Image Classification using Resnet50

Habib Ur Rehman Follow 5 min read · Feb 5, 2024

22 1



> *Before we implement the resnet50 for image classification, Lets get familiar with What is a Neural Network and What is Convolution in a Neural Network*

What is a Neural Network

A Neural Network is a Computational Network inspired by the structure and functioning of the human brain's interconnected network of neurons. It's a fundamental component of machine learning and artificial intelligence, used for tasks such as pattern recognition, classification, regression, and more. Let's break down the key components and concepts of neural networks in detail:

- **Neurons**: Computational units that receive inputs, apply weights and biases, and produce outputs using activation functions.
- **Layers:** Neurons are organized into layers: input, hidden, and output. Hidden layers learn patterns in data.
- **Activation Functions:** Introduce non-linearities, such as sigmoid, tanh, ReLU, to enable complex modeling.
- **Forward Propagation**: Input data flows through the network, passing through layers and activation functions to produce outputs.
- **Loss Function**: Measures the difference between predicted and actual outputs.
- **Backpropagation:** Optimization algorithm adjusts weights and biases based on loss, propagated backwards through the network.
- **Training:** Iteratively exposes the network to labeled data, adjusting parameters to minimize loss.
- **Applications:** Used in various fields like computer vision, natural language processing, and speech recognition for pattern recognition and prediction.

What is Convolution?

In statistics, convolution refers to a mathematical operation that combines two functions to produce a third function, representing the distribution of the sum of random variables.

In Deep Learning, convolution refers to a fundamental operation used in convolutional neural networks (CNNs) for feature extraction. It involves applying a filter (also known as a kernel) to an input image to produce a feature map.

- In the context of image processing, a filter is a small matrix of weights that slides over the input image, computing the dot product at each position.
- The output at each position in the feature map represents the activation of a particular feature detected by the filter.

The Resnet50

ResNet-50 is a deep convolutional neural network architecture introduced by Microsoft Research in 2015. It is known for its depth and its use of skip connections, which address the vanishing gradient problem in very deep networks.

Skip Connections

- Skip connections, also known as residual connections, enable the network to learn residual mappings instead of directly learning the desired underlying mapping.
- The idea is to add shortcut connections that skip one or more layers, allowing the network to learn the residual (difference) between the input and the output of those layers.
- This helps in mitigating the vanishing gradient problem and facilitates the training of very deep networks.

Residual Block

- The basic building block of ResNet architectures is the residual block.
- It consists of multiple convolutional layers with skip connections.
- The key innovation is the introduction of a "shortcut" or "skip" connection that adds the original input of a block to its output before applying the activation function.
- Mathematically, the output of a residual block is $F(x)+x$, where $F(x)$ represents the output of the block's convolutional layers, and x represents the input to the block.

Identity Function

- In a residual block, if the input and output dimensions are the same, the shortcut connection simply performs an identity mapping.
- The identity function allows the network to learn only the residuals, making it easier to optimize and train deeper networks.
- It also enables the network to learn when to apply transformations to the input features and when to keep them unchanged.

Lets Use aResnet50 Model for Image Classification

We will classify Image as either cat or as dog so it will be binary classification.

import the necessary libraries

```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import Flatten, Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
import pathlib
from google.colab import drive
import os
```

Load the Dataset from Google Drive

```
drive.mount('/content/drive')

# Path to dataset
train_directory = '/content/drive/MyDrive/archive/train'
test_directory = '/content/drive/MyDrive/archive/test'
```

Lets see how many instances of each class for training are there

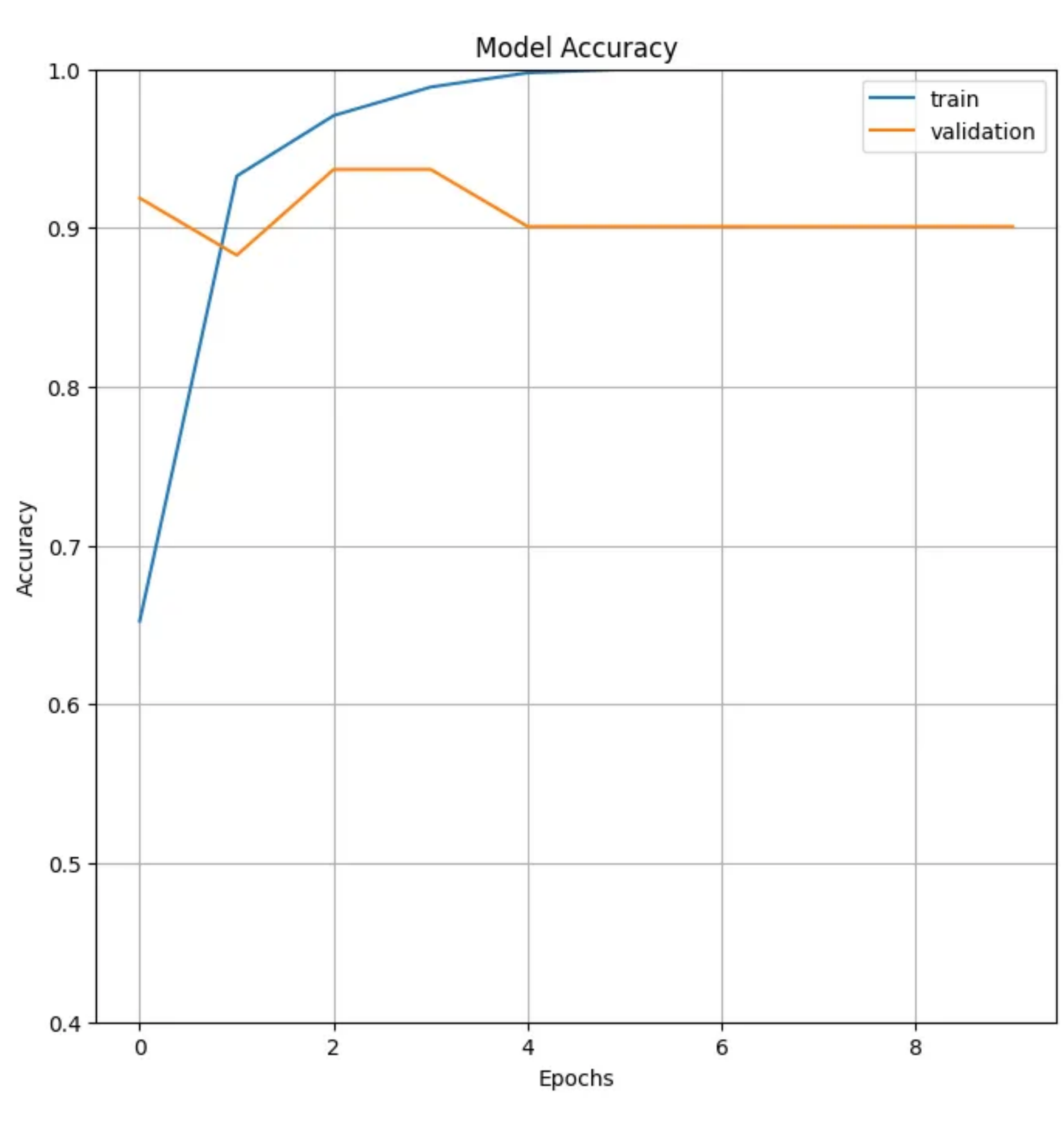
```
categories = os.listdir(train_directory)
# Number of images for each disease
nums = {}
for lebal in categories:
    nums[lebal] = len(os.listdir(train_dir + '/' + lebal))

# converting the nums dictionary to pandas dataframe passing index as plant name

img_per_class = pd.DataFrame(nums.values(), index=nums.keys(), columns=["no. of
img_per_class
```

	no. of images
cats	279
dogs	278

Training the model, We will not train the previously learned weights which the resnet50 model had learned imagenet dataset but we will be adding our own fully connected input and output layers

```
# Define image dimensions and batch size
img_height, img_width = 180, 180
batch_size = 32

# Load the training dataset
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    train_directory,
    validation_split=0.2,
    subset="training",
    seed=123,
    image_size=(img_height, img_width),
    batch_size=batch_size,
    label_mode='binary'  # binary classification for cats and dogs
)

# Load the validation dataset
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    train_directory,
    validation_split=0.2,
    subset="validation",
    seed=123,
    image_size=(img_height, img_width),
    batch_size=batch_size,
    label_mode='binary'  # binary classification for cats and dogs
)

# Use ResNet50 as base model
base_model = tf.keras.applications.ResNet50(
    include_top=False,
    weights='imagenet',
    input_shape=(img_height, img_width, 3)
)

# Freeze the layers of the base model
base_model.trainable = False

# Create a new model
model = Sequential([
    base_model,
    Flatten(),
    Dense(384, activation='relu'),
    Dense(2, activation='softmax')  # binary classification, so 1 neuron with si
])

# Compile the model
model.compile(
    optimizer=Adam(),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

# Train the model
history = model.fit(train_ds, validation_data=val_ds, epochs=10)

# Plot training history
plt.figure(figsize=(8, 8))
epochs_range = range(10)
plt.plot(epochs_range, history.history['accuracy'], label="Training Accuracy")
plt.plot(epochs_range, history.history['val_accuracy'], label="Validation Accura
plt.axis(ymin=0.4, ymax=1)
plt.grid()
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['train', 'validation'])
plt.show()

# Save the model
model.save("cats_vs_dogs_model.keras")
```

Visualizing Train vs Validation accuracy



As from the graph we can see that we have a validation accuracy of about 90%

Making Prediction using a cat or a dog image

```
# Make predictions on a sample image
image_path = "images.jpg"
img = tf.keras.preprocessing.image.load_img(image_path, target_size=(img_height,
img_array = tf.keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)  # Create a batch

predictions = model.predict(img_array)

# Interpret the predictions
class_names = ["cat", "dog"]
predicted_class_index = np.argmax(predictions)
predicted_class = class_names[predicted_class_index]

print("Predicted class:", predicted_class)
```

lets give the model an image of a cat

here is the model output:

1/1 [==============================] — 0s 44ms/step Predicted class: cat

Resnet50 Cnn Model Image Classification Neural Networks

22 1

Written by Habib Ur Rehman
1 Follower · 1 Following
Follow

Responses (1)

Write a response

What are your thoughts?

Jorgecardete
Feb 9, 2024

Great content!

1 Reply

Recommended from Medium

Jo Wang
CNN Basics: Convolutional Layers and Pooling Layer | How to...
Key Ingredient 1: Convolutional Layers
Oct 29, 2024 31

Piyush Kashyap
Transfer Learning in PyTorch: Fine-Tuning Pretrained Models for...
In recent years, deep learning has revolutionized the way we approach comple...
Dec 4, 2024

In Coding Beauty by Tari Ibaba
This new IDE from Google is an absolute game changer
This new IDE from Google is seriously revolutionary.
Mar 11 4.8K 278

In Biased-Algorithms by Amit Yadav
A Practical Guide to Implementing Early Stopping in PyTorch for...
I understand that learning data science can be really challenging...
Oct 28, 2024 20

In Lexiconia by Mohamad Mahmood
Multi-Label Classification Dataset Repository
This repository is a collection of multi-label classification datasets sourced from various...
Mar 23

Zeeshan Saghir
How Much RAM Is Enough to Run LLMs in 2025: 8GB, 16GB, or More?
8GB of RAM might get you by in 2025, but if you're serious about LLMs, 16GB or more is...
Mar 20 50

See more recommendations

Help Status About Careers Press Blog Privacy Rules Terms Text to speech