```
!unzip "drive/MyDrive/dog-vision/dog-breed-identification.zip" -d "drive/MyDrive/dog-vision/"

    unzip:  cannot find or open drive/MyDrive/dog-vision/dog-breed-identification.zip, drive/MyDrive/dog-vision/dog-breed-identificatio
```

Classification dog breed

1. Problem: Xác định các giống chó từ hình ảnh
2. Data: Lấy data từ Kaggle

   https://kaggle.com/c/dog-breed-identification/data
3. Evaluation: file chứa các kết quả dự đoán

   https://www.kaggle.com/competitions/dog-breed-identification/overview/evaluation
4. Feature: Một vài thông tin về data.

- Dữ liệu phi cấu trúc (hình ảnh)
- 120 kết quả output (120 giống chó)
- Bộ dữ liệu 10222 hình ảnh

```
import tensorflow as tf
import tensorflow_hub as hub
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
from IPython.display import Image
print("GPU","availabe" if tf.config.list_physical_devices("GPU") else "no")

    GPU availabe
```

Xử lý label sang matrix

1. Chuyển đổi breed
2. Chuyển đổi id

```
label_csv = pd.read_csv("drive/MyDrive/dog-vision/labels.csv")
breed = label_csv["breed"].to_numpy()
label = np.unique(breed)
boolen_breed = [boolen == label for boolen in breed]
```

```
boolen_breed[0]

    array([False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False,  True, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False, False, False, False, False, False, False,
           False, False, False])
```

```
filename = ["drive/MyDrive/dog-vision/train/" + namejpg + ".jpg" for namejpg in label_csv['id']]
```

```
Image(filename[0])
```

Check len label and image in train folder

```
len(os.listdir('/content/drive/MyDrive/dog-vision/train'))==len(filename)
```

```
True
```

Tạo bộ dữ liệu để train và validation

```
x = filename
y = boolen_breed
```

```
NUM_IMAGE = 10000 #@param {type: "slider",min: 1000, max: 10000, step: 1000}
```

NUM_IMAGE: 10000

```
from sklearn.model_selection import train_test_split
x_train, x_val , y_train ,y_val = train_test_split(x[:NUM_IMAGE],y[:NUM_IMAGE],test_size = 0.2,random_state = 40)
```

Tiền xử lý ảnh (-> numpy array) sau đó chuyển dữ liệu sang data batchs

```
#size image define: 224 x 224
IMG_SIZE = 224
BATCH_SIZE = 32
def process_image (image_path):
  image = tf.io.read_file(image_path)
  # giải mã hình ảnh theo 3 kênh màu RGB
  image = tf.image.decode_jpeg(image, channels = 3)
  # chuyển đổi từ hình ảnh sang số float
  image = tf.image.convert_image_dtype(image, tf.float32)
  #resize
  image = tf.image.resize(image, size = [IMG_SIZE,IMG_SIZE])
  return image

def get_image_label(image,label):
  image = process_image(image)
  return image,label

def data_batch(x, y = None, batch_size = BATCH_SIZE, val_data = False, test_data = False):
  if test_data:
    data = tf.data.Dataset.from_tensor_slices((tf.constant(x)))
    data_batch = data.map(process_image).batch(batch_size)
    return data_batch
  elif val_data:
    data = tf.data.Dataset.from_tensor_slices((tf.constant(x),tf.constant(y)))
    data_batch = data.map(get_image_label).batch(batch_size)
    return data_batch
  else:
    data = tf.data.Dataset.from_tensor_slices((tf.constant(x),tf.constant(y)))
    data = data.shuffle(buffer_size = len(x))
    data_batch = data.map(get_image_label).batch(batch_size)
    return data_batch
```

```
data_train = data_batch(x= x_train,y = y_train)
data_validation = data_batch(x = x_val,y = y_val,val_data = True)
```

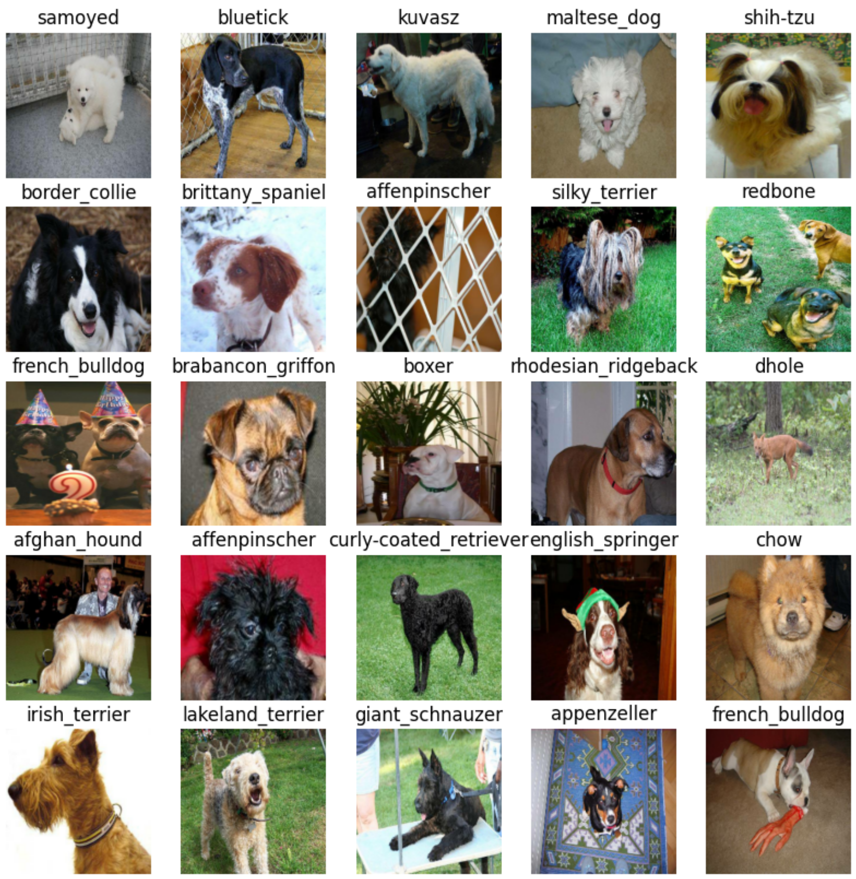
```
data_train.element_spec
```

```
(TensorSpec(shape=(None, 224, 224, 3), dtype=tf.float32, name=None),
 TensorSpec(shape=(None, 120), dtype=tf.bool, name=None))
```

Visualize data batch

```
def visualize_data_batch(data_batch):
  image,label = next(data_batch.as_numpy_iterator())
  plt.figure(figsize = (10,10))
  for i in range(25):
    ax = plt.subplot(5,5,i+1)
    plt.imshow(image[i])
    plt.title(np.unique(breed)[label[i].argmax()])
    plt.axis('off')
```

```
visualize_data_batch(data_train)
```



Chuẩn bị input, output và model

```
INPUT_SHAPE = [224, 224, 3] # thông tin trên data_train.element_spec
OUTPUT_SHAPE = 120
MODEL_URL = "https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet21k_ft1k_b0/classification/2"
def create_model (input_shape = INPUT_SHAPE, output_shape = OUTPUT_SHAPE, model_url = MODEL_URL):
  print("building model: ", model_url)
  model = tf.keras.Sequential([
        hub.KerasLayer(model_url, input_shape = input_shape),
        tf.keras.layers.Dense(units= output_shape,
                              activation = 'softmax')
        ])
  model.compile(
      loss = tf.keras.losses.CategoricalCrossentropy(),
      optimizer= tf.keras.optimizers.Adam(),
      metrics = ['accuracy']
  )
  return model
```

```
model = create_model()
model.summary()
```

```
building model:  https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet21k_ft1k_b0/classification/2
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 keras_layer (KerasLayer)    (None, 1000)              7200312   
                                                                 
 dense (Dense)               (None, 120)               120120    
                                                                 
=================================================================
Total params: 7,320,432
Trainable params: 120,120
Non-trainable params: 7,200,312
_________________________________________________________________
```

Tạo các hàm callback

```
%load_ext tensorboard
import datetime as dt
def tensorboard_callback():
  dir = os.path.join("drive/MyDrive/dog-vision/tensorboard/",dt.datetime.now().strftime("%Y%m%d-%H%M%S"))
  return tf.keras.callbacks.TensorBoard(dir)

early_stop = tf.keras.callbacks.EarlyStopping(monitor = 'val_accuracy',
                                              patience = 3)
```

```
NUM_EPOCHS = 40 #@param {type: "slider", min: 10, max: 100, step: 10}
```

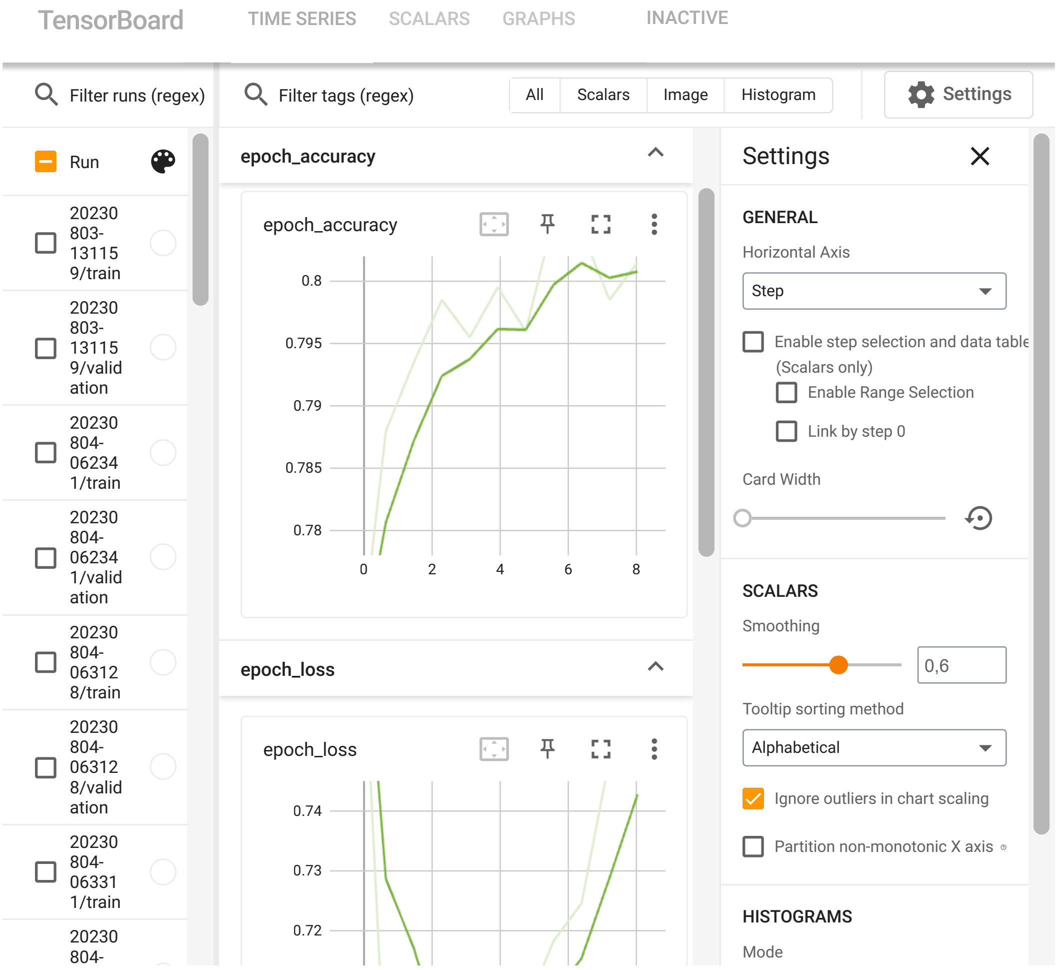
NUM_EPOCHS: 40

```
@tf.autograph.experimental.do_not_convert
def train_model():
  model = create_model()
  tensorboard = tensorboard_callback()
  model.fit(x= data_train,
            epochs = NUM_EPOCHS,
            validation_freq = 1,
            validation_data = data_validation,
            callbacks = [tensorboard,early_stop])
  return model
train_model()
```

```
building model:  https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet21k_ft1k_b0/classification/2
Epoch 1/40
250/250 [==============================] - 2216s 9s/step - loss: 1.5565 - accuracy: 0.6096 - val_loss: 0.7879 - val_accuracy: 0.768
Epoch 2/40
250/250 [==============================] - 35s 139ms/step - loss: 0.4962 - accuracy: 0.8481 - val_loss: 0.6931 - val_accuracy: 0.78
Epoch 3/40
250/250 [==============================] - 39s 157ms/step - loss: 0.3070 - accuracy: 0.9089 - val_loss: 0.7060 - val_accuracy: 0.79
Epoch 4/40
250/250 [==============================] - 34s 138ms/step - loss: 0.2046 - accuracy: 0.9445 - val_loss: 0.6830 - val_accuracy: 0.79
Epoch 5/40
250/250 [==============================] - 37s 150ms/step - loss: 0.1422 - accuracy: 0.9681 - val_loss: 0.6977 - val_accuracy: 0.79
Epoch 6/40
250/250 [==============================] - 33s 131ms/step - loss: 0.1035 - accuracy: 0.9818 - val_loss: 0.7006 - val_accuracy: 0.79
Epoch 7/40
250/250 [==============================] - 37s 149ms/step - loss: 0.0784 - accuracy: 0.9881 - val_loss: 0.7070 - val_accuracy: 0.79
Epoch 8/40
250/250 [==============================] - 32s 130ms/step - loss: 0.0615 - accuracy: 0.9931 - val_loss: 0.7183 - val_accuracy: 0.80
Epoch 9/40
250/250 [==============================] - 33s 133ms/step - loss: 0.0482 - accuracy: 0.9955 - val_loss: 0.7246 - val_accuracy: 0.80
Epoch 10/40
250/250 [==============================] - 39s 157ms/step - loss: 0.0419 - accuracy: 0.9959 - val_loss: 0.7488 - val_accuracy: 0.79
Epoch 11/40
250/250 [==============================] - 35s 139ms/step - loss: 0.0340 - accuracy: 0.9975 - val_loss: 0.7634 - val_accuracy: 0.80
<keras.engine.sequential.Sequential at 0x7a98d96ca740>
```

Đánh giá mô hình

```
%tensorboard --logdir drive/MyDrive/dog-vision/tensorboard/
```



Settings

GENERAL

Horizontal Axis: Step

☐ Enable step selection and data table (Scalars only)

☐ Enable Range Selection

☐ Link by step 0

Card Width

SCALARS

Smoothing: 0,6

Tooltip sorting method: Alphabetical

☑ Ignore outliers in chart scaling

☐ Partition non-monotonic X axis

HISTOGRAMS

Mode

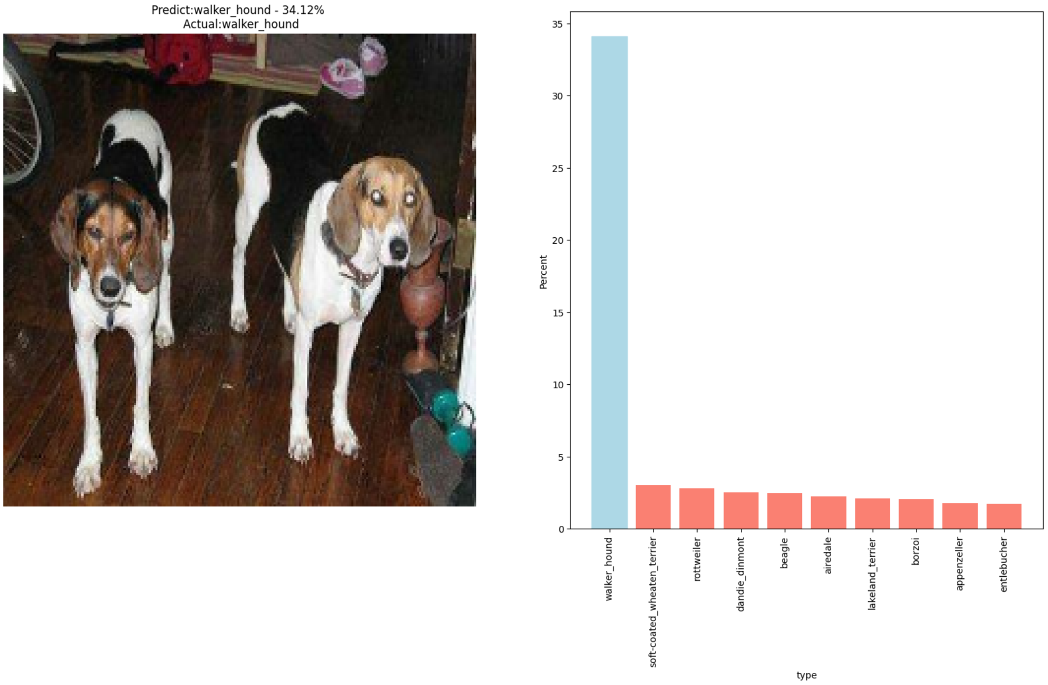
Visualize data prediction

```
predict_dog = model.predict(data_validation)
def unbatch(batch_data):
  image_unbatch = []
  label_unbatch = []
  for image,label in batch_data.unbatch().as_numpy_iterator():
    image_unbatch.append(image)
    label_unbatch.append(label)
  return image_unbatch,label_unbatch
image_unbatch,label_unbatch = unbatch(data_validation)
def visualize_data_predict(batch_data, position = 0):
  plt.figure(figsize = (20,10))
  ax = plt.subplot(1,2,1)
  plt.imshow(image_unbatch[position])
  plt.title("Predict:{} - {:0.2f}% \n Actual:{}".format(label[predict_dog[position].argmax()],
                                                  predict_dog[position].max()*100,
                                                  label[label_unbatch[position].argmax()]))

  plt.axis("off")
  ax = plt.subplot(1,2,2)
  propotion = np.argsort(predict_dog[position])[::-1]
  index_top10 = propotion[:10]
  label_top10 = label[index_top10]
  value_top10 = predict_dog[position][index_top10]*100
  plt_10 = plt.bar(x = label_top10,height = value_top10,color = "Salmon")
  plt.xticks(rotation = 90)
  plt.ylabel("Percent")
  plt.xlabel("type")
  if np.any(np.isin(label_top10,label[label_unbatch[position].argmax()])):
    plt_10[np.isin(label_top10,label[label_unbatch[position].argmax()]).argmax()].set_color("lightBlue")
def find_true(predict,actual_batch):
  arr = []
  for i in range(len(predict)):
    if label[predict[i].argmax()] == label[label_unbatch[i].argmax()]:
      arr.append(i)
  return arr
```

```
63/63 [==============================] - 9s 136ms/step
```

```
visualize_data_predict(data_validation, position = 1590)
```



Save and reload

```
def save_model(model,name = None, dir = None):
  modeldir = os.path.join(dir,dt.datetime.now().strftime("%Y%m%d-%H%M%S"))
  model_path = modeldir + '-' + name + '.hb'
  model.save(model_path)
  print("Saved in:")
  return model_path
def reload_model(model_path):
  model = tf.keras.models.load_model(model_path,custom_objects = {"Keraslayers": hub.KerasLayer})
  return model
```

```
save_model(model,name = "full_data", dir = "drive/MyDrive/dog-vision/model")
```

✓ 2 giây hoàn thành lúc 17:28 ● ×