03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

**Transfer Learning**

**So what exactly transfer learning?**

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 1/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

**Simply said , transfer learning is the phenonmenon which allows you to transfer what state of the art machine learning models have learnt, and you use it for your custom problem.**

from IPython.display import YouTubeVideo YouTubeVideo('yofjFQddwHE', width=600, height=300)

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 2/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

[Transfer Learning (C3W2L07)](https://www.youtube.com/watch?v=yofjFQddwHE)

So transfer learning is kindoff a method where a model developed for a task is reused as the starting point for a model on a second task. It is actually a very popular approach in deep learning where pre-trained models (models which are already trained) are used as the starting point on computer vision and natural processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems.

A below mentioned owchart represents how basically transfer learning works pracitce. Deep learning model 1 transfers the knowledge it learnt(weight, biases) which can be then used by the deep learning model 2.

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 3/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

**Okayy! But here the question comes that why do we need transfer learning. Why can't we train our own neural network from Scratch?**

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 4/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 5/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

The Answer to that is most deep learning networks that are used for real world problems, learn around millions of parameters to give a high accuracy.

Now let's say i am working on the problem of face detection. Sure I can train my deep learning model from scratch for this.

But here's the catch:

**Any image classi cation network will learn very similar weights and features in order to identify a face.**

So why not use what state of the art models have learnt and use it for our problem?

Models that have been trained on extensive data sets like COCO or ImageNet, are in general good at recognizing objects in images. So it only makes sense to directly use what they have learnt and ne tune it to our purpose.

**So in short, transfer learning allows us to reduce massive time and space complexity by using what other state-of-the-art models have**

For the sake of understanding **Transfer learning** we are going to use **Resnet**

Let's see what exactly resnet is?

So over the years, the deep learning architectures became deeper and deeper i.e, adding more layers, to solve more and more complex tasks which also helped in improving the performance of classi cation and recognition tasks and also making them robust.

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 6/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

What is ResNet?

Research paper: <https://arxiv.org/pdf/1512.03385.pdf>

NOTE: Read the reseach paper properly and if you are not able to understand some concepts mentioned in the paper then refer the video mentioned in the below link.

from IPython.display import YouTubeVideo YouTubeVideo('GWt6Fu05voI', width=600, height=300)

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 7/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

[[Classic] Deep Residual Learning for Image Recognition (Paper …](https://www.youtube.com/watch?v=GWt6Fu05voI)

Resnet stands for Residual Network. It is one of the famous deep learning models that was introduced by Shaoqing Ren, Kaiming He, Jian Sun, and Xiangyu Zhang in their paper. The paper was named **"Deep Residual Learning for Image Recognition"** in 2015. The ResNet model is one of the popular and most successful deep learning models so far.

Deep residual networks like the popular ResNet-50 model is a convolution neural network(CNN) that is 50 layers deep. A residual neural network (ResNet) is an arti cial neural network (ANN) of a kind that stacks residual blocks on top of each other to form a network.

When ResNet was rst introduced, it was revolutionary for proving a new solution to a huge problem for deep neural networks at the time: the vanishing gradient problem. Although neural networks are universal function approximators, at a certain threshold adding more layers makes training become slower and makes the accuracy saturate.

This is due to the backpropagation of gradients as it goes from the nal layers to the earliest ones — multiplying a number between 0 and 1 many times makes it increasingly smaller: thus the gradient begins to “disappear” when reaching the earlier layers. That means the earlier layers are not only slower to train but are also more prone to error. That’s a huge problem as the earliest layers are the building blocks of the whole network — they are responsible for identifying the basic, core features!

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 8/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

To mitigate this problem, ResNet incorporates identity shortcut connections which essentially skip the training of one or more layers — creating

**Residual Learning: a building block**

The authors then proposed an “optimized” residual block, adding an extension called a bottleneck. It would reduce the dimensionality in the

rst two CONV layers (1/4 of the lters learned in the nal CONV layer) and then increase again during the nal CONV layer. Here are two residual modules stacked on top of each other.

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 9/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

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).

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 10/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

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.

What are Skip Connections in ResNet?

These skip connections work in two ways. Firstly, they alleviate the issue of vanishing gradient by setting up an alternate shortcut for the gradient to pass through. In addition, they enable the model to learn an identity function. This ensures that the higher layers of the model do not perform any worse than the lower layers.

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 11/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

In short, the residual blocks make it considerably easier for the layers to learn identity functions. As a result, ResNet improves the e ciency of deep neural networks with more neural layers while minimizing the percentage of errors. In other words, the skip connections add the outputs from previous layers to the outputs of stacked layers, making it possible to train much deeper networks than previously possible.

The overall network architecture looked like this, and our model will be similar to.

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 12/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

ResNet-34 Architecture

The rst 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 lters. However, compared to VGGNets, ResNets have fewer lters 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.

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 13/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

It also followed two simple design rules – the layers had the same number of lters for the same output feature map size, and the number of

lters 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 rst 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 modi ed 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

Deep Residual Learning for Image Recognition

In recent years, the eld of **Computer Vision** has undergone far-reaching transformations due to the introduction of the new technologies. As a direct result of these advnacements, it has become possible to computer vision models to surpass the humans in e ciently solving different problems related to image recognition, object detection, face recognition, image classi cation, etc.

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 14/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

In this regard, the introduction of deep convolutional neural networks or CNNs to solve more complicated tasks in computer vision, it comes with own set of issues. It has been observed that training the neural networks becomes more di cult with the increase in the number of added layers, and in some cases dwindles as well.

It is here that the use of ResNet assumes importance. Deeper neural networks are more di cult to train. With ResNet, it becomes possible to

surpasses the di culties of training very deep neural networks.

Bamn! you have learned all the things you need to know before implementing ResNet. Now it's time to code

As you already know that in this assignment you'll be learning the implementation of transfer learing, so for the sake of that we are going to use the resnet architecture.

Also in the upcoming assignments you will learn to implement the architectures completely from scratch.

Now, before getting into code let's just understand how to use transfer learning?

How to Use Transfer Learning?

Two common approaches are as follows:

1. Develope Model Approach 2. Pre-trained Model Approach

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 15/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

Develop Model Approach

1. **Select Source Task:** You must select a related predictive modeling problem with an abundance of data where there is some relationship in the input data, output data, and/or concepts learned during the mapping from input to output data.

2. **Develop Source Model:** Next, you must develop a skillful model for this rst task. The model must be better than a naive model to ensure that some feature learning has been performed.

3. **Reuse Model:** The model t on the source task can then be used as the starting point for a model on the second task of interest. This may involve using all or parts of the model, depending on the modeling technique used.

4. **Tune Model:** Optionally, the model may need to be adapted or re ned on the input-output pair data available for the task of interest.

Pre-trained Model Approach

from IPython.display import YouTubeVideo YouTubeVideo('shl9dz5ExXM', width=600, height=300)

[Pretrained Models | When And Why To Use Pre-trained Model | H…](https://www.youtube.com/watch?v=shl9dz5ExXM)

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 16/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

1. **Select Source Model:** A pre-trained source model is chosen from available models. Many research institutions release models on large and challenging datasets that may be included in the pool of candidate models from which to choose from.

2. **Reuse Model:** The model pre-trained model can then be used as the starting point for a model on the second task of interest. This may involve using all or parts of the model, depending on the modeling technique used.

3. **Tune Model:** Optionally, the model may need to be adapted or re ned on the input-output pair data available for the task of interest.

NOTE: This second type of transfer learning is common in the eld of deep learning.

So now let's start implementing the **ResNet-50** model to solve a problem of **Image Classi cation**

As usual the rst step is to import the required libraries.

**Import all the required libraries**

# import matplotlib.pyplot import matplotlib.pyplot as plt

#import numpy import numpy as np

# import PIL import PIL

# import tensorflow import tensorflow as tf

# from keras.layers.core import Dense, Flatten

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 17/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

from keras.layers.core import Dense, Flatten

# from tensorflow.keras import layers from tensorflow.keras import layers

# from tensorflow.keras.models import Sequential from tensorflow.keras.models import Sequential

# from tensorflow.keras.optimizers import Adam from tensorflow.keras.optimizers import Adam

Now, the next step is to import your dataset

Use the below mentioned link for dataset

Dataset link: [https://storage.googleapis.com/download.tensor ow.org/example\_images/ ower\_photos.tgz](https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz)

**Import your dataset**

Now here we'll be working on the Tensor ow ower classi cation problem. The data set contains of 5 classes of owers, for which we will try to build a classi er.

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 18/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

**Sample images of 5 classes from tensor ow dataset**

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 19/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

# import pathlib

import pathlib

# dataset\_url = link of the dataset

dataset\_url = "https://storage.googleapis.com/download.tensorflow.org/example\_images/flower\_photos.tgz"

# data\_dir = tf.keras.utils.get\_file('flower\_photos', origin=dataset\_url, untar=True) data\_dir = tf.keras.utils.get\_file('flower\_photos', origin=dataset\_url, untar=True)

# data\_dir = pathlib.path(data\_dir) data\_dir = pathlib.Path(data\_dir)

Downloading data from <https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz> 228818944/228813984 [==============================] - 4s 0us/step

228827136/228813984 [==============================] - 4s 0us/step

# print the data\_dir

print(data\_dir)

/root/.keras/datasets/flower\_photos

Now just see the below mentioned code, and in the next cell do the same just replace **dandelion** with **roses**

dandelion = list(data\_dir.glob('dandelion/\*')) print(dandelion[0]) PIL.Image.open(str(dandelion[0]))

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 20/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

/root/.keras/datasets/flower\_photos/dandelion/19443726008\_8c9c68efa7\_m.jpg

# create a variable named roses which will be equal to list(data\_dir.glob('roses/\*'))

roses = list(data\_dir.glob('roses/\*'))

# print roses and pass 3 print(roses[3])

# use open function and pass str(roses[3]) PIL.Image.open(str(roses[3]))

/r oot/.k eras/d at asets/flower\_photos /rose s/1 56 995 090 54 \_d3 e1 252 86f\_n.jpg

**Splitting the data**

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 21/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

Now you need to split the image data into training and validation. With each epoch, our model will get trained on the training subset, while it checks its performance on the validation data at each epoch.

# take img height and img width as 180 img\_height,img\_width=180,180

# take batch size as 32 batch\_size=32

# Preprocess the training data i.e., train\_ds

train\_ds = tf.keras.preprocessing.image\_dataset\_from\_directory( data\_dir,

label\_mode='categorical', validation\_split=0.2, subset="training", seed=123,

image\_size=(img\_height, img\_width), batch\_size=batch\_size)

Found 3670 files belonging to 5 classes. Using 2936 files for training.

From above code, some things need be to be noted:

1. We have reformatted the input to a dimension of 180,180. This ensures uniformity across all images. You can change this according to your custom problem.

2. We mention the validation split as 0.2 This means that 80% of data will reserved for training while 20% will be used for validation.

3. We are keeping the batch size as 32. If you are working on a system with lower ram con guration, you can reduce the batch size further.

4. We mention subset as training, which means we are rst creating the training subset.

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 22/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

Now, we'll run the same code again to produce the validation data. The only change here is in the **subset attribute**.

# preprocess the validation data i.e., val\_ds

val\_ds = tf.keras.preprocessing.image\_dataset\_from\_directory( data\_dir,

validation\_split=0.2, label\_mode='categorical', subset="validation", seed=123,

image\_size=(img\_height, img\_width), batch\_size=batch\_size)

Found 3670 files belonging to 5 classes. Using 734 files for validation.

# create a variable named class\_names which will be equal to trian\_ds.class\_names

class\_names = train\_ds.class\_names

# print the class\_names print(class\_names)

['daisy', 'dandelion', 'roses', 'sunflowers', 'tulips']

Now, the next step is to visualize our data

**Visualize the data**

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 23/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

So for the purpose of visualization we are going to use the matplotlib library. we'll be visualizing the 6 images in our data.

# import matplotlib.pyplot

import matplotlib.pyplot as plt

# give the figsize of 10,10 plt.figure(figsize=(10, 10))

# for images, labels in train\_ds.take(1): for images, labels in train\_ds.take(1):

# for i in range (6) for i in range(6):

# ax = plt.subplot(3, 3, i + 1) ax = plt.subplot(3, 3, i + 1)

# plt.imshow(images[i].numpy().astype("uint8")) plt.imshow(images[i].numpy().astype("uint8")) # plt.axis("off")

plt.axis("off")

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 24/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

Since, we have visualized our data. Now the next step is to import the pre-trained model.

**Import the Pre-trained model**

Now this is where the power of transfer learning comes in. From **keras applications**, we can pick and select any of the state=of-the-art models and use it for our problem.

List of available models have been mentioned below.

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 25/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

**List of available models in keras**

In this assignment we are going to use ResNet-50, but the similar procedure can be used for any other model as well.

So, now let's import the model

# declaere a variable named resnet\_model which will be equal to sequential

resnet\_model = Sequential()

# use pretrained\_model which will be equal to tf.keras.applications.ResNet50(include\_top=False,input\_shape=(180,180,3),poo pretrained\_model= tf.keras.applications.ResNet50(include\_top=False,

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 26/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

input\_shape=(180,180,3), pooling='avg',classes=5, weights='imagenet')

# use for layer in prerained\_model.layers: for layer in pretrained\_model.layers:

# give layer.trainable as False layer.trainable=False

# inside resnet\_model.add pass the prertrained\_model resnet\_model.add(pretrained\_model)

Downloading data from [https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50\_weights\_tf\_dim\_or](https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5) 94773248/94765736 [==============================] - 1s 0us/step

94781440/94765736 [==============================] - 1s 0us/step

Here, we need to note few points on the basis of above code.

1. While importing the ResNet50 class, we mention include\_top=False. This ensures that we can add our own custom input and output layers according to our data.

2. We mention the weights='imagenet'. This means that the Resnet50 model will use the weights it learnt while being trained on the imagenet data.

3. Finally, we mention layer.trainable= False in the pretrained model. This ensures that the model does not learn the weights again, saving us a lot of time and space complexity.

Now, that we have imported a pre-trained model, we will also add a fully connected and output layer where actual learning will take place.

So in the input layer we need to use the relu activation function with 512 neurons and in the output layer we need to use the softmax activation function and we are having 5 output neurons corresponding to the 5 classes in our data.

# add flatten

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 27/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

resnet\_model.add(Flatten())

# add a Dense layer of 512 neurons and give activation as 'relu' resnet\_model.add(Dense(512, activation='relu'))

# add a dense layer of 5 neurons and give activation as 'softmax' resnet\_model.add(Dense(5, activation='softmax'))

Now, inorder to look at your model architecture, just call the summary attribute.

# call the summary attribute of the model

resnet\_model.summary()

Model: "sequential" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param # =================================================================

resnet50 (Functional) (None, 2048) 23587712

flatten (Flatten)

dense (Dense)

dense\_1 (Dense)

(None, 2048)

(None, 512)

(None, 5)

0

1049088

2565

================================================================= Total params: 24,639,365

Trainable params: 1,051,653

Non-trainable params: 23,587,712 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Model Summary for ResNet-50**

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 28/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

**NOTE: One key point which needs to get noted is that the total number of parameters in the ResNet50 model is 24 million. But the trainable parameters are only one million.**

That is precisely how transfer learning saves us massive time,space and computational complexity.

Now that your model is ready you simply compile it and train it over 10 epochs for now.

# compile the model with Adam(lr=0.001) optimizer, use categorical\_crossentropy as loss, and metrics will be equal to accu

resnet\_model.compile(optimizer=Adam(lr=0.001),loss='categorical\_crossentropy',metrics=['accuracy'])

/usr/local/lib/python3.7/dist-packages/keras/optimizer\_v2/adam.py:105: UserWarning: The `lr` argument is deprecated, super(Adam, self).\_\_init\_\_(name, \*\*kwargs)

# train upto 10 epochs epochs=10

# fit the model in history variable and inside that pass train\_ds, validation=val\_ds,epochs=epochs history = resnet\_model.fit(

train\_ds, validation\_data=val\_ds, epochs=epochs

)

Epoch 1/10

92/92 [==============================] - 357s 4s/step - loss: 0.7292 - accuracy: 0.7650 - val\_loss: 0.4028 - val\_acc Epoch 2/10

92/92 [==============================] - 355s 4s/step - loss: 0.2920 - accuracy: 0.8934 - val\_loss: 0.4241 - val\_acc Epoch 3/10

92/92 [==============================] - 356s 4s/step - loss: 0.1824 - accuracy: 0.9349 - val\_loss: 0.3560 - val\_acc Epoch 4/10

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 29/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

92/92 [==============================] - 358s 4s/step - loss: 0.0953 - accuracy: 0.9697 - val\_loss: 0.4127 - val\_acc Epoch 5/10

92/92 [==============================] - 359s 4s/step - loss: 0.0655 - accuracy: 0.9813 - val\_loss: 0.3664 - val\_acc Epoch 6/10

92/92 [==============================] - 358s 4s/step - loss: 0.0313 - accuracy: 0.9935 - val\_loss: 0.3900 - val\_acc Epoch 7/10

92/92 [==============================] - 359s 4s/step - loss: 0.0150 - accuracy: 0.9990 - val\_loss: 0.4271 - val\_acc Epoch 8/10

92/92 [==============================] - 359s 4s/step - loss: 0.0076 - accuracy: 1.0000 - val\_loss: 0.4098 - val\_acc Epoch 9/10

92/92 [==============================] - 359s 4s/step - loss: 0.0053 - accuracy: 0.9997 - val\_loss: 0.4170 - val\_acc Epoch 10/10

92/92 [==============================] - 358s 4s/step - loss: 0.0039 - accuracy: 1.0000 - val\_loss: 0.4143 - val\_acc

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 30/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

**Have Patience, training this will take some amount of time. Wait till your model gets trained upto the provided epochs.........**

So once your model is trained, the task is to move onto the next step which is evaluating the mode

**NOTE: It took around 1 hr to train this model for 10 epochs**

**Model Evaluation**

For the pupose of evaliuation we will be using matplotlib library basically to plot the train and validation accuracy with respect to each epoch. These logs had been basically stored in the variable named history during the time of training.

# create a variable named fig1 which will be equal to plt.gcf()

fig1 = plt.gcf()

# plt.plot(history.history['accuracy']) plt.plot(history.history['accuracy'])

# use history.history and pass val\_accuracy plt.plot(history.history['val\_accuracy'])

# use plt.axis and pass ymin as 0.4 and ymax as 1 plt.axis(ymin=0.4,ymax=1)

# plot the grid plt.grid()

# give title as model accuracy plt.title('Model Accuracy')

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 31/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

# name y-axis as Accuracy plt.ylabel('Accuracy')

# name x-axis as Epochs plt.xlabel('Epochs')

#inside legend list of string of train and validation plt.legend(['train', 'validation'])

# show the figure plt.show()

# inside history.history pass loss plt.plot(history.history['loss'])

# inside history.history pass Val\_loss plt.plot(history.history['val\_loss'])

# plot the grid plt.grid()

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 32/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

# give the title as Model Loss plt.title('Model Loss')

# label y-axis as Loss plt.ylabel('Loss')

# label x-axis as Epochs plt.xlabel('Epochs')

# inside legend pass list of strings of train and validation plt.legend(['train', 'validation'])

# show the figure plt.show()

The model does seem to have over t a little bit, but will be talking about the measures to prevent over tting in some other upcoming assignment.

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 33/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

For now since the validation accuracy is good enough which is something around 90%,so it's time to proceed with the next step of making

**Model Inference:**

So inorder to make predictions on any image, we simply have to run few pre-processing steps to ensure that the image dimensions are same as the one that our model was trained on. So for that purpose you need to use opencv library.

So for now you need to run the predictions on a sample image of **rose** from our data

# import cv2

import cv2

# create a variable named image and read the image using cv2 and pass str(roses[1]) image=cv2.imread(str(roses[1]))

# create a variable named image\_resized and resize the image, (image height and image width) image\_resized= cv2.resize(image, (img\_height,img\_width))

# use expand\_dims function and pass image\_resized and axis will be equal to 0 image=np.expand\_dims(image\_resized,axis=0)

# print the shape of image print(image.shape)

(1, 180, 180, 3)

Now for making predictions, you simply need to call the **predict method:**

# create a variable named pred which will be equal to predict function of resenet\_model and pass image pred=resnet\_model.predict(image)

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 34/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

However, when you try to print the predictions, you will receive an array of 5 numbers, since we used the softmax classi er.

Now, go ahead and just print the prediction.

# print the prediction

print(pred)

[[1.3909271e-06 5.5880606e-05 9.8953080e-01 2.9112573e-07 1.0411634e-02]]

Now it's time to get the output label prediction,so for that just create the variable named **output\_class** which will be equal to **class\_names[np.argmax(pred)]** and then just print the output\_class

# create a variable named output\_class which will be equal to class\_names[np.argmax(pred)]

output\_class=class\_names[np.argmax(pred)]

# print the output\_class

print("The predicted class is", output\_class)

The predicted class is roses

So nally you have got the prediction.

From this assignment you were able to see the power of transfer learning and how you can use the state of art of the deep learning models and train it on your own custom data..

**Congratulations!**

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 35/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

**You nally have completed this assignment and now it's time to take some break**

↳ *1 cell hidden*

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 36/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

**FeedBack**

We hope you’ve enjoyed this course so far. We’re committed to help you use this course to its full potential, so that you have a great learning experience. And that’s why we need your help in form of a feedback here.

Please ll this feedback form <https://docs.google.com/forms/d/e/1FAIpQLSfjBmH0yJSSA34IhSVx4h2eDMgOAeG4Dk-yHid__NMTk3Hq5g/viewform>

[Colab paid products](https://colab.research.google.com/signup?utm_source=footer&utm_medium=link&utm_campaign=footer_links) - [Cancel contracts here](https://colab.research.google.com/cancel-subscription)

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 37/38

03/09/2022, 13:18 Transfer\_learning\_withcode.ipynb - Colaboratory

https://colab.research.google.com/drive/14OtGejRYUXN1JbYit4YyaQkp7QOMY4YR?authuser=1#scrollTo=0joqXVOTXiQw&printMode=true 38/38