- Deep Residual Learning for Image Recognition 2015 Paper

Residual Block

- In traditional neural networks, each layer feeds into next layer

In a network with `residual blocks`

- each layer feeds into next layer and directly into layers about 2–3 hops away this is all about residual concep

Understanding intuition behind Residual Block:

1. Why Residual Block was required in first place?
2. Why it is so important?
3. How similar it looks to some other state-of-the-art architectures?

Lets answer above questions

There are many interpretation of `Why residual blocks are awesome` and `How & Why they are one of key ideas that can make a neural network show state-of-the-art performances on a wide range of tasks`

We know

- neural networks are Universal Function Approximators
- accuracy of NN increases with increasing number of layers

There is a limit to number of layers added that results in an accuracy improvement If neural networks were universal function approximators, then they should have been able to learn any simple or complex function Because of some problems like

- `Vanishing Gradients`
- `Curse of Dimensionality` If we have sufficiently deep networks, it may not be able to learn simple functions like an identity function

1. If we keep increasing number of layers, we will see that `accuracy will saturate at one point and eventually degrade`
   - This is usually not caused due to `overfitting`
2. It might seem that `shallower networks are learning better than their deeper counterparts`
   - In practice this is what was observed and is popularly known as **Degradation Problem**

In **`Degradation Problem`** we know that shallower networks perform better than deeper counterparts with few more layers added to them So, why not skip these extra layers and at least match accuracy of shallow sub-networks

`How possible one can skip layers from training?`

Training of few layers can be skipped using `Skip Connections` or *`Residual Connections`*

This is what you will see in above image

- We can directly learn an `identity function` by relying on `skip connections only`
- This is exact reason why `skip connections are also called identity shortcut connections`
  - `skip connections` or `identity shortcut connections` is One solution for all problems

`Why we are calling Skip Connections also Residual Connections?`

Lets answer

1. `Why call it Residual?`
2. `Where is Residue?`

Consider a neural network block with input as x We would like to `learn true distribution` --> $f(x)$

Let us denote `Residual` or `difference` between this as:

$$R(x) = output - input$$

$$R(x) = f(x) - x$$

rearanging above equation:

$$f(x) = R(x) + x$$

$f(x)$ ==> True Output

$R(x)$ ==> Residual

Our `Residual Block is overall trying to learn true output` --> $f(x)$

Look closely at above image

- we have an identity connection coming as x
- layers are actually trying to learn `Residual` --> $R(x)$

`Concluding:`

- Layers in a traditional network are learning true output --> $f(x)$
- Layers in a Residual Network(ResNet) are learning residual --> $R(x)$, hence name : Residual Block

It is also Observed that:

- it is easier to learn residual of output and input, rather than only input
- in addition, network can now learn identity function by simply setting residual as zero

In backpropagation, problem of `Vanishing Gradient` arises with increasing number of layers

- because of these `skip connections` we can propagate larger gradients to initial layers
- these layers also could learn as fast as final layers which giving us `ability to train deeper networks`

Lets see **`How to arrange Residual Block and Identity Connections for optimal gradient flow`**

- It is Observed that `Pre-activations with Batch Normalizations generally give best results`
  - see `right-most residual block in image`, this will gives most promising results

res

`BN` => batch normalization

While training ResNets

- we either `train layers in residual blocks` or
- `Skip training for those layers using skip connections`

Different parts of networks will be trained at different rates for different training data points `based on how error flows backward in network`

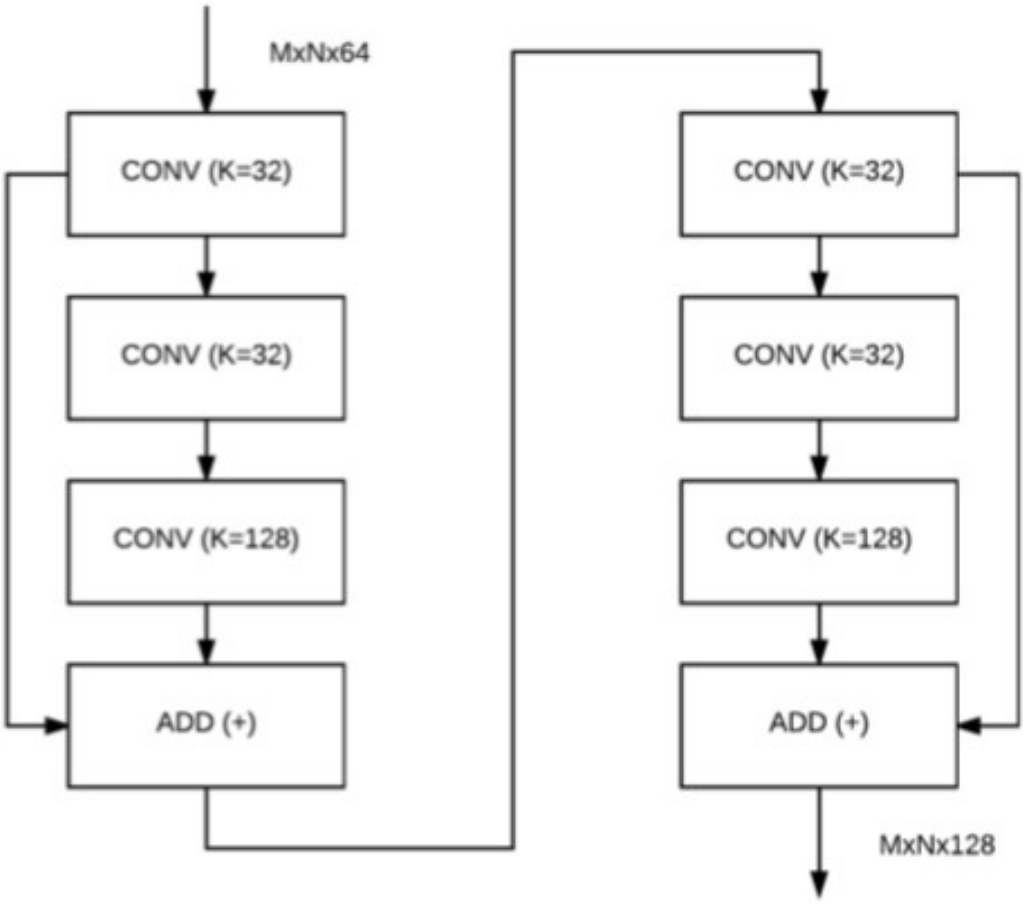
This can be thought of as training an ensemble of different models on dataset and getting best possible accuracy

In general

- We do not know `optimal number of layers` (or residual blocks) required for a neural network which might depend on complexity of dataset
- Instead of treating `number of layers as an important hyperparameter to tune`
  - by adding `skip connections to our network`, we are allowing network to skip training for layers that are
    - not useful and
    - do not add value in overall accuracy
- In a way, skip connections make our neural networks dynamic to tune number of layers during training optimally
- ResNet is an artificial neural network (ANN) of a kind that stacks residual blocks on top of each other to form a network
- Deep residual networks like `ResNet-50 model` is a kind of `CNN that is 50 layers deep`
  - in 2015 ResNet was a solution to `vanishing gradient problem`
- ResNet incorporates identity shortcut connections which essentially skip the training of one or more layers — creating a residual block.

The authors then proposed an “optimized” residual block, adding an extension called a bottleneck. It would reduce the dimensionality in the first two CONV layers (1/4 of the filters

learned in the final CONV layer) and then increase again during the final CONV layer. Here are two residual modules stacked on top of each other.



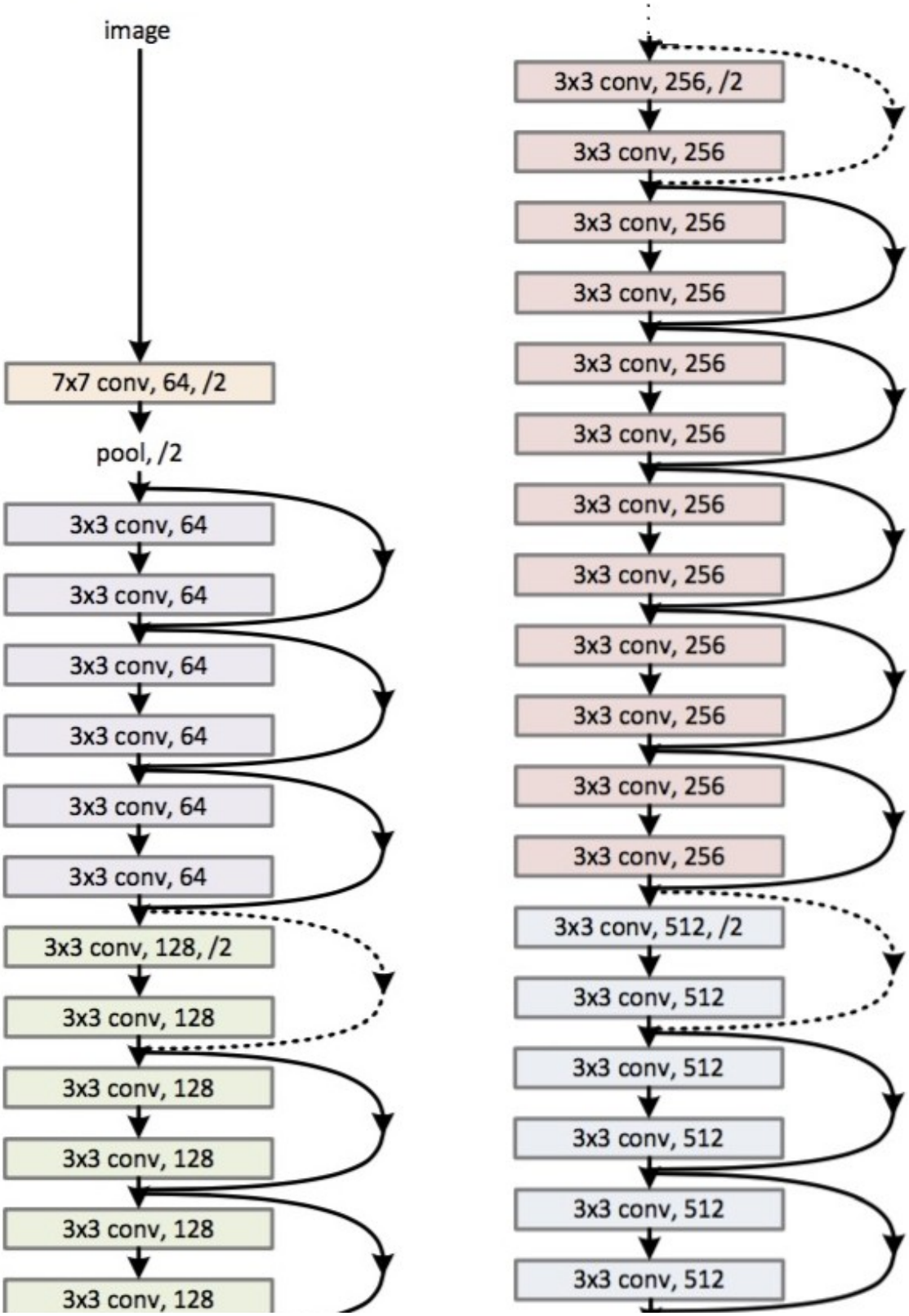
Finally a second paper on the residual module got published which was called **Identity Mapping in Deep Residual Networks** which provided an even better version of the residual block: the **pre-activation residual model** This allows the gradients to propagate through the shortcut connections to any of the earlier layers without any hindrance.

So instead of starting with a convolution (weight), we start with a series of (BN => RELU => CONV) * N layers (assuming bottleneck is being used). Then, the residual module outputs the addition operation that's fed into the next residual module in the network (since residual modules are stacked on top of each other).

What is Deep Residual Learning used for?

ResNet was created with the aim of tackling this exact problem. Deep residual nets make use of residual blocks to improve the accuracy of the models. The concept of "skip connections," which lies at the core of the residual blocks, is the strength of this type of neural network.

34-layer residual



ResNet-34 Architecture

The first ResNet architecture was the Resnet-34 which involved the insertion of shortcut connections in turning a plain network into its residual network counterpart. In this case, the plain network was inspired by VGG neural networks (VGG-16, VGG-19), with the convolutional networks having 3×3 filters. However, compared to VGGNets, ResNets have fewer filters and lower complexity. The 34-layer ResNet achieves a performance of 3.6 bn FLOPs, compared to 1.8bn FLOPs of smaller 18-layer ResNets.

It also followed two simple design rules – the layers had the same number of filters for the same output feature map size, and the number of filters doubled in case the feature map size was halved in order to preserve the time complexity per layer. It consisted of 34 weighted layers.

The shortcut connections (Skip connections) were added to this plain network. While the input and output dimensions were the same, the identity shortcuts were directly used. With an increase in the dimensions, there were two options to be considered. The first was that the shortcut would still perform identity mapping while extra zero entries would be padded for increasing dimensions. The other option was to use the projection shortcut to match dimensions.

Resnet-50 Architecture

While the Resnet50 architecture is based on the above model, there is one major difference. In this case, the building block was modified into a bottleneck design due to concerns over the time taken to train the layers. This used a stack of 3 layers instead of the earlier 2.

Therefore, each of the 2-layer blocks in Resnet34 was replaced with a 3-layer bottleneck block, forming the Resnet 50 architecture. This has much higher accuracy than the 34-layer ResNet model. The 50-layer ResNet achieves a performance of 3.8 bn FLOPS

ResNet-101 and ResNet-152 Architecture

Large Residual Networks such as 101-layer ResNet101 or ResNet152 are constructed by using more 3-layer blocks. And even at increased network depth, the 152-layer ResNet has much lower complexity (at 11.3bn FLOPS) than VGG-16 or VGG-19 nets (15.3/19.6bn FLOPS

Resnet50 : Residual Network on Custom Dataset

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os

dataset_path =
os.listdir('imagenet-object-localization-challenge/ILSVRC/Data/CLS-
LOC/train2')
```

```
labels_type =
os.listdir('imagenet-object-localization-challenge/ILSVRC/Data/CLS-
LOC/train2')
#print (room_types)  #what kinds of rooms are in this dataset``

print("Total Labels Found: ", len(dataset_path))
```

```
Total Labels Found:  11
```

```
labels = []

for item in labels_type:
 # Get all the file names
 all_labels =
os.listdir('imagenet-object-localization-challenge/ILSVRC/Data/CLS-
LOC/train2' + '/' +item)


 # Add them to the list
 for label in all_labels:
    labels.append((item,
str('imagenet-object-localization-challenge/ILSVRC/Data/CLS-LOC/train2
' + '/' +item) + '/' + label))



# Build a dataframe
data_df = pd.DataFrame(data=labels, columns=['labels type', 'image'])
#print(rooms_df.head())
#print(rooms_df.tail())


# Let's check how many samples for each category are present
print("Total number of rooms in the dataset: ", len(data_df))

labels_count = data_df['labels type'].value_counts()

#print("rooms in each category: ")
#print(room_count)

data_df.head(11)
```

```
Total number of rooms in the dataset:  14300
```

	labels type	image
0	n01491361	imagenet-object-localization-challenge/ILSVRC/...
1	n01491361	imagenet-object-localization-challenge/ILSVRC/...
2	n01491361	imagenet-object-localization-challenge/ILSVRC/...
3	n01491361	imagenet-object-localization-challenge/ILSVRC/...

```
4    n01491361  imagenet-object-localization-challenge/ILSVRC/...
5    n01491361  imagenet-object-localization-challenge/ILSVRC/...
6    n01491361  imagenet-object-localization-challenge/ILSVRC/...
7    n01491361  imagenet-object-localization-challenge/ILSVRC/...
8    n01491361  imagenet-object-localization-challenge/ILSVRC/...
9    n01491361  imagenet-object-localization-challenge/ILSVRC/...
10   n01491361  imagenet-object-localization-challenge/ILSVRC/...
```

```
import cv2
path =
'imagenet-object-localization-challenge/ILSVRC/Data/CLS-LOC/train2/'


im_size = 64

images = []
labels = []

for i in labels_type:
    data_path = path + str(i)  # entered in 1st folder and then 2nd
folder and then 3rd folder
    filenames = [i for i in os.listdir(data_path) ]
   # print(filenames)  # will get the names of all images
    for f in filenames:
        img = cv2.imread(data_path + '/' + f)  # reading that image as
array
        #print(img)  # will get the image as an array
        img = cv2.resize(img, (im_size, im_size))
        images.append(img)
        labels.append(i)
labels
```

```
['n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
```

```
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
```

```
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
```

```
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
```

```
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
```

```
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
```

```
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
```

```
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
```

```
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
```

```
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
```

```
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
```

```
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
```

```
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
```

```
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
```

```
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
```

```
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
```

```
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
```

```
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
```

```
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
```

```
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
```

```
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
 'n01491361',
```

```
 'n01491361',
 ...]

images=(np.array(images)).astype('float32') / 255.0
images

array([[[[0.15294118, 0.08235294, 0.03921569],
         [0.14901961, 0.09019608, 0.03921569],
         [0.18431373, 0.08235294, 0.04313726],
         ...,
         [0.13725491, 0.07450981, 0.05490196],
         [0.14901961, 0.05490196, 0.03137255],
         [0.10588235, 0.07450981, 0.02352941]],

        [[0.20784314, 0.07843138, 0.05098039],
         [0.18039216, 0.09019608, 0.05098039],
         [0.23529412, 0.09411765, 0.04705882],
         ...,
         [0.13725491, 0.0627451 , 0.03137255],
         [0.13333334, 0.07058824, 0.03529412],
         [0.1254902 , 0.05098039, 0.01568628]],

        [[0.40784314, 0.39607844, 0.3764706 ],
         [0.4627451 , 0.38431373, 0.36078432],
         [0.25490198, 0.09411765, 0.07058824],
         ...,
         [0.15294118, 0.0627451 , 0.03921569],
         [0.12156863, 0.0627451 , 0.01960784],
         [0.12156863, 0.05882353, 0.01568628]],

        ...,

        [[0.80784315, 0.7254902 , 0.62352943],
         [0.8745098 , 0.8156863 , 0.72156864],
         [0.88235295, 0.827451  , 0.7294118 ],
         ...,
         [0.38039216, 0.32156864, 0.31764707],
         [0.35686275, 0.30980393, 0.30588236],
         [0.3764706 , 0.2901961 , 0.2784314 ]],

        [[0.8509804 , 0.8235294 , 0.7529412 ],
         [0.8117647 , 0.7411765 , 0.6313726 ],
         [0.8235294 , 0.827451  , 0.7137255 ],
         ...,
         [0.38039216, 0.32156864, 0.27450982],
         [0.36862746, 0.3137255 , 0.3254902 ],
         [0.34117648, 0.2627451 , 0.2627451 ]],

        [[0.73333335, 0.7647059 , 0.6627451 ],
         [0.85490197, 0.8352941 , 0.7137255 ],
```

```
         [0.8039216 , 0.8392157 , 0.7647059 ],
         ...,
         [0.5058824 , 0.4745098 , 0.48235294],
         [0.44705883, 0.44313726, 0.42352942],
         [0.4       , 0.38431373, 0.35686275]]],


       [[[0.53333336, 0.21960784, 0.03137255],
         [0.5529412 , 0.21176471, 0.01960784],
         [0.6431373 , 0.2509804 , 0.03921569],
         ...,
         [0.5058824 , 0.21568628, 0.01960784],
         [0.4627451 , 0.2       , 0.00784314],
         [0.44313726, 0.21568628, 0.02745098]],

        [[0.44313726, 0.2       , 0.01176471],
         [0.4117647 , 0.18039216, 0.00784314],
         [0.5882353 , 0.27058825, 0.07450981],
         ...,
         [0.6156863 , 0.23137255, 0.        ],
         [0.48235294, 0.1882353 , 0.01176471],
         [0.4       , 0.17254902, 0.01568628]],

        [[0.54901963, 0.21176471, 0.00784314],
         [0.47843137, 0.20784314, 0.01568628],
         [0.50980395, 0.2       , 0.01568628],
         ...,
         [0.41568628, 0.19215687, 0.01568628],
         [0.46666667, 0.19215687, 0.00784314],
         [0.45490196, 0.21960784, 0.03529412]],

        ...,

        [[0.2509804 , 0.09411765, 0.02352941],
         [0.23137255, 0.08627451, 0.        ],
         [0.23921569, 0.09803922, 0.00784314],
         ...,
         [0.23921569, 0.10196079, 0.01176471],
         [0.2784314 , 0.10588235, 0.01568628],
         [0.27058825, 0.11372549, 0.01568628]],

        [[0.25490198, 0.09803922, 0.02745098],
         [0.2627451 , 0.10196079, 0.01568628],
         [0.27058825, 0.10980392, 0.02352941],
         ...,
         [0.2627451 , 0.11372549, 0.01960784],
         [0.2784314 , 0.11764706, 0.02745098],
         [0.27058825, 0.10196079, 0.01960784]],

        [[0.2509804 , 0.10196079, 0.03137255],
```

```
         [0.25490198, 0.09411765, 0.00784314],
         [0.2627451 , 0.10196079, 0.01568628],
         ...,
         [0.2627451 , 0.10588235, 0.01176471],
         [0.2627451 , 0.10196079, 0.00392157],
         [0.26666668, 0.10196079, 0.01568628]]],


       [[[0.69411767, 0.74509805, 0.67058825],
         [0.74509805, 0.7882353 , 0.7529412 ],
         [0.8352941 , 0.87058824, 0.88235295],
         ...,
         [0.50980395, 0.4745098 , 0.34509805],
         [0.45882353, 0.42745098, 0.3137255 ],
         [0.40784314, 0.38039216, 0.27450982]],

        [[0.7372549 , 0.78431374, 0.7529412 ],
         [0.827451  , 0.8627451 , 0.8784314 ],
         [0.84313726, 0.8784314 , 0.8901961 ],
         ...,
         [0.5058824 , 0.47058824, 0.34117648],
         [0.4627451 , 0.43137255, 0.31764707],
         [0.4117647 , 0.3882353 , 0.28235295]],

        [[0.81960785, 0.85490197, 0.8666667 ],
         [0.8352941 , 0.87058824, 0.88235295],
         [0.84705883, 0.88235295, 0.89411765],
         ...,
         [0.52156866, 0.4862745 , 0.36078432],
         [0.4627451 , 0.43529412, 0.32156864],
         [0.41960785, 0.39215687, 0.29411766]],

        ...,

        [[0.34117648, 0.3254902 , 0.2509804 ],
         [0.38039216, 0.36078432, 0.2784314 ],
         [0.4       , 0.3882353 , 0.29411766],
         ...,
         [0.11372549, 0.12941177, 0.10588235],
         [0.10588235, 0.11764706, 0.10196079],
         [0.09803922, 0.10980392, 0.09019608]],

        [[0.32156864, 0.31764707, 0.23921569],
         [0.35686275, 0.34117648, 0.26666668],
         [0.39215687, 0.37254903, 0.2901961 ],
         ...,
         [0.12156863, 0.12941177, 0.11372549],
         [0.10588235, 0.11764706, 0.09803922],
         [0.09803922, 0.11372549, 0.09411765]],
```

```
        [[0.4       , 0.3882353 , 0.37254903],
         [0.32941177, 0.3137255 , 0.23137255],
         [0.36862746, 0.35686275, 0.27450982],
         ...,
         [0.10980392, 0.1254902 , 0.10588235],
         [0.10196079, 0.11764706, 0.09803922],
         [0.09019608, 0.10588235, 0.08627451]]],


       ...,


       [[[0.5294118 , 0.5647059 , 0.4745098 ],
         [0.6039216 , 0.6392157 , 0.54509807],
         [0.18431373, 0.19215687, 0.19215687],
         ...,
         [0.6313726 , 0.627451  , 0.7529412 ],
         [0.60784316, 0.6039216 , 0.7176471 ],
         [0.5764706 , 0.5647059 , 0.6901961 ]],

        [[0.5019608 , 0.53333336, 0.43137255],
         [0.50980395, 0.5411765 , 0.4509804 ],
         [0.10196079, 0.10980392, 0.10980392],
         ...,
         [0.6509804 , 0.6509804 , 0.77254903],
         [0.6509804 , 0.6431373 , 0.7529412 ],
         [0.6156863 , 0.6117647 , 0.7372549 ]],

        [[0.5372549 , 0.57254905, 0.49019608],
         [0.5176471 , 0.54509807, 0.4627451 ],
         [0.6901961 , 0.72156864, 0.6313726 ],
         ...,
         [0.6784314 , 0.654902  , 0.8117647 ],
         [0.6431373 , 0.6509804 , 0.78039217],
         [0.62352943, 0.62352943, 0.7647059 ]],

        ...,

        [[0.1882353 , 0.12156863, 0.08627451],
         [0.19607843, 0.12941177, 0.08627451],
         [0.2       , 0.13333334, 0.08235294],
         ...,
         [0.34117648, 0.38039216, 0.3254902 ],
         [0.34117648, 0.38039216, 0.29411766],
         [0.32941177, 0.3764706 , 0.29411766]],

        [[0.17254902, 0.12156863, 0.08235294],
         [0.18431373, 0.12941177, 0.08627451],
         [0.19215687, 0.1254902 , 0.08627451],
         ...,
```

```
         [0.3372549 , 0.39215687, 0.32156864],
         [0.34901962, 0.3882353 , 0.30980393],
         [0.32941177, 0.37254903, 0.2901961 ]],

        [[0.16078432, 0.12156863, 0.08627451],
         [0.18039216, 0.13333334, 0.09411765],
         [0.18431373, 0.1254902 , 0.09019608],
         ...,
         [0.3529412 , 0.40784314, 0.33333334],
         [0.34117648, 0.38431373, 0.30980393],
         [0.33333334, 0.37254903, 0.3019608 ]]],


       [[[0.7490196 , 0.6       , 0.28235295],
         [0.75686276, 0.6039216 , 0.28235295],
         [0.7647059 , 0.59607846, 0.31764707],
         ...,
         [0.6156863 , 0.41568628, 0.02352941],
         [0.62352943, 0.4       , 0.01176471],
         [0.6039216 , 0.3882353 , 0.00784314]],

        [[0.7490196 , 0.59607846, 0.32156864],
         [0.7607843 , 0.6       , 0.3137255 ],
         [0.7529412 , 0.59607846, 0.3019608 ],
         ...,
         [0.627451  , 0.41568628, 0.04313726],
         [0.6156863 , 0.40392157, 0.00784314],
         [0.6156863 , 0.40784314, 0.03529412]],

        [[0.7607843 , 0.59607846, 0.3529412 ],
         [0.7490196 , 0.59607846, 0.29411766],
         [0.7647059 , 0.59607846, 0.30980393],
         ...,
         [0.6313726 , 0.40392157, 0.02745098],
         [0.62352943, 0.40392157, 0.05098039],
         [0.6313726 , 0.41960785, 0.26666668]],

        ...,

        [[0.75686276, 0.78431374, 0.77254903],
         [0.7529412 , 0.8       , 0.8039216 ],
         [0.49411765, 0.56078434, 0.5254902 ],
         ...,
         [0.76862746, 0.7882353 , 0.7921569 ],
         [0.7921569 , 0.84705883, 0.9137255 ],
         [0.77254903, 0.79607844, 0.85490197]],

        [[0.6901961 , 0.74509805, 0.77254903],
         [0.5882353 , 0.6392157 , 0.6431373 ],
         [0.6509804 , 0.75686276, 0.80784315],
```

```
         ...,
         [0.7176471 , 0.7490196 , 0.8117647 ],
         [0.7647059 , 0.77254903, 0.8117647 ],
         [0.80784315, 0.7882353 , 0.8627451 ]],

        [[0.7411765 , 0.8156863 , 0.8509804 ],
         [0.7294118 , 0.8156863 , 0.8901961 ],
         [0.7411765 , 0.8       , 0.85490197],
         ...,
         [0.7058824 , 0.7372549 , 0.7607843 ],
         [0.7254902 , 0.75686276, 0.7921569 ],
         [0.7254902 , 0.75686276, 0.8039216 ]]],


       [[[0.74509805, 0.77254903, 0.79607844],
         [0.9843137 , 0.9764706 , 0.99607843],
         [0.99215686, 0.99215686, 0.99607843],
         ...,
         [0.05882353, 0.03921569, 0.04313726],
         [0.07450981, 0.05490196, 0.05098039],
         [0.05098039, 0.02745098, 0.03137255]],

        [[0.60784316, 0.6431373 , 0.654902  ],
         [0.7058824 , 0.69803923, 0.7372549 ],
         [0.6666667 , 0.6313726 , 0.6509804 ],
         ...,
         [0.01568628, 0.00784314, 0.00784314],
         [0.03529412, 0.01568628, 0.01568628],
         [0.05098039, 0.02745098, 0.03529412]],

        [[0.29803923, 0.3647059 , 0.42352942],
         [0.54509807, 0.6117647 , 0.63529414],
         [0.64705884, 0.58431375, 0.5921569 ],
         ...,
         [0.03137255, 0.02352941, 0.02352941],
         [0.05098039, 0.04313726, 0.04313726],
         [0.09803922, 0.1254902 , 0.1254902 ]],

        ...,

        [[0.4745098 , 0.5411765 , 0.5686275 ],
         [0.41568628, 0.53333336, 0.52156866],
         [0.4627451 , 0.5529412 , 0.56078434],
         ...,
         [0.06666667, 0.05882353, 0.10196079],
         [0.0627451 , 0.0627451 , 0.08627451],
         [0.08627451, 0.07058824, 0.10196079]],

        [[0.43529412, 0.54901963, 0.5647059 ],
         [0.4627451 , 0.5372549 , 0.54901963],
```

```
         [0.43137255, 0.50980395, 0.5137255 ],
         ...,
         [0.06666667, 0.07058824, 0.07843138],
         [0.07058824, 0.10196079, 0.10588235],
         [0.10588235, 0.10588235, 0.1254902 ]],

        [[0.4745098 , 0.5686275 , 0.5686275 ],
         [0.38039216, 0.5058824 , 0.49411765],
         [0.43137255, 0.5529412 , 0.54901963],
         ...,
         [0.13333334, 0.17254902, 0.20784314],
         [0.05882353, 0.07843138, 0.09019608],
         [0.07843138, 0.10196079, 0.1254902 ]]], dtype=float32)
```

```
images = images.astype('float32') / 255.0
images
```

```
array([[[[0.15294118, 0.08235294, 0.03921569],
         [0.14901961, 0.09019608, 0.03921569],
         [0.18431373, 0.08235294, 0.04313726],
         ...,
         [0.13725491, 0.07450981, 0.05490196],
         [0.14901961, 0.05490196, 0.03137255],
         [0.10588235, 0.07450981, 0.02352941]],

        [[0.20784314, 0.07843138, 0.05098039],
         [0.18039216, 0.09019608, 0.05098039],
         [0.23529412, 0.09411765, 0.04705882],
         ...,
         [0.13725491, 0.0627451 , 0.03137255],
         [0.13333334, 0.07058824, 0.03529412],
         [0.1254902 , 0.05098039, 0.01568628]],

        [[0.40784314, 0.39607844, 0.3764706 ],
         [0.4627451 , 0.38431373, 0.36078432],
         [0.25490198, 0.09411765, 0.07058824],
         ...,
         [0.15294118, 0.0627451 , 0.03921569],
         [0.12156863, 0.0627451 , 0.01960784],
         [0.12156863, 0.05882353, 0.01568628]],

        ...,

        [[0.80784315, 0.7254902 , 0.62352943],
         [0.8745098 , 0.8156863 , 0.72156864],
         [0.88235295, 0.827451  , 0.7294118 ],
         ...,
         [0.38039216, 0.32156864, 0.31764707],
         [0.35686275, 0.30980393, 0.30588236],
         [0.3764706 , 0.2901961 , 0.2784314 ]],
```

```
       [[0.8509804 , 0.8235294 , 0.7529412 ],
        [0.8117647 , 0.7411765 , 0.6313726 ],
        [0.8235294 , 0.827451  , 0.7137255 ],
        ...,
        [0.38039216, 0.32156864, 0.27450982],
        [0.36862746, 0.3137255 , 0.3254902 ],
        [0.34117648, 0.2627451 , 0.2627451 ]],

       [[0.73333335, 0.7647059 , 0.6627451 ],
        [0.85490197, 0.8352941 , 0.7137255 ],
        [0.8039216 , 0.8392157 , 0.7647059 ],
        ...,
        [0.5058824 , 0.4745098 , 0.48235294],
        [0.44705883, 0.44313726, 0.42352942],
        [0.4       , 0.38431373, 0.35686275]]],


      [[[0.53333336, 0.21960784, 0.03137255],
        [0.5529412 , 0.21176471, 0.01960784],
        [0.6431373 , 0.2509804 , 0.03921569],
        ...,
        [0.5058824 , 0.21568628, 0.01960784],
        [0.4627451 , 0.2       , 0.00784314],
        [0.44313726, 0.21568628, 0.02745098]],

       [[0.44313726, 0.2       , 0.01176471],
        [0.4117647 , 0.18039216, 0.00784314],
        [0.5882353 , 0.27058825, 0.07450981],
        ...,
        [0.6156863 , 0.23137255, 0.        ],
        [0.48235294, 0.1882353 , 0.01176471],
        [0.4       , 0.17254902, 0.01568628]],

       [[0.54901963, 0.21176471, 0.00784314],
        [0.47843137, 0.20784314, 0.01568628],
        [0.50980395, 0.2       , 0.01568628],
        ...,
        [0.41568628, 0.19215687, 0.01568628],
        [0.46666667, 0.19215687, 0.00784314],
        [0.45490196, 0.21960784, 0.03529412]],

       ...,

       [[0.2509804 , 0.09411765, 0.02352941],
        [0.23137255, 0.08627451, 0.        ],
        [0.23921569, 0.09803922, 0.00784314],
        ...,
        [0.23921569, 0.10196079, 0.01176471],
        [0.2784314 , 0.10588235, 0.01568628],
```

```
         [0.27058825, 0.11372549, 0.01568628]],

        [[0.25490198, 0.09803922, 0.02745098],
         [0.2627451 , 0.10196079, 0.01568628],
         [0.27058825, 0.10980392, 0.02352941],
         ...,
         [0.2627451 , 0.11372549, 0.01960784],
         [0.2784314 , 0.11764706, 0.02745098],
         [0.27058825, 0.10196079, 0.01960784]],

        [[0.2509804 , 0.10196079, 0.03137255],
         [0.25490198, 0.09411765, 0.00784314],
         [0.2627451 , 0.10196079, 0.01568628],
         ...,
         [0.2627451 , 0.10588235, 0.01176471],
         [0.2627451 , 0.10196079, 0.00392157],
         [0.26666668, 0.10196079, 0.01568628]]],


       [[[0.69411767, 0.74509805, 0.67058825],
         [0.74509805, 0.7882353 , 0.7529412 ],
         [0.8352941 , 0.87058824, 0.88235295],
         ...,
         [0.50980395, 0.4745098 , 0.34509805],
         [0.45882353, 0.42745098, 0.3137255 ],
         [0.40784314, 0.38039216, 0.27450982]],

        [[0.7372549 , 0.78431374, 0.7529412 ],
         [0.827451  , 0.8627451 , 0.8784314 ],
         [0.84313726, 0.8784314 , 0.8901961 ],
         ...,
         [0.5058824 , 0.47058824, 0.34117648],
         [0.4627451 , 0.43137255, 0.31764707],
         [0.4117647 , 0.3882353 , 0.28235295]],

        [[0.81960785, 0.85490197, 0.8666667 ],
         [0.8352941 , 0.87058824, 0.88235295],
         [0.84705883, 0.88235295, 0.89411765],
         ...,
         [0.52156866, 0.4862745 , 0.36078432],
         [0.4627451 , 0.43529412, 0.32156864],
         [0.41960785, 0.39215687, 0.29411766]],

        ...,

        [[0.34117648, 0.3254902 , 0.2509804 ],
         [0.38039216, 0.36078432, 0.2784314 ],
         [0.4       , 0.3882353 , 0.29411766],
         ...,
         [0.11372549, 0.12941177, 0.10588235],
```

```
         [0.10588235, 0.11764706, 0.10196079],
         [0.09803922, 0.10980392, 0.09019608]],

        [[0.32156864, 0.31764707, 0.23921569],
         [0.35686275, 0.34117648, 0.26666668],
         [0.39215687, 0.37254903, 0.2901961 ],
         ...,
         [0.12156863, 0.12941177, 0.11372549],
         [0.10588235, 0.11764706, 0.09803922],
         [0.09803922, 0.11372549, 0.09411765]],

        [[0.4       , 0.3882353 , 0.37254903],
         [0.32941177, 0.3137255 , 0.23137255],
         [0.36862746, 0.35686275, 0.27450982],
         ...,
         [0.10980392, 0.1254902 , 0.10588235],
         [0.10196079, 0.11764706, 0.09803922],
         [0.09019608, 0.10588235, 0.08627451]]],


       ...,


       [[[1.        , 1.        , 1.        ],
         [1.        , 1.        , 1.        ],
         [1.        , 1.        , 1.        ],
         ...,
         [1.        , 1.        , 1.        ],
         [1.        , 1.        , 1.        ],
         [1.        , 1.        , 1.        ]],

        [[1.        , 1.        , 1.        ],
         [1.        , 1.        , 1.        ],
         [1.        , 1.        , 1.        ],
         ...,
         [1.        , 1.        , 1.        ],
         [1.        , 1.        , 1.        ],
         [1.        , 1.        , 1.        ]],

        [[1.        , 1.        , 1.        ],
         [1.        , 1.        , 1.        ],
         [1.        , 1.        , 1.        ],
         ...,
         [1.        , 1.        , 1.        ],
         [1.        , 1.        , 1.        ],
         [1.        , 1.        , 1.        ]],

        ...,

        [[0.972549  , 0.98039216, 0.9843137 ],
```

```
         [0.972549  , 0.98039216, 0.9843137 ],
         [0.9607843 , 0.9843137 , 0.98039216],
         ...,
         [0.9254902 , 0.9411765 , 0.9411765 ],
         [0.9529412 , 0.972549  , 0.96862745],
         [0.9647059 , 0.972549  , 0.9764706 ]],

        [[0.972549  , 0.98039216, 0.9843137 ],
         [0.972549  , 0.98039216, 0.9843137 ],
         [0.9607843 , 0.9843137 , 0.98039216],
         ...,
         [0.92941177, 0.9372549 , 0.9411765 ],
         [0.9490196 , 0.96862745, 0.972549  ],
         [0.9529412 , 0.9764706 , 0.972549  ]],

        [[0.972549  , 0.98039216, 0.9843137 ],
         [0.972549  , 0.972549  , 0.99607843],
         [0.972549  , 0.972549  , 0.99607843],
         ...,
         [0.9529412 , 0.9411765 , 0.9490196 ],
         [0.9764706 , 0.9607843 , 0.96862745],
         [0.9607843 , 0.96862745, 0.96862745]]],


       [[[0.06666667, 0.40784314, 0.23529412],
         [0.15294118, 0.54509807, 0.28235295],
         [0.14117648, 0.31764707, 0.30980393],
         ...,
         [0.36862746, 0.47843137, 0.48235294],
         [0.39215687, 0.5411765 , 0.5058824 ],
         [0.2784314 , 0.5568628 , 0.4       ]],

        [[0.02352941, 0.33333334, 0.18431373],
         [0.03921569, 0.56078434, 0.36862746],
         [0.13725491, 0.19607843, 0.22352941],
         ...,
         [0.48235294, 0.5254902 , 0.59607846],
         [0.40784314, 0.47058824, 0.4627451 ],
         [0.2784314 , 0.6039216 , 0.39215687]],

        [[0.17254902, 0.35686275, 0.34901962],
         [0.03921569, 0.49803922, 0.28235295],
         [0.29411766, 0.4509804 , 0.5686275 ],
         ...,
         [0.4117647 , 0.45490196, 0.5176471 ],
         [0.39215687, 0.42745098, 0.41960785],
         [0.23529412, 0.57254905, 0.3529412 ]],

        ...,
```

```
       [[0.25882354, 0.5058824 , 0.34117648],
        [0.02352941, 0.05490196, 0.0627451 ],
        [0.0627451 , 0.07058824, 0.07058824],
        ...,
        [0.21960784, 0.3647059 , 0.43137255],
        [0.3372549 , 0.5647059 , 0.54901963],
        [0.21176471, 0.38431373, 0.35686275]],

       [[0.2784314 , 0.58431375, 0.36078432],
        [0.20392157, 0.2509804 , 0.24705882],
        [0.2784314 , 0.29803923, 0.29803923],
        ...,
        [0.34509805, 0.3529412 , 0.34901962],
        [0.0627451 , 0.3019608 , 0.19607843],
        [0.21176471, 0.43529412, 0.4117647 ]],

       [[0.21960784, 0.47843137, 0.31764707],
        [0.21568628, 0.35686275, 0.2627451 ],
        [0.43529412, 0.48235294, 0.45490196],
        ...,
        [0.24705882, 0.32941177, 0.33333334],
        [0.22745098, 0.44313726, 0.35686275],
        [0.09803922, 0.37254903, 0.30588236]]],


      [[[0.39607844, 0.5254902 , 0.5294118 ],
        [0.23921569, 0.42352942, 0.45490196],
        [0.3137255 , 0.49019608, 0.5411765 ],
        ...,
        [0.11764706, 0.27450982, 0.36078432],
        [0.3764706 , 0.52156866, 0.5921569 ],
        [0.3372549 , 0.5137255 , 0.5568628 ]],

       [[0.3529412 , 0.47058824, 0.5372549 ],
        [0.33333334, 0.53333336, 0.5686275 ],
        [0.41568628, 0.6039216 , 0.67058825],
        ...,
        [0.3882353 , 0.48235294, 0.53333336],
        [0.2       , 0.31764707, 0.3529412 ],
        [0.05098039, 0.16078432, 0.16862746]],

       [[0.27058825, 0.5176471 , 0.5411765 ],
        [0.23529412, 0.34509805, 0.39607844],
        [0.30588236, 0.49803922, 0.54509807],
        ...,
        [0.39607844, 0.48235294, 0.5137255 ],
        [0.24313726, 0.46666667, 0.45882353],
        [0.48235294, 0.6862745 , 0.7372549 ]],

       ...,
```

```
        [[0.2784314 , 0.46666667, 0.54509807],
         [0.47058824, 0.6039216 , 0.6745098 ],
         [0.3764706 , 0.54509807, 0.5921569 ],
         ...,
         [0.01960784, 0.02745098, 0.02745098],
         [0.01960784, 0.02745098, 0.02745098],
         [0.02352941, 0.03137255, 0.03137255]],

        [[0.36078432, 0.5372549 , 0.62352943],
         [0.3372549 , 0.5019608 , 0.5921569 ],
         [0.28235295, 0.40784314, 0.49803922],
         ...,
         [0.02352941, 0.03137255, 0.03137255],
         [0.01960784, 0.02745098, 0.02745098],
         [0.01960784, 0.02745098, 0.02745098]],

        [[0.34117648, 0.5372549 , 0.61960787],
         [0.3019608 , 0.50980395, 0.6156863 ],
         [0.3254902 , 0.5137255 , 0.6       ],
         ...,
         [0.01960784, 0.02745098, 0.02745098],
         [0.00784314, 0.01568628, 0.01568628],
         [0.01960784, 0.02745098, 0.02745098]]], dtype=float32)
```

```
np.save("data.npy", images)
```

```
images = np.load("data.npy")
```

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
import numpy as np

# Sample data (replace with your actual data)
y = data_df['labels type'].values

# Label Encoding
y_labelencoder = LabelEncoder()
y_encoded = y_labelencoder.fit_transform(y)  # Converts to integer
labels

# Reshape for OneHotEncoder
y_reshaped = y_encoded.reshape(-1, 1)

# OneHot Encoding - Modern approach
onehotencoder = OneHotEncoder(sparse_output=False)  #
sparse_output=False returns array instead of sparse matrix
Y = onehotencoder.fit_transform(y_reshaped)

print(Y.shape)  # Should show (n_samples, n_categories)
```

```
(14300, 11)
```

```
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split


images, Y = shuffle(images, Y, random_state=1)

train_x, test_x, train_y, test_y = train_test_split(images, Y,
test_size=0.05, random_state=415)

#inpect the shape of the training and testing.
print(train_x.shape)
print(train_y.shape)
print(test_x.shape)
print(test_y.shape)
```

```
(13585, 64, 64, 3)
(13585, 11)
(715, 64, 64, 3)
(715, 11)
```

```
import numpy as np
from keras import layers
from keras.layers import Input, Add, Dense, Activation, ZeroPadding2D,
BatchNormalization, Flatten, Conv2D, AveragePooling2D, MaxPooling2D,
GlobalMaxPooling2D
from keras.models import Model, load_model
from keras.preprocessing import image
from keras.utils import layer_utils
from keras.utils.data_utils import get_file
from keras.applications.imagenet_utils import preprocess_input
import pydot
from IPython.display import SVG
from keras.utils.vis_utils import model_to_dot
from keras.utils import plot_model

from keras.initializers import glorot_uniform
import scipy.misc
from matplotlib.pyplot import imshow

from keras.initializers import glorot_uniform
from keras.layers import Input, Add, Dense, Activation, ZeroPadding2D,
BatchNormalization, Flatten, Conv2D, AveragePooling2D, MaxPooling2D,
GlobalMaxPooling2D
```

```
2025-04-26 18:24:56.859585: I
tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow
binary is optimized with oneAPI Deep Neural Network Library (oneDNN)
to use the following CPU instructions in performance-critical
operations:  SSE4.1 SSE4.2 AVX AVX2 AVX_VNNI FMA
To enable them in other operations, rebuild TensorFlow with the
appropriate compiler flags.
```

```
2025-04-26 18:24:56.969953: I tensorflow/core/util/util.cc:169] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
```

Images

Identity Block

what is meaning of padding='valid' and padding='same' ?

In convolutional neural networks (CNNs), **`padding`** determines how the input data is padded before applying convolution. Here's the key difference between `padding='valid'` and `padding='same'`:

1. `padding='valid'` (No Padding)

- **Behavior**:
  - No padding is added to the input.
  - The convolution operates **only on valid positions** where the kernel fully overlaps with the input.
  - The output size is smaller than the input size.
- **Output Size Formula**:

```
output_size = floor((input_size - kernel_size) / stride) + 1
```

  - Example: Input=5×5, Kernel=3×3, Stride=1 → Output=3×3.
- **When to Use**:
  - When you want to reduce spatial dimensions (downsampling).
  - Common in early layers of CNNs to shrink feature maps.

2. `padding='same'` (Zero Padding)

- **Behavior**:
  - Pads the input with zeros **to ensure the output size equals the input size** (when stride=1).
  - Output dimensions are preserved (if `stride=1`).
- **Output Size Formula**:

```
output_size = ceil(input_size / stride)
```

  - Example: Input=5×5, Kernel=3×3, Stride=1 → Output=5×5 (pads 1 pixel on each side).

- **Padding Calculation**:
  - Total padding added:

```
pad_total = (kernel_size - 1)
```

  - Split equally on both sides (e.g., Kernel=3×3 → pad=1 on each side).
- **When to Use**:
  - When you want to **preserve spatial dimensions** (e.g., in deeper layers).
  - Useful for autoencoders or segmentation tasks.

Key Differences

Feature	`padding='valid'`	`padding='same'`
Padding	None	Zero-padded
Output Size	Smaller than input	Same as input (if stride=1)
Information Loss	Loses edge information	Preserves edges
Compute Efficiency	Faster (less computation)	Slightly slower

Visual Example

Input: 5×5, Kernel: 3×3, Stride: 1

- **`valid`**:

```
Input: 5x5 → Output: 3x3 (no padding)
```

- **`same`**:

```
Input: 5x5 → Output: 5x5 (pads with 1 zero on each side)
```

Code Example (TensorFlow/Keras):

```
from tensorflow.keras.layers import Conv2D

# Valid padding (output shrinks)
conv_valid = Conv2D(filters=32, kernel_size=3, padding='valid')

# Same padding (output size = input size)
conv_same = Conv2D(filters=32, kernel_size=3, padding='same')
```

When to Choose Which?

- Use **`valid`** to:
  - Reduce computational cost.
  - Gradually downsample feature maps (e.g., in VGG/ResNet).
- Use **`same`** to:

- Preserve spatial resolution (e.g., U-Net for segmentation).
- Avoid losing edge information.

Edge Cases

- If `stride > 1`, `same` padding ensures:

```
output_size = ceil(input_size / stride)
```

- For asymmetric padding (e.g., kernel=5×5), `same` adds 2 pixels on one side and 1 on the other.

```
# x is input, y=F(x)
# identity block simply means input should be equal to output.
#  y = x + F(x)   the layers in a traditional network are learning the
true output H(x)
# F(x) = y - x   the layers in a residual network are learning the
residual F(x)
# Hence, the name: Residual Block.



def identity_block(X, f, filters, stage, block):
    """

    Arguments:
    X -- input of shape (m, height, width, channel)
    f -- shape of the middle CONV's window for the main path
    filters -- python list of integers, defining the number of filters
in the CONV layers of the main path
    stage -- integer, used to name the layers, depending on their
position in the network
    block -- string/character, used to name the layers, depending on
their position in the network

    Returns:
    X -- output of the identity block, tensor of shape (n_H, n_W, n_C)
    """

    # defining name basis
    conv_name_base = 'res' + str(stage) + block + '_branch'
    bn_name_base = 'bn' + str(stage) + block + '_branch'

    # Retrieve Filters
    F1, F2, F3 = filters

    # Saving the input value.we need this later to add to the output.
    X_shortcut = X

    # First component of main path
```

```
    X = Conv2D(filters = F1, kernel_size = (1, 1), strides = (1,1),
padding = 'valid', name = conv_name_base + '2a')(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2a')(X)
    X = Activation('relu')(X)


    # Second component of main path (≈3 lines)
    X = Conv2D(filters = F2, kernel_size = (f, f), strides = (1,1),
padding = 'same', name = conv_name_base + '2b')(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2b')(X)
    X = Activation('relu')(X)

    # Third component of main path (≈2 lines)
    X = Conv2D(filters = F3, kernel_size = (1, 1), strides = (1,1),
padding = 'valid', name = conv_name_base + '2c')(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2c')(X)

    # Final step: Add shortcut value to main path, and pass it through
a RELU activation
    X = Add()([X, X_shortcut])
    X = Activation('relu')(X)


    return X
```

Convolutional Block

```
def convolutional_block(X, f, filters, stage, block, s = 2):

    # defining name basis
    conv_name_base = 'res' + str(stage) + block + '_branch'
    bn_name_base = 'bn' + str(stage) + block + '_branch'

    # Retrieve Filters
    F1, F2, F3 = filters

    # Save the input value
    X_shortcut = X


    # First layer
    X = Conv2D(F1, (1, 1), strides = (s,s), name = conv_name_base +
'2a')(X) # 1,1 is filter size
    X = BatchNormalization(axis = 3, name = bn_name_base + '2a')(X)  #
normalization on channels
    X = Activation('relu')(X)
```

```
    # Second layer  (f,f)=3*3 filter by default
    X = Conv2D(filters = F2, kernel_size = (f, f), strides = (1,1),
padding = 'same', name = conv_name_base + '2b')(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2b')(X)
    X = Activation('relu')(X)


    # Third layer
    X = Conv2D(filters = F3, kernel_size = (1, 1), strides = (1,1),
padding = 'valid', name = conv_name_base + '2c')(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2c')(X)


    ##### SHORTCUT PATH ####
    X_shortcut = Conv2D(filters = F3, kernel_size = (1, 1), strides =
(s,s), padding = 'valid', name = conv_name_base + '1')(X_shortcut)
    X_shortcut = BatchNormalization(axis = 3, name = bn_name_base +
'1')(X_shortcut)

    # Final step: Add shortcut value here, and pass it through a RELU
activation
    X = Add()([X, X_shortcut])
    X = Activation('relu')(X)


    return X
```

Creating ResNet 50

Resnet50

```
#Each ResNet block is either 2 layer deep
def ResNet50(input_shape=(64, 64, 3), classes=11):
    """
    Implementation of the ResNet50 architecture:
    CONV2D -> BATCHNORM -> RELU -> MAXPOOL -> CONVBLOCK -> IDBLOCK*2 -
> CONVBLOCK -> IDBLOCK*3
    -> CONVBLOCK -> IDBLOCK*5 -> CONVBLOCK -> IDBLOCK*2 -> AVGPOOL ->
TOPLAYER

    """

    # Define the input as a tensor with shape input_shape
    X_input = Input(input_shape)

    # Zero-Padding
    X = ZeroPadding2D((3, 3))(X_input) #3,3 padding
```

```
    # Stage 1
    X = Conv2D(64, (7, 7), strides=(2, 2), name='conv1')(X) #64
filters of 7*7
    X = BatchNormalization(axis=3, name='bn_conv1')(X) #batchnorm
applied on channels
    X = Activation('relu')(X)
    X = MaxPooling2D((3, 3), strides=(2, 2))(X) #window size is 3*3

    # Stage 2
    X = convolutional_block(X, f=3, filters=[64, 64, 256], stage=2,
block='a', s=1)
    # convolutional_block is a function defined above.
Convolutional_block have 3 layers.
    #filters=[64, 64, 256] first 64 is for 1st layer and 2nd 64 is for
2nd layer and 256 is for 3rd layer of convultional block
    # below are the conv layers from convolutional_block function
    #X = Conv2D(F1, (1, 1), strides = (s,s), name = conv_name_base +
'2a')(X)
    #X = Conv2D(F2, kernel_size = (f, f), strides = (1,1), padding =
'same', name = conv_name_base + '2b')(X)
    #X = Conv2D(F3, (1, 1), strides = (s,s), name = conv_name_base +
'2a')(X)

    X = identity_block(X, 3, [64, 64, 256], stage=2, block='b')
    #X = Conv2D(filters = F1, kernel_size = (1, 1), strides = (1,1),
padding = 'valid', name = conv_name_base + '2a')(X)
    #X = Conv2D(filters = F2, kernel_size = (f, f), strides = (1,1),
padding = 'same', name = conv_name_base + '2b')(X)
    #X = Conv2D(filters = F3, kernel_size = (1, 1), strides = (1,1),
padding = 'valid', name = conv_name_base + '2c')(X)

    X = identity_block(X, 3, [64, 64, 256], stage=2, block='c')
    #X = Conv2D(filters = F1, kernel_size = (1, 1), strides = (1,1),
padding = 'valid', name = conv_name_base + '2a')(X)
    #X = Conv2D(filters = F2, kernel_size = (f, f), strides = (1,1),
padding = 'same', name = conv_name_base + '2b')(X)
    #X = Conv2D(filters = F3, kernel_size = (1, 1), strides = (1,1),
padding = 'valid', name = conv_name_base + '2c')(X)


    ### START CODE HERE ###

    # Stage 3
    X = convolutional_block(X, f = 3, filters = [128, 128, 512], stage
= 3, block='a', s = 2)
    X = identity_block(X, 3, [128, 128, 512], stage=3, block='b')
    X = identity_block(X, 3, [128, 128, 512], stage=3, block='c')
    X = identity_block(X, 3, [128, 128, 512], stage=3, block='d')
```

```
    # Stage 4
    X = convolutional_block(X, f = 3, filters = [256, 256, 1024],
stage = 4, block='a', s = 2)
    X = identity_block(X, 3, [256, 256, 1024], stage=4, block='b')
    X = identity_block(X, 3, [256, 256, 1024], stage=4, block='c')
    X = identity_block(X, 3, [256, 256, 1024], stage=4, block='d')
    X = identity_block(X, 3, [256, 256, 1024], stage=4, block='e')
    X = identity_block(X, 3, [256, 256, 1024], stage=4, block='f')

    # Stage 5
    X = convolutional_block(X, f = 3, filters = [512, 512, 2048],
stage = 5, block='a', s = 2)
    X = identity_block(X, 3, [512, 512, 2048], stage=5, block='b')
    X = identity_block(X, 3, [512, 512, 2048], stage=5, block='c')

    # AVGPOOL
    X = AveragePooling2D((2,2), name="avg_pool")(X)

    ### END CODE HERE ###

    # output layer
    X = Flatten()(X)
    X = Dense(classes, activation='softmax', name='fc' + str(classes),
kernel_initializer = glorot_uniform(seed=0))(X)


    # Create model
    model = Model(inputs = X_input, outputs = X, name='ResNet50')

    return model

model = ResNet50(input_shape = (64, 64, 3), classes = 11)
```

```
2025-04-26 18:25:22.150104: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:980] successful
NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2025-04-26 18:25:22.166009: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:980] successful
NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2025-04-26 18:25:22.166150: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:980] successful
NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2025-04-26 18:25:22.166771: I
tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow
binary is optimized with oneAPI Deep Neural Network Library (oneDNN)
to use the following CPU instructions in performance-critical
operations:  SSE4.1 SSE4.2 AVX AVX2 AVX_VNNI FMA
```

```
To enable them in other operations, rebuild TensorFlow with the
appropriate compiler flags.
2025-04-26 18:25:22.167863: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:980] successful
NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2025-04-26 18:25:22.168010: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:980] successful
NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2025-04-26 18:25:22.168096: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:980] successful
NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2025-04-26 18:25:22.611548: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:980] successful
NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2025-04-26 18:25:22.611694: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:980] successful
NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2025-04-26 18:25:22.611778: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:980] successful
NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2025-04-26 18:25:22.611865: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1616] Created
device /job:localhost/replica:0/task:0/device:GPU:0 with 6179 MB
memory:  -> device: 0, name: NVIDIA GeForce RTX 4060 Laptop GPU, pci
bus id: 0000:01:00.0, compute capability: 8.9
```

```
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

model.summary()
```

```
Model: "ResNet50"
__________________________________________________________________
_____________________________
 Layer (type)                   Output Shape         Param #
Connected to
==================================================================
=============================
 input_1 (InputLayer)           [(None, 64, 64, 3)]  0           []



 zero_padding2d (ZeroPadding2D)  (None, 70, 70, 3)   0
['input_1[0][0]']
```

```
 conv1 (Conv2D)                 (None, 32, 32, 64)   9472
['zero_padding2d[0][0]']


 bn_conv1 (BatchNormalization)  (None, 32, 32, 64)   256
['conv1[0][0]']


 activation (Activation)        (None, 32, 32, 64)   0
['bn_conv1[0][0]']


 max_pooling2d (MaxPooling2D)   (None, 15, 15, 64)   0
['activation[0][0]']


 res2a_branch2a (Conv2D)        (None, 15, 15, 64)   4160
['max_pooling2d[0][0]']


 bn2a_branch2a (BatchNormalizat  (None, 15, 15, 64)  256
['res2a_branch2a[0][0]']
 ion)



 activation_1 (Activation)      (None, 15, 15, 64)   0
['bn2a_branch2a[0][0]']


 res2a_branch2b (Conv2D)        (None, 15, 15, 64)   36928
['activation_1[0][0]']


 bn2a_branch2b (BatchNormalizat  (None, 15, 15, 64)  256
['res2a_branch2b[0][0]']
 ion)



 activation_2 (Activation)      (None, 15, 15, 64)   0
['bn2a_branch2b[0][0]']


 res2a_branch2c (Conv2D)        (None, 15, 15, 256)  16640
['activation_2[0][0]']


 res2a_branch1 (Conv2D)         (None, 15, 15, 256)  16640
```

```
['max_pooling2d[0][0]']


 bn2a_branch2c (BatchNormalizat  (None, 15, 15, 256)  1024
['res2a_branch2c[0][0]']
 ion)



 bn2a_branch1 (BatchNormalizati  (None, 15, 15, 256)  1024
['res2a_branch1[0][0]']
 on)



 add (Add)                      (None, 15, 15, 256)  0
['bn2a_branch2c[0][0]',

'bn2a_branch1[0][0]']


 activation_3 (Activation)      (None, 15, 15, 256)  0
['add[0][0]']


 res2b_branch2a (Conv2D)        (None, 15, 15, 64)   16448
['activation_3[0][0]']


 bn2b_branch2a (BatchNormalizat  (None, 15, 15, 64)  256
['res2b_branch2a[0][0]']
 ion)



 activation_4 (Activation)      (None, 15, 15, 64)   0
['bn2b_branch2a[0][0]']


 res2b_branch2b (Conv2D)        (None, 15, 15, 64)   36928
['activation_4[0][0]']


 bn2b_branch2b (BatchNormalizat  (None, 15, 15, 64)  256
['res2b_branch2b[0][0]']
 ion)



 activation_5 (Activation)      (None, 15, 15, 64)   0
['bn2b_branch2b[0][0]']
```

```
 res2b_branch2c (Conv2D)        (None, 15, 15, 256)  16640
['activation_5[0][0]']


 bn2b_branch2c (BatchNormalizat  (None, 15, 15, 256)  1024
['res2b_branch2c[0][0]']
 ion)



 add_1 (Add)                    (None, 15, 15, 256)  0
['bn2b_branch2c[0][0]',

'activation_3[0][0]']


 activation_6 (Activation)      (None, 15, 15, 256)  0
['add_1[0][0]']


 res2c_branch2a (Conv2D)        (None, 15, 15, 64)   16448
['activation_6[0][0]']


 bn2c_branch2a (BatchNormalizat  (None, 15, 15, 64)  256
['res2c_branch2a[0][0]']
 ion)



 activation_7 (Activation)      (None, 15, 15, 64)   0
['bn2c_branch2a[0][0]']


 res2c_branch2b (Conv2D)        (None, 15, 15, 64)   36928
['activation_7[0][0]']


 bn2c_branch2b (BatchNormalizat  (None, 15, 15, 64)  256
['res2c_branch2b[0][0]']
 ion)



 activation_8 (Activation)      (None, 15, 15, 64)   0
['bn2c_branch2b[0][0]']


 res2c_branch2c (Conv2D)        (None, 15, 15, 256)  16640
```

```
['activation_8[0][0]']


 bn2c_branch2c (BatchNormalizat  (None, 15, 15, 256)  1024
['res2c_branch2c[0][0]']
 ion)



 add_2 (Add)                    (None, 15, 15, 256)  0
['bn2c_branch2c[0][0]',

'activation_6[0][0]']


 activation_9 (Activation)      (None, 15, 15, 256)  0
['add_2[0][0]']


 res3a_branch2a (Conv2D)        (None, 8, 8, 128)    32896
['activation_9[0][0]']


 bn3a_branch2a (BatchNormalizat  (None, 8, 8, 128)   512
['res3a_branch2a[0][0]']
 ion)



 activation_10 (Activation)     (None, 8, 8, 128)    0
['bn3a_branch2a[0][0]']


 res3a_branch2b (Conv2D)        (None, 8, 8, 128)    147584
['activation_10[0][0]']


 bn3a_branch2b (BatchNormalizat  (None, 8, 8, 128)   512
['res3a_branch2b[0][0]']
 ion)



 activation_11 (Activation)     (None, 8, 8, 128)    0
['bn3a_branch2b[0][0]']


 res3a_branch2c (Conv2D)        (None, 8, 8, 512)    66048
['activation_11[0][0]']
```

```
 res3a_branch1 (Conv2D)          (None, 8, 8, 512)    131584
['activation_9[0][0]']


 bn3a_branch2c (BatchNormalizat  (None, 8, 8, 512)   2048
['res3a_branch2c[0][0]']
 ion)



 bn3a_branch1 (BatchNormalizati  (None, 8, 8, 512)   2048
['res3a_branch1[0][0]']
 on)



 add_3 (Add)                     (None, 8, 8, 512)    0
['bn3a_branch2c[0][0]',

'bn3a_branch1[0][0]']


 activation_12 (Activation)     (None, 8, 8, 512)    0
['add_3[0][0]']


 res3b_branch2a (Conv2D)         (None, 8, 8, 128)    65664
['activation_12[0][0]']


 bn3b_branch2a (BatchNormalizat  (None, 8, 8, 128)   512
['res3b_branch2a[0][0]']
 ion)



 activation_13 (Activation)     (None, 8, 8, 128)    0
['bn3b_branch2a[0][0]']


 res3b_branch2b (Conv2D)         (None, 8, 8, 128)    147584
['activation_13[0][0]']


 bn3b_branch2b (BatchNormalizat  (None, 8, 8, 128)   512
['res3b_branch2b[0][0]']
 ion)



 activation_14 (Activation)     (None, 8, 8, 128)    0
```

```
['bn3b_branch2b[0][0]']


 res3b_branch2c (Conv2D)        (None, 8, 8, 512)    66048
['activation_14[0][0]']


 bn3b_branch2c (BatchNormalizat  (None, 8, 8, 512)   2048
['res3b_branch2c[0][0]']
 ion)



 add_4 (Add)                    (None, 8, 8, 512)    0
['bn3b_branch2c[0][0]',

'activation_12[0][0]']


 activation_15 (Activation)     (None, 8, 8, 512)    0
['add_4[0][0]']


 res3c_branch2a (Conv2D)        (None, 8, 8, 128)    65664
['activation_15[0][0]']


 bn3c_branch2a (BatchNormalizat  (None, 8, 8, 128)   512
['res3c_branch2a[0][0]']
 ion)



 activation_16 (Activation)     (None, 8, 8, 128)    0
['bn3c_branch2a[0][0]']


 res3c_branch2b (Conv2D)        (None, 8, 8, 128)    147584
['activation_16[0][0]']


 bn3c_branch2b (BatchNormalizat  (None, 8, 8, 128)   512
['res3c_branch2b[0][0]']
 ion)



 activation_17 (Activation)     (None, 8, 8, 128)    0
['bn3c_branch2b[0][0]']
```

```
 res3c_branch2c (Conv2D)        (None, 8, 8, 512)    66048
['activation_17[0][0]']


 bn3c_branch2c (BatchNormalizat  (None, 8, 8, 512)   2048
['res3c_branch2c[0][0]']
 ion)



 add_5 (Add)                    (None, 8, 8, 512)    0
['bn3c_branch2c[0][0]',

'activation_15[0][0]']


 activation_18 (Activation)     (None, 8, 8, 512)    0
['add_5[0][0]']


 res3d_branch2a (Conv2D)        (None, 8, 8, 128)    65664
['activation_18[0][0]']


 bn3d_branch2a (BatchNormalizat  (None, 8, 8, 128)   512
['res3d_branch2a[0][0]']
 ion)



 activation_19 (Activation)     (None, 8, 8, 128)    0
['bn3d_branch2a[0][0]']


 res3d_branch2b (Conv2D)        (None, 8, 8, 128)    147584
['activation_19[0][0]']


 bn3d_branch2b (BatchNormalizat  (None, 8, 8, 128)   512
['res3d_branch2b[0][0]']
 ion)



 activation_20 (Activation)     (None, 8, 8, 128)    0
['bn3d_branch2b[0][0]']


 res3d_branch2c (Conv2D)        (None, 8, 8, 512)    66048
['activation_20[0][0]']
```

```
 bn3d_branch2c (BatchNormalizat  (None, 8, 8, 512)   2048
['res3d_branch2c[0][0]']
 ion)


 add_6 (Add)                    (None, 8, 8, 512)    0
['bn3d_branch2c[0][0]',

'activation_18[0][0]']


 activation_21 (Activation)     (None, 8, 8, 512)    0
['add_6[0][0]']


 res4a_branch2a (Conv2D)        (None, 4, 4, 256)    131328
['activation_21[0][0]']


 bn4a_branch2a (BatchNormalizat  (None, 4, 4, 256)   1024
['res4a_branch2a[0][0]']
 ion)


 activation_22 (Activation)     (None, 4, 4, 256)    0
['bn4a_branch2a[0][0]']


 res4a_branch2b (Conv2D)        (None, 4, 4, 256)    590080
['activation_22[0][0]']


 bn4a_branch2b (BatchNormalizat  (None, 4, 4, 256)   1024
['res4a_branch2b[0][0]']
 ion)


 activation_23 (Activation)     (None, 4, 4, 256)    0
['bn4a_branch2b[0][0]']


 res4a_branch2c (Conv2D)        (None, 4, 4, 1024)   263168
['activation_23[0][0]']


 res4a_branch1 (Conv2D)         (None, 4, 4, 1024)   525312
['activation_21[0][0]']
```

```
 bn4a_branch2c (BatchNormalizat  (None, 4, 4, 1024)  4096
['res4a_branch2c[0][0]']
 ion)


 bn4a_branch1 (BatchNormalizati  (None, 4, 4, 1024)  4096
['res4a_branch1[0][0]']
 on)


 add_7 (Add)                    (None, 4, 4, 1024)   0
['bn4a_branch2c[0][0]',

'bn4a_branch1[0][0]']


 activation_24 (Activation)     (None, 4, 4, 1024)   0
['add_7[0][0]']


 res4b_branch2a (Conv2D)        (None, 4, 4, 256)    262400
['activation_24[0][0]']


 bn4b_branch2a (BatchNormalizat  (None, 4, 4, 256)   1024
['res4b_branch2a[0][0]']
 ion)


 activation_25 (Activation)     (None, 4, 4, 256)    0
['bn4b_branch2a[0][0]']


 res4b_branch2b (Conv2D)        (None, 4, 4, 256)    590080
['activation_25[0][0]']


 bn4b_branch2b (BatchNormalizat  (None, 4, 4, 256)   1024
['res4b_branch2b[0][0]']
 ion)


 activation_26 (Activation)     (None, 4, 4, 256)    0
['bn4b_branch2b[0][0]']
```

```
 res4b_branch2c (Conv2D)        (None, 4, 4, 1024)   263168
['activation_26[0][0]']


 bn4b_branch2c (BatchNormalizat  (None, 4, 4, 1024)  4096
['res4b_branch2c[0][0]']
 ion)



 add_8 (Add)                    (None, 4, 4, 1024)   0
['bn4b_branch2c[0][0]',

'activation_24[0][0]']


 activation_27 (Activation)     (None, 4, 4, 1024)   0
['add_8[0][0]']


 res4c_branch2a (Conv2D)        (None, 4, 4, 256)    262400
['activation_27[0][0]']


 bn4c_branch2a (BatchNormalizat  (None, 4, 4, 256)   1024
['res4c_branch2a[0][0]']
 ion)



 activation_28 (Activation)     (None, 4, 4, 256)    0
['bn4c_branch2a[0][0]']


 res4c_branch2b (Conv2D)        (None, 4, 4, 256)    590080
['activation_28[0][0]']


 bn4c_branch2b (BatchNormalizat  (None, 4, 4, 256)   1024
['res4c_branch2b[0][0]']
 ion)



 activation_29 (Activation)     (None, 4, 4, 256)    0
['bn4c_branch2b[0][0]']


 res4c_branch2c (Conv2D)        (None, 4, 4, 1024)   263168
['activation_29[0][0]']
```

```
 bn4c_branch2c (BatchNormalizat  (None, 4, 4, 1024)  4096
['res4c_branch2c[0][0]']
 ion)


 add_9 (Add)                    (None, 4, 4, 1024)   0
['bn4c_branch2c[0][0]',

'activation_27[0][0]']


 activation_30 (Activation)     (None, 4, 4, 1024)   0
['add_9[0][0]']


 res4d_branch2a (Conv2D)        (None, 4, 4, 256)    262400
['activation_30[0][0]']


 bn4d_branch2a (BatchNormalizat  (None, 4, 4, 256)   1024
['res4d_branch2a[0][0]']
 ion)


 activation_31 (Activation)     (None, 4, 4, 256)    0
['bn4d_branch2a[0][0]']


 res4d_branch2b (Conv2D)        (None, 4, 4, 256)    590080
['activation_31[0][0]']


 bn4d_branch2b (BatchNormalizat  (None, 4, 4, 256)   1024
['res4d_branch2b[0][0]']
 ion)


 activation_32 (Activation)     (None, 4, 4, 256)    0
['bn4d_branch2b[0][0]']


 res4d_branch2c (Conv2D)        (None, 4, 4, 1024)   263168
['activation_32[0][0]']


 bn4d_branch2c (BatchNormalizat  (None, 4, 4, 1024)  4096
```

```
['res4d_branch2c[0][0]']
 ion)



 add_10 (Add)                   (None, 4, 4, 1024)   0
['bn4d_branch2c[0][0]',

'activation_30[0][0]']


 activation_33 (Activation)     (None, 4, 4, 1024)   0
['add_10[0][0]']


 res4e_branch2a (Conv2D)        (None, 4, 4, 256)    262400
['activation_33[0][0]']


 bn4e_branch2a (BatchNormalizat  (None, 4, 4, 256)   1024
['res4e_branch2a[0][0]']
 ion)



 activation_34 (Activation)     (None, 4, 4, 256)    0
['bn4e_branch2a[0][0]']


 res4e_branch2b (Conv2D)        (None, 4, 4, 256)    590080
['activation_34[0][0]']


 bn4e_branch2b (BatchNormalizat  (None, 4, 4, 256)   1024
['res4e_branch2b[0][0]']
 ion)



 activation_35 (Activation)     (None, 4, 4, 256)    0
['bn4e_branch2b[0][0]']


 res4e_branch2c (Conv2D)        (None, 4, 4, 1024)   263168
['activation_35[0][0]']


 bn4e_branch2c (BatchNormalizat  (None, 4, 4, 1024)  4096
['res4e_branch2c[0][0]']
 ion)
```

```
 add_11 (Add)                   (None, 4, 4, 1024)   0
['bn4e_branch2c[0][0]',

'activation_33[0][0]']


 activation_36 (Activation)     (None, 4, 4, 1024)   0
['add_11[0][0]']


 res4f_branch2a (Conv2D)        (None, 4, 4, 256)    262400
['activation_36[0][0]']


 bn4f_branch2a (BatchNormalizat  (None, 4, 4, 256)   1024
['res4f_branch2a[0][0]']
 ion)



 activation_37 (Activation)     (None, 4, 4, 256)    0
['bn4f_branch2a[0][0]']


 res4f_branch2b (Conv2D)        (None, 4, 4, 256)    590080
['activation_37[0][0]']


 bn4f_branch2b (BatchNormalizat  (None, 4, 4, 256)   1024
['res4f_branch2b[0][0]']
 ion)



 activation_38 (Activation)     (None, 4, 4, 256)    0
['bn4f_branch2b[0][0]']


 res4f_branch2c (Conv2D)        (None, 4, 4, 1024)   263168
['activation_38[0][0]']


 bn4f_branch2c (BatchNormalizat  (None, 4, 4, 1024)  4096
['res4f_branch2c[0][0]']
 ion)



 add_12 (Add)                   (None, 4, 4, 1024)   0
```

```
['bn4f_branch2c[0][0]',

'activation_36[0][0]']


 activation_39 (Activation)     (None, 4, 4, 1024)   0
['add_12[0][0]']


 res5a_branch2a (Conv2D)        (None, 2, 2, 512)    524800
['activation_39[0][0]']


 bn5a_branch2a (BatchNormalizat  (None, 2, 2, 512)   2048
['res5a_branch2a[0][0]']
 ion)



 activation_40 (Activation)     (None, 2, 2, 512)    0
['bn5a_branch2a[0][0]']


 res5a_branch2b (Conv2D)        (None, 2, 2, 512)    2359808
['activation_40[0][0]']


 bn5a_branch2b (BatchNormalizat  (None, 2, 2, 512)   2048
['res5a_branch2b[0][0]']
 ion)



 activation_41 (Activation)     (None, 2, 2, 512)    0
['bn5a_branch2b[0][0]']


 res5a_branch2c (Conv2D)        (None, 2, 2, 2048)   1050624
['activation_41[0][0]']


 res5a_branch1 (Conv2D)         (None, 2, 2, 2048)   2099200
['activation_39[0][0]']


 bn5a_branch2c (BatchNormalizat  (None, 2, 2, 2048)  8192
['res5a_branch2c[0][0]']
 ion)
```

```
 bn5a_branch1 (BatchNormalizati  (None, 2, 2, 2048)  8192
['res5a_branch1[0][0]']
 on)



 add_13 (Add)                   (None, 2, 2, 2048)   0
['bn5a_branch2c[0][0]',

'bn5a_branch1[0][0]']


 activation_42 (Activation)     (None, 2, 2, 2048)   0
['add_13[0][0]']


 res5b_branch2a (Conv2D)        (None, 2, 2, 512)    1049088
['activation_42[0][0]']


 bn5b_branch2a (BatchNormalizat  (None, 2, 2, 512)   2048
['res5b_branch2a[0][0]']
 ion)



 activation_43 (Activation)     (None, 2, 2, 512)    0
['bn5b_branch2a[0][0]']


 res5b_branch2b (Conv2D)        (None, 2, 2, 512)    2359808
['activation_43[0][0]']


 bn5b_branch2b (BatchNormalizat  (None, 2, 2, 512)   2048
['res5b_branch2b[0][0]']
 ion)



 activation_44 (Activation)     (None, 2, 2, 512)    0
['bn5b_branch2b[0][0]']


 res5b_branch2c (Conv2D)        (None, 2, 2, 2048)   1050624
['activation_44[0][0]']


 bn5b_branch2c (BatchNormalizat  (None, 2, 2, 2048)  8192
['res5b_branch2c[0][0]']
 ion)
```

```
 add_14 (Add)                   (None, 2, 2, 2048)   0
['bn5b_branch2c[0][0]',

'activation_42[0][0]']


 activation_45 (Activation)     (None, 2, 2, 2048)   0
['add_14[0][0]']


 res5c_branch2a (Conv2D)        (None, 2, 2, 512)    1049088
['activation_45[0][0]']


 bn5c_branch2a (BatchNormalizat  (None, 2, 2, 512)   2048
['res5c_branch2a[0][0]']
 ion)



 activation_46 (Activation)     (None, 2, 2, 512)    0
['bn5c_branch2a[0][0]']


 res5c_branch2b (Conv2D)        (None, 2, 2, 512)    2359808
['activation_46[0][0]']


 bn5c_branch2b (BatchNormalizat  (None, 2, 2, 512)   2048
['res5c_branch2b[0][0]']
 ion)



 activation_47 (Activation)     (None, 2, 2, 512)    0
['bn5c_branch2b[0][0]']


 res5c_branch2c (Conv2D)        (None, 2, 2, 2048)   1050624
['activation_47[0][0]']


 bn5c_branch2c (BatchNormalizat  (None, 2, 2, 2048)  8192
['res5c_branch2c[0][0]']
 ion)
```

```
 add_15 (Add)                   (None, 2, 2, 2048)   0
['bn5c_branch2c[0][0]',

'activation_45[0][0]']


 activation_48 (Activation)     (None, 2, 2, 2048)   0
['add_15[0][0]']


 avg_pool (AveragePooling2D)    (None, 1, 1, 2048)   0
['activation_48[0][0]']


 flatten (Flatten)              (None, 2048)         0
['avg_pool[0][0]']


 fc11 (Dense)                   (None, 11)           22539
['flatten[0][0]']


==================================================================
==============================
Total params: 23,610,251
Trainable params: 23,557,131
Non-trainable params: 53,120
__________________________________________________________________
______________________________

model.fit(train_x, train_y, epochs = 100, batch_size = 32)

Epoch 1/100
425/425 [==============================] - 12s 28ms/step - loss:
1.3445 - accuracy: 0.5292
Epoch 2/100
425/425 [==============================] - 12s 28ms/step - loss:
1.1822 - accuracy: 0.5787
Epoch 3/100
425/425 [==============================] - 12s 28ms/step - loss:
1.2706 - accuracy: 0.5648
Epoch 4/100
425/425 [==============================] - 12s 28ms/step - loss:
1.2382 - accuracy: 0.5683
Epoch 5/100
425/425 [==============================] - 12s 28ms/step - loss:
1.1356 - accuracy: 0.5982
Epoch 6/100
425/425 [==============================] - 12s 28ms/step - loss:
1.1014 - accuracy: 0.6210
```

```
Epoch 7/100
425/425 [==============================] - 12s 28ms/step - loss:
0.9847 - accuracy: 0.6562
Epoch 8/100
425/425 [==============================] - 12s 28ms/step - loss:
0.9259 - accuracy: 0.6765
Epoch 9/100
425/425 [==============================] - 12s 28ms/step - loss:
0.8340 - accuracy: 0.7076
Epoch 10/100
425/425 [==============================] - 12s 28ms/step - loss:
0.7245 - accuracy: 0.7408
Epoch 11/100
425/425 [==============================] - 12s 28ms/step - loss:
0.7728 - accuracy: 0.7335
Epoch 12/100
425/425 [==============================] - 12s 28ms/step - loss:
0.6964 - accuracy: 0.7552
Epoch 13/100
425/425 [==============================] - 12s 28ms/step - loss:
0.6909 - accuracy: 0.7597
Epoch 14/100
425/425 [==============================] - 12s 28ms/step - loss:
0.4903 - accuracy: 0.8335
Epoch 15/100
425/425 [==============================] - 12s 28ms/step - loss:
0.5504 - accuracy: 0.8147
Epoch 16/100
425/425 [==============================] - 12s 28ms/step - loss:
0.4267 - accuracy: 0.8526
Epoch 17/100
425/425 [==============================] - 12s 28ms/step - loss:
0.4751 - accuracy: 0.8425
Epoch 18/100
425/425 [==============================] - 12s 28ms/step - loss:
0.3171 - accuracy: 0.9014
Epoch 19/100
425/425 [==============================] - 12s 28ms/step - loss:
0.2866 - accuracy: 0.9023
Epoch 20/100
425/425 [==============================] - 12s 28ms/step - loss:
0.2481 - accuracy: 0.9170
Epoch 21/100
425/425 [==============================] - 12s 28ms/step - loss:
0.3020 - accuracy: 0.9049
Epoch 22/100
425/425 [==============================] - 12s 28ms/step - loss:
0.2194 - accuracy: 0.9298
Epoch 23/100
```

```
425/425 [==============================] - 12s 28ms/step - loss:
0.1818 - accuracy: 0.9416
Epoch 24/100
425/425 [==============================] - 12s 28ms/step - loss:
0.1944 - accuracy: 0.9388
Epoch 25/100
425/425 [==============================] - 12s 28ms/step - loss:
0.3108 - accuracy: 0.9089
Epoch 26/100
425/425 [==============================] - 12s 29ms/step - loss:
0.1340 - accuracy: 0.9575
Epoch 27/100
425/425 [==============================] - 12s 29ms/step - loss:
0.1553 - accuracy: 0.9526
Epoch 28/100
425/425 [==============================] - 12s 29ms/step - loss:
0.1643 - accuracy: 0.9538
Epoch 29/100
425/425 [==============================] - 12s 29ms/step - loss:
0.1359 - accuracy: 0.9575
Epoch 30/100
425/425 [==============================] - 12s 28ms/step - loss:
0.1268 - accuracy: 0.9615
Epoch 31/100
425/425 [==============================] - 12s 29ms/step - loss:
0.1189 - accuracy: 0.9658
Epoch 32/100
425/425 [==============================] - 12s 29ms/step - loss:
0.1679 - accuracy: 0.9524
Epoch 33/100
425/425 [==============================] - 12s 28ms/step - loss:
0.0922 - accuracy: 0.9693
Epoch 34/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0946 - accuracy: 0.9703
Epoch 35/100
425/425 [==============================] - 12s 29ms/step - loss:
0.1392 - accuracy: 0.9593
Epoch 36/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0922 - accuracy: 0.9715
Epoch 37/100
425/425 [==============================] - 12s 29ms/step - loss:
0.1012 - accuracy: 0.9690
Epoch 38/100
425/425 [==============================] - 12s 28ms/step - loss:
0.0862 - accuracy: 0.9758
Epoch 39/100
425/425 [==============================] - 12s 29ms/step - loss:
```

```
0.1150 - accuracy: 0.9657
Epoch 40/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0882 - accuracy: 0.9738
Epoch 41/100
425/425 [==============================] - 12s 29ms/step - loss:
0.1543 - accuracy: 0.9559
Epoch 42/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0714 - accuracy: 0.9776
Epoch 43/100
425/425 [==============================] - 12s 29ms/step - loss:
0.1033 - accuracy: 0.9703
Epoch 44/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0686 - accuracy: 0.9772
Epoch 45/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0678 - accuracy: 0.9786
Epoch 46/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0666 - accuracy: 0.9752
Epoch 47/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0649 - accuracy: 0.9785
Epoch 48/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0525 - accuracy: 0.9812
Epoch 49/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0515 - accuracy: 0.9833
Epoch 50/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0675 - accuracy: 0.9778
Epoch 51/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0581 - accuracy: 0.9813
Epoch 52/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0447 - accuracy: 0.9848
Epoch 53/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0588 - accuracy: 0.9790
Epoch 54/100
425/425 [==============================] - 12s 28ms/step - loss:
0.0666 - accuracy: 0.9774
Epoch 55/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0658 - accuracy: 0.9787
```

```
Epoch 56/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0533 - accuracy: 0.9822
Epoch 57/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0422 - accuracy: 0.9857
Epoch 58/100
425/425 [==============================] - 12s 28ms/step - loss:
0.0508 - accuracy: 0.9832
Epoch 59/100
425/425 [==============================] - 12s 28ms/step - loss:
0.0418 - accuracy: 0.9864
Epoch 60/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0305 - accuracy: 0.9881
Epoch 61/100
425/425 [==============================] - 12s 28ms/step - loss:
0.0519 - accuracy: 0.9818
Epoch 62/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0506 - accuracy: 0.9831
Epoch 63/100
425/425 [==============================] - 13s 31ms/step - loss:
0.0539 - accuracy: 0.9829
Epoch 64/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0580 - accuracy: 0.9826
Epoch 65/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0480 - accuracy: 0.9838
Epoch 66/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0330 - accuracy: 0.9873
Epoch 67/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0322 - accuracy: 0.9884
Epoch 68/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0463 - accuracy: 0.9832
Epoch 69/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0385 - accuracy: 0.9858
Epoch 70/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0411 - accuracy: 0.9858
Epoch 71/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0360 - accuracy: 0.9865
Epoch 72/100
```

```
425/425 [==============================] - 12s 29ms/step - loss:
0.0295 - accuracy: 0.9895
Epoch 73/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0357 - accuracy: 0.9873
Epoch 74/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0468 - accuracy: 0.9842
Epoch 75/100
425/425 [==============================] - 12s 28ms/step - loss:
0.0398 - accuracy: 0.9866
Epoch 76/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0353 - accuracy: 0.9876
Epoch 77/100
425/425 [==============================] - 12s 28ms/step - loss:
0.0455 - accuracy: 0.9847
Epoch 78/100
425/425 [==============================] - 12s 28ms/step - loss:
0.0248 - accuracy: 0.9901
Epoch 79/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0256 - accuracy: 0.9900
Epoch 80/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0282 - accuracy: 0.9886
Epoch 81/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0303 - accuracy: 0.9881
Epoch 82/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0211 - accuracy: 0.9916
Epoch 83/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0284 - accuracy: 0.9898
Epoch 84/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0400 - accuracy: 0.9856
Epoch 85/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0350 - accuracy: 0.9888
Epoch 86/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0330 - accuracy: 0.9883
Epoch 87/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0432 - accuracy: 0.9856
Epoch 88/100
425/425 [==============================] - 12s 29ms/step - loss:
```

```
0.0283 - accuracy: 0.9892
Epoch 89/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0139 - accuracy: 0.9940
Epoch 90/100
425/425 [==============================] - 12s 28ms/step - loss:
0.0251 - accuracy: 0.9906
Epoch 91/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0256 - accuracy: 0.9902
Epoch 92/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0312 - accuracy: 0.9890
Epoch 93/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0202 - accuracy: 0.9926
Epoch 94/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0386 - accuracy: 0.9864
Epoch 95/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0320 - accuracy: 0.9886
Epoch 96/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0206 - accuracy: 0.9925
Epoch 97/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0226 - accuracy: 0.9918
Epoch 98/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0245 - accuracy: 0.9914
Epoch 99/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0265 - accuracy: 0.9904
Epoch 100/100
425/425 [==============================] - 12s 29ms/step - loss:
0.0216 - accuracy: 0.9908

<keras.callbacks.History at 0x7700d47fc8b0>

preds = model.evaluate(test_x, test_y)
print ("Loss = " + str(preds[0]))
print ("Test Accuracy = " + str(preds[1]))

23/23 [==============================] - 1s 17ms/step - loss: 2.9854 -
accuracy: 0.6042
Loss = 2.985416889190674
Test Accuracy = 0.6041958332061768
```

```
from matplotlib.pyplot import imread
img_path = 'imagenet-object-localization-challenge/ILSVRC/Data/CLS-LOC/train2/n01518878/n01518878_2.JPEG'

img = image.load_img(img_path, target_size=(64, 64))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)
print('Input image shape:', x.shape)
my_image = imread(img_path)
imshow(my_image)
print(model.predict(x))
```

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing import image  # Correct import
from tensorflow.keras.applications.resnet50 import preprocess_input

img_path = 'imagenet-object-localization-challenge/ILSVRC/Data/CLS-LOC/train2/n01518878/n01518878_2.JPEG'

# Load and preprocess image (Keras way)
img = image.load_img(img_path, target_size=(64, 64))  # Now works!
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)  # Normalize for ResNet
print('Input image shape:', x.shape)

# Display the image (using matplotlib)
my_image = plt.imread(img_path)  # Alternative: Use Keras' img directly
plt.imshow(my_image)
plt.axis('off')
plt.show()

# Predict
print(model.predict(x))
```

```
Input image shape: (1, 64, 64, 3)
```

```
1/1 [==============================] - 1s 642ms/step
[[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.applications.resnet50 import preprocess_input

img_path = 'imagenet-object-localization-challenge/ILSVRC/Data/CLS-
LOC/train2/n01440764/n01440764_18.JPEG'

# Load image (Matplotlib way)
my_image = plt.imread(img_path)
plt.imshow(my_image)
plt.axis('off')
plt.show()

# Manually preprocess to match Keras expectations
from PIL import Image
img = Image.fromarray(my_image).resize((64, 64))  # Resize
x = np.array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)  # Normalize
print('Input image shape:', x.shape)
print(model.predict(x))
```

```
Input image shape: (1, 64, 64, 3)
1/1 [==============================] - 0s 15ms/step
[[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```