Transfer Learning for CNNs in Keras

Vahid Kiani

University of Bojnord

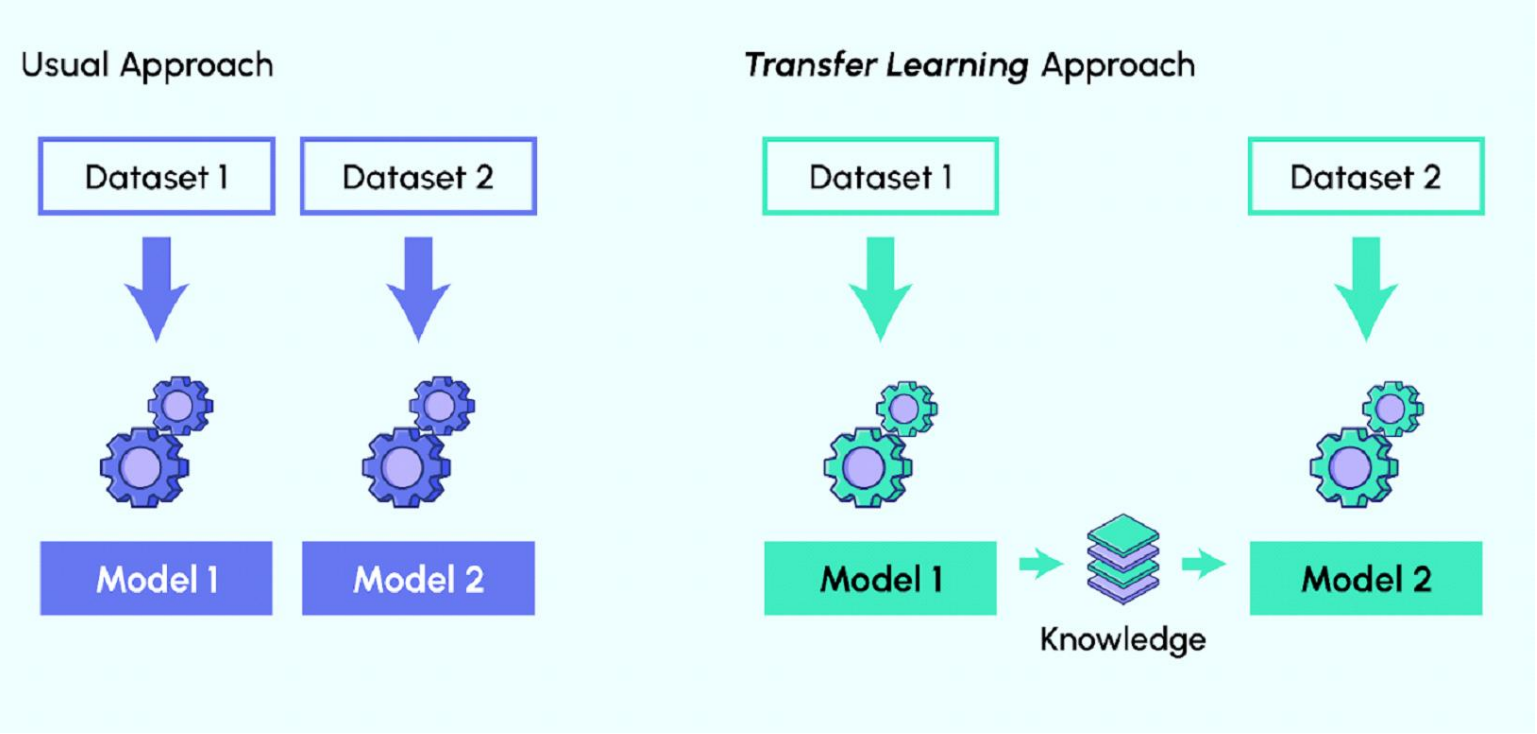
https://github.com/vkiani/deeplearning

Outline

- Transfer Learning
- VGG 16
- Mobilenet
- TF-Hub Website
- Coding in Python with TFHub
- Coding in Python with Keras Applications

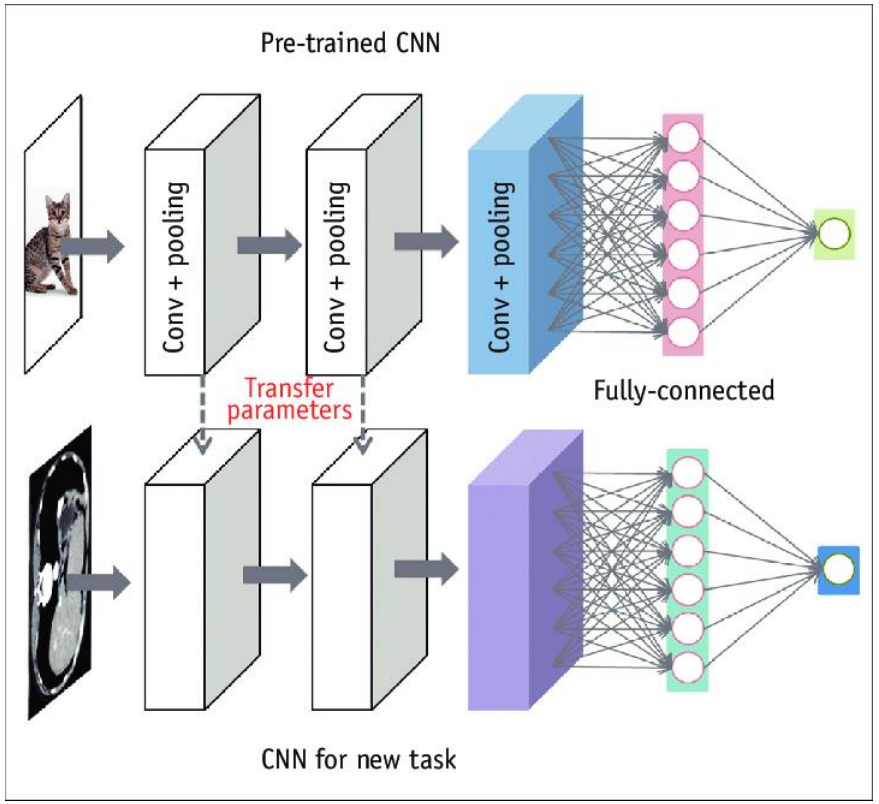
Transfer Learning

- Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned.

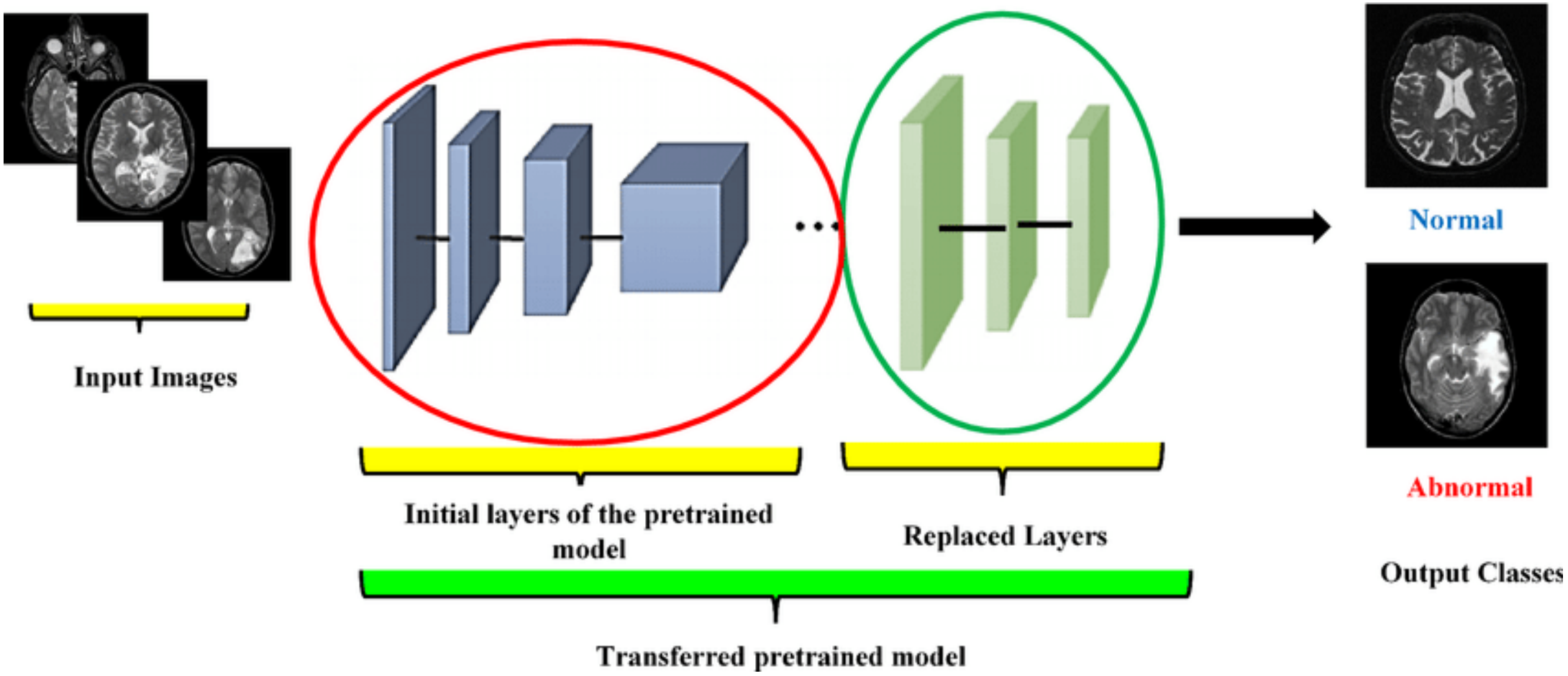


Pre-trained Model

- A pre-trained model is a saved network that was previously trained on a large dataset, typically on a large-scale image-classification task.

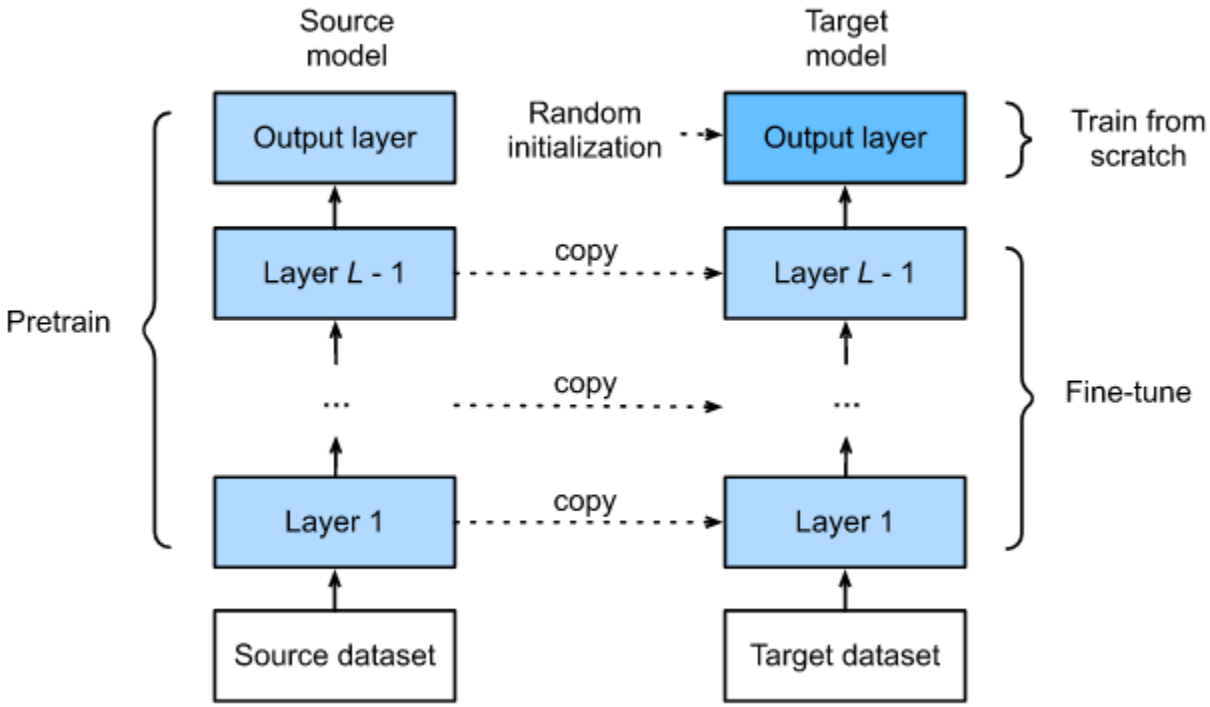


Transfer Learning



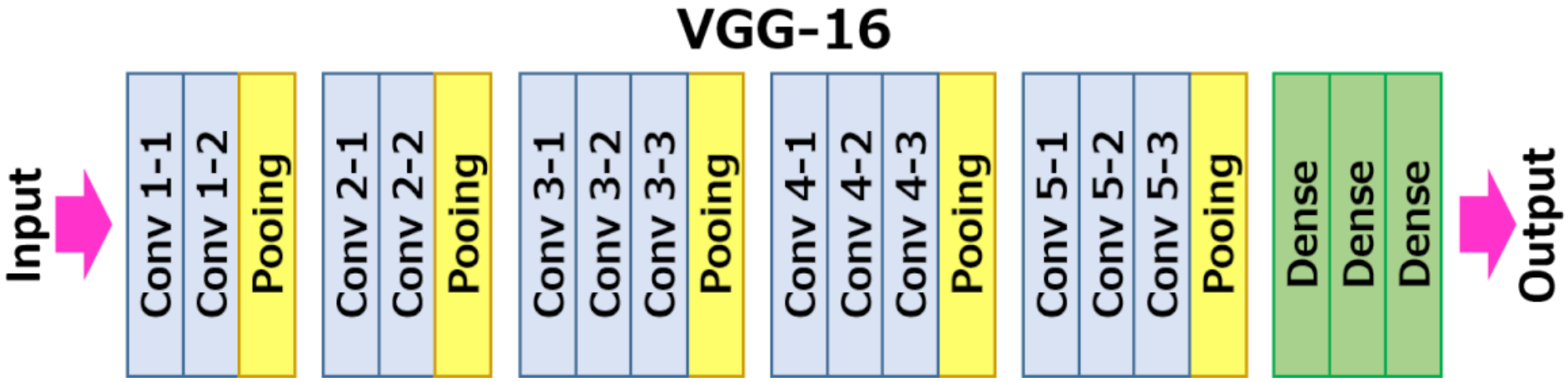
Fine-tuning

- When you use a pretrained model, you can train it on a dataset specific to your task. This is known as fine-tuning, an incredibly powerful training technique.



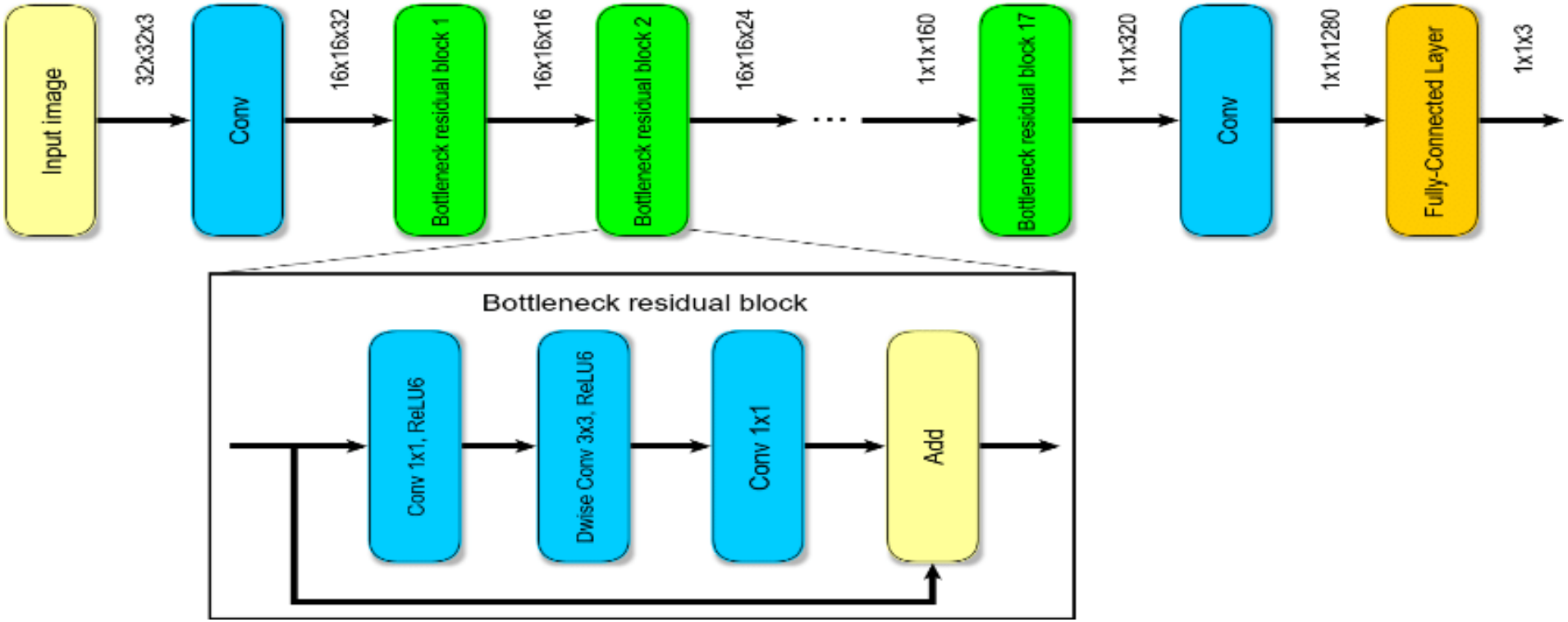
VGG-16

- 16 Layers with weights, 3x3 convolutions, 2x2 max poolings
- It can be trained on 4 GPUs for 2–3 weeks
- Achieved 74% accuracy at ImageNet Challenge (1000 classes)
- Trained by University of Oxford, 2014



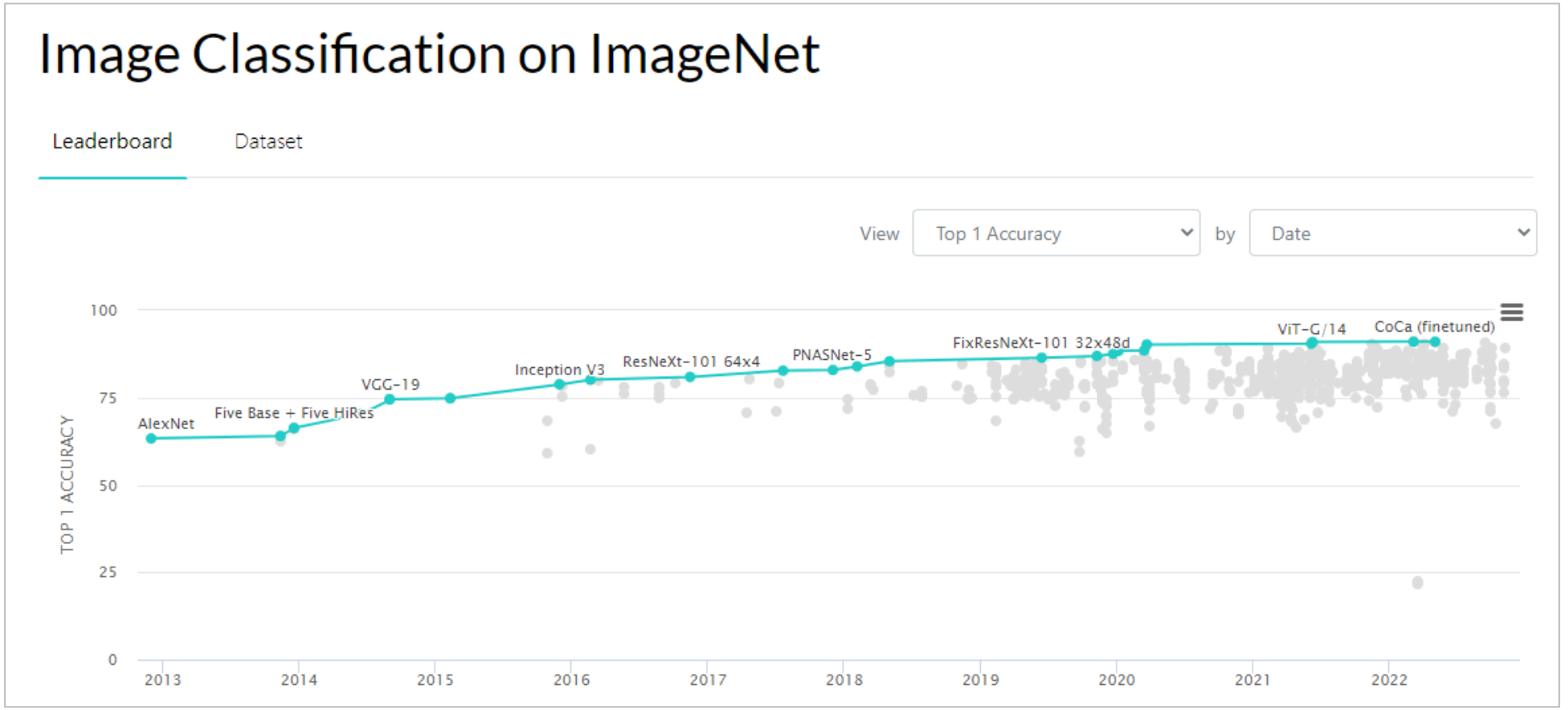
Mobilenet v2

- Trained at google, with 1.4 Million images, 1000 classes, 74% accuracy
- Performs well on mobile devices



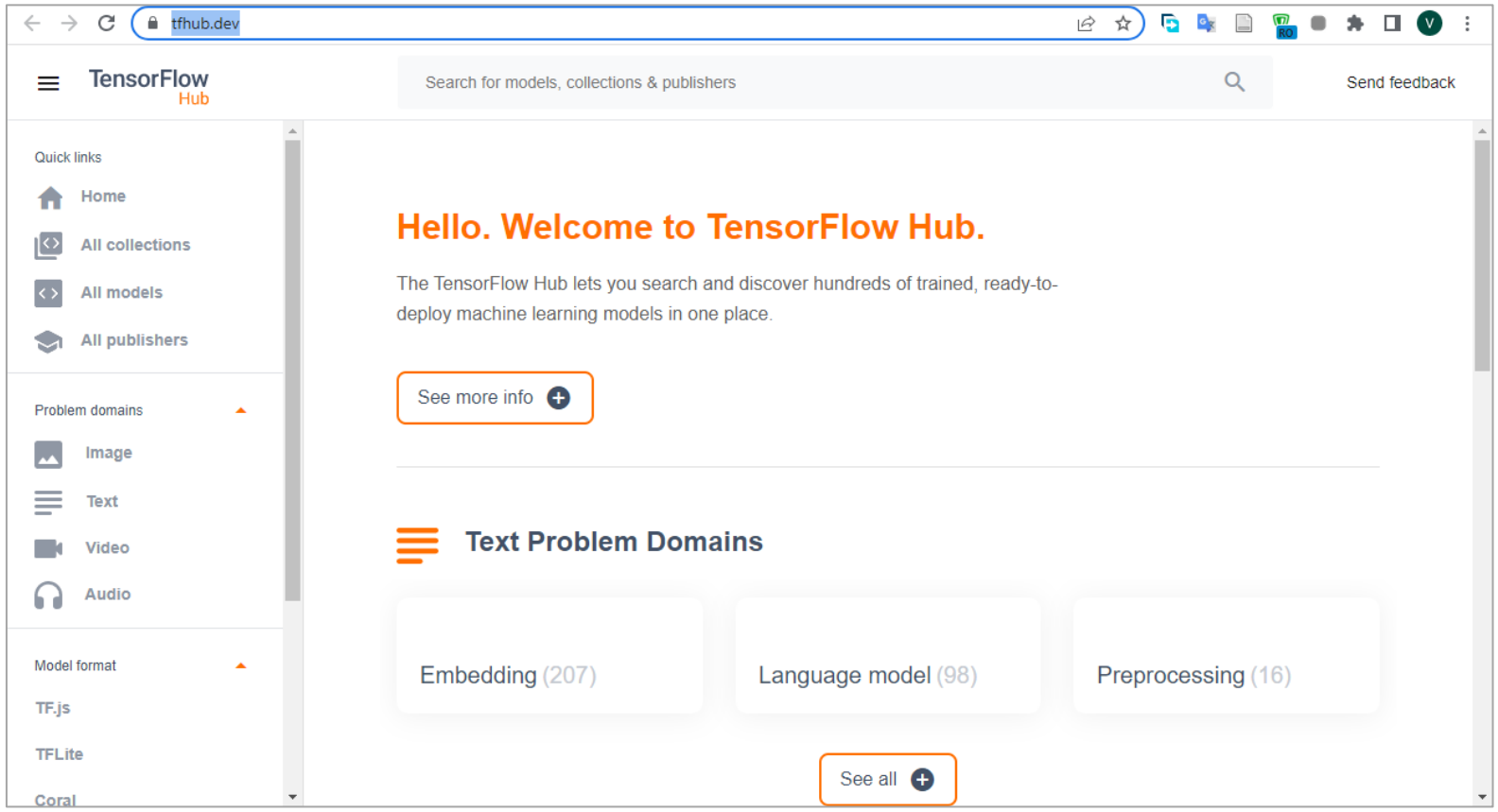
ImageNet Dataset

- 1.4 Million images, 1000 classes, entities from WordNet



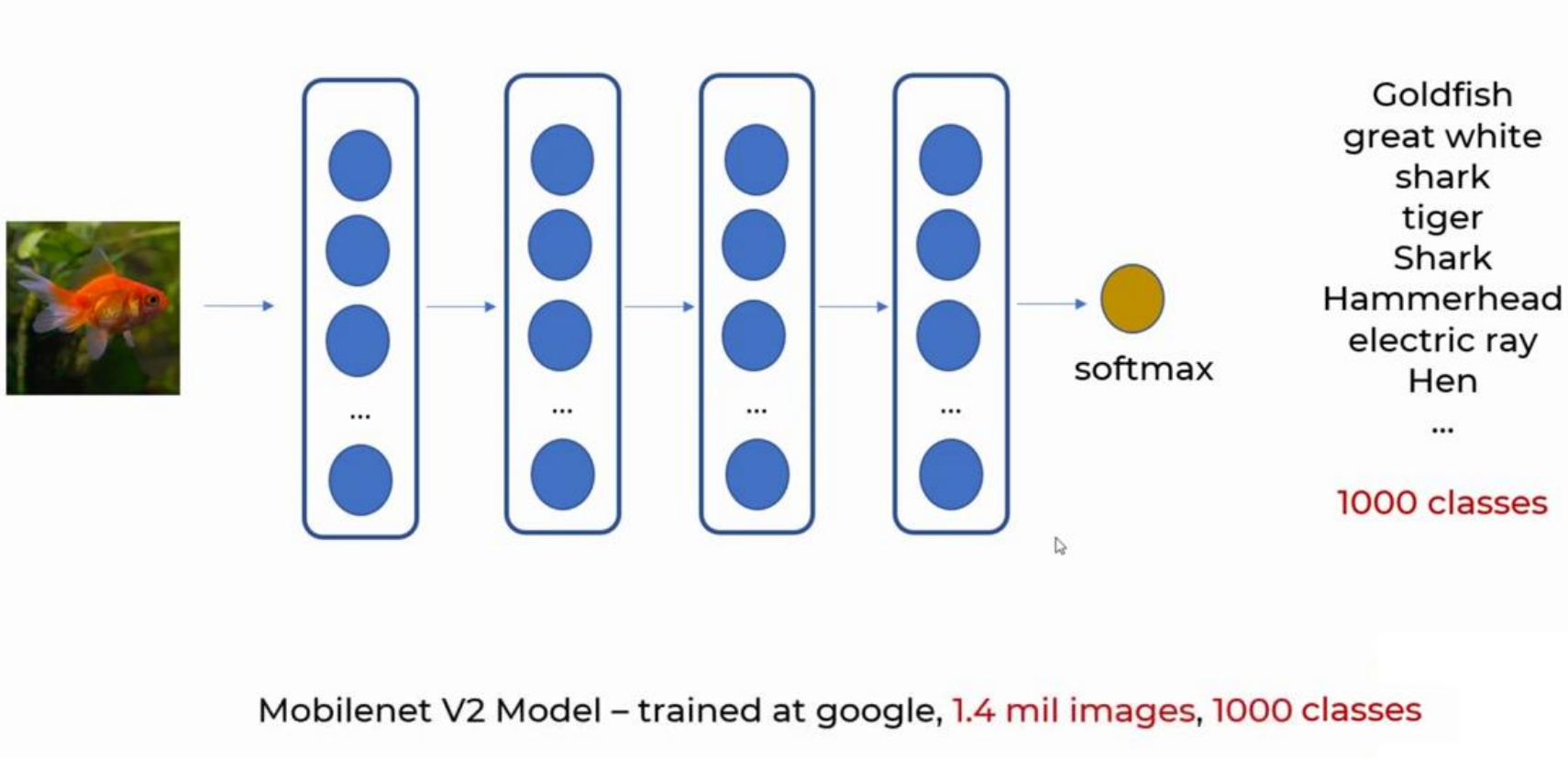
TensorFlow Hub

- https://tfhub.dev/



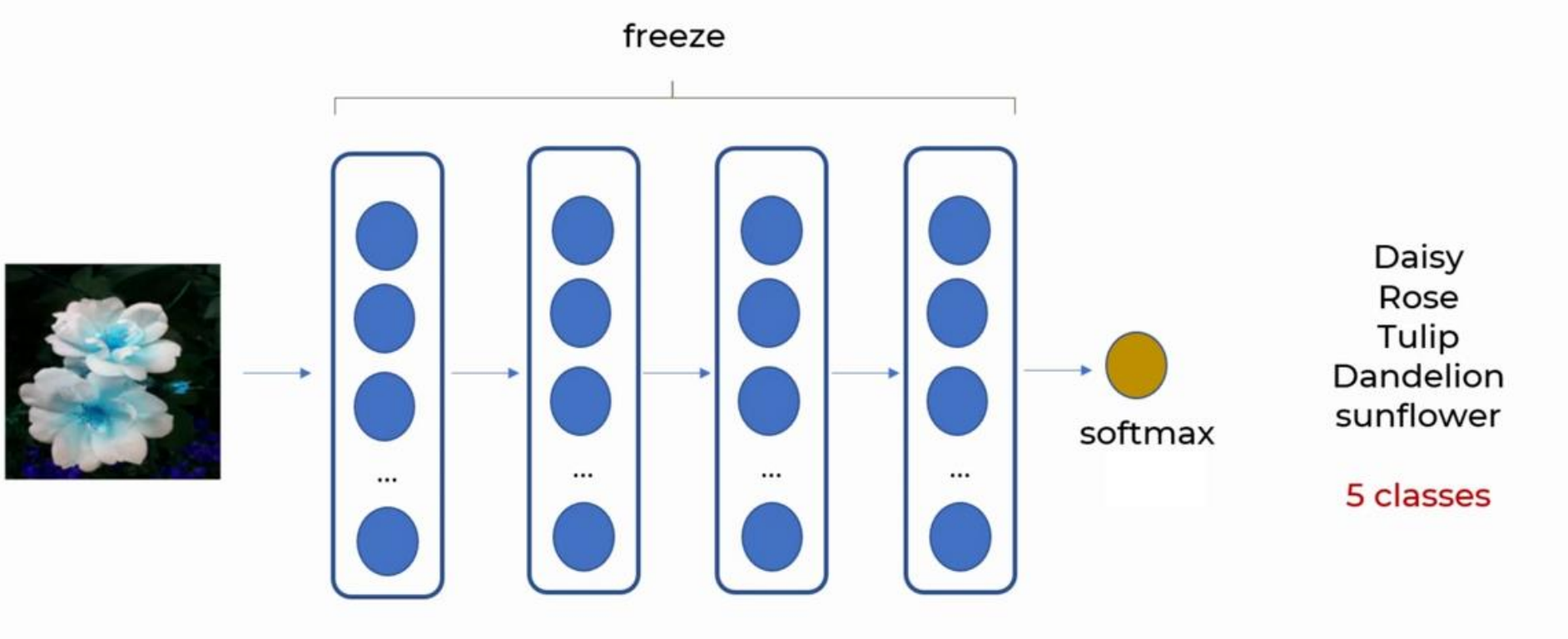
Example

- Mobilenet already trained on object detection task



Example

- We apply Mobilenet on our new task: flowers classification!



Python Example for TF-Hub

- Download flower photos dataset
- Untar the dataset
- Extract list of images from disk
- Read images from disk
- Split dataset
- Data standardization
- Load Mobilenet v2 from TF-Hub
- Create classification model

Download Flower Photos Dataset

- Download and untar the dataset

```
import tensorflow as tf

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,
                                   cache_dir='.', untar=True)
```

Extract List of Images from Disk

```
import pathlib
data_dir = pathlib.Path(data_dir)
data_dir

flowers_images_dict = {
    'roses': list(data_dir.glob('roses/*')),
    'daisy': list(data_dir.glob('daisy/*')),
    'dandelion': list(data_dir.glob('dandelion/*')),
    'sunflowers': list(data_dir.glob('sunflowers/*')),
    'tulips': list(data_dir.glob('tulips/*')),
}
```

```
flowers_labels_dict = {
    'roses': 0,
    'daisy': 1,
    'dandelion': 2,
    'sunflowers': 3,
    'tulips': 4,
}
```

```
flowers_images_dict['roses'][0]
```

Read Images from Disk

```
import cv2 as cv

x, y = [], []

for flower_name, images in flowers_images_dict.items():
    for image in images:
        img = cv.imread(str(image))
        resized_img = cv.resize(img,(224,224))
        x.append(resized_img)
        y.append(flowers_labels_dict[flower_name])
```

```
import numpy as np

x = np.array(x)
y = np.array(y)
```

Split Dataset & Data Standardization

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)
```

```
class_labels = list( flowers_labels_dict.keys() )

class_labels
```

```
x_train = x_train / 255.0
x_test = x_test / 255.0
```

Load Mobilenet v2 from TF-Hub

```
import tensorflow as tf
import tensorflow_hub as hub
from tensorflow import keras

mobile_net_address = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4"

mobile_net_layer = hub.KerasLayer(
    mobile_net_address, input_shape=(224, 224, 3), trainable=False)
```

Create Classification Model

```
num_classes = len(class_labels)

model = tf.keras.Sequential([
  mobile_net_layer,
  keras.layers.Dense(num_classes, activation='sigmoid')
])

model.summary()
```

Compile the Model & Fit

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

model.fit(x_train, y_train, epochs=10)
```

```
Epoch 5/10
81/81 [==============================] - 3s 41ms/step - loss: 0.2454 - accuracy: 0.9253
Epoch 6/10
81/81 [==============================] - 3s 42ms/step - loss: 0.2128 - accuracy: 0.9416
Epoch 7/10
81/81 [==============================] - 4s 44ms/step - loss: 0.1886 - accuracy: 0.9506
Epoch 8/10
81/81 [==============================] - 3s 42ms/step - loss: 0.1688 - accuracy: 0.9607
Epoch 9/10
81/81 [==============================] - 3s 42ms/step - loss: 0.1536 - accuracy: 0.9630
Epoch 10/10
81/81 [==============================] - 3s 42ms/step - loss: 0.1387 - accuracy: 0.9692
```

Evaluate the Model

```
model.evaluate(x_test, y_test)
```

[0.359561026096344, 0.8728428483009338]

Exercise

- Run this sample code in Google Colab with GPU
- Evaluate performance with Mobilenet V2
- Try other architectures at classification layers (last layers of model)
- Evaluate performance with fine-tunning Mobilenet V2

Python Example for Keras Applications

- Load and prepare data like previous section
- Load VGG16 from Keras Applications
- Create classification model

Load VGG16 from Keras Applications

```
from keras.applications.vgg16 import VGG16

vgg16_layer = VGG16(include_top=False, input_shape=(224, 224, 3))

for layer in vgg16_layer.layers:
  layer.trainable = False
```

Create Classification Model

```
num_classes = len(class_labels)

model = tf.keras.Sequential([
  vgg16_layer,
  keras.layers.Flatten(),
  keras.layers.Dense(128, activation='relu', kernel_initializer='he_uniform'),
  keras.layers.Dense(num_classes, activation='sigmoid')
])

model.summary()
```

Compile the Model & Fit

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

model.fit(x_train, y_train, epochs=10)
```

```
Epoch 5/10
81/81 [==============================] - 12s 152ms/step - loss: 0.0302 - accuracy: 0.9988
Epoch 6/10
81/81 [==============================] - 12s 151ms/step - loss: 0.0185 - accuracy: 1.0000
Epoch 7/10
81/81 [==============================] - 12s 153ms/step - loss: 0.0122 - accuracy: 0.9996
Epoch 8/10
81/81 [==============================] - 12s 154ms/step - loss: 0.0085 - accuracy: 1.0000
Epoch 9/10
81/81 [==============================] - 13s 156ms/step - loss: 0.0066 - accuracy: 1.0000
Epoch 10/10
81/81 [==============================] - 13s 157ms/step - loss: 0.0052 - accuracy: 1.0000
```

Evaluate the Model

```
model.evaluate(x_test, y_test)
```

[0.4765527844429016, 0.8528610467910767]

Conclusion

- Pre-trained models are complex models trained in multiple days or weeks
- Transfer learning lets us to apply these huge models on our new problems
- We can train these models in a few minutes by transfer learning
- We can fine-tune the pre-trained model on new problem domain to get better results