```
[1] from google.colab import files
    uploaded = files.upload()
```

```
Choose files dataset.zip
- dataset.zip(application/x-zip-compressed) - 5030166 bytes, last modified: 24/06/2025 - 100% done
Saving dataset.zip to dataset.zip
```

```
[3] import zipfile
    import os

    # Replace 'yourfile.zip' with the name of your uploaded zip file
    with zipfile.ZipFile("dataset.zip", 'r') as zip_ref:
        zip_ref.extractall("dataset")  # or "." to extract to current directory
```

```
import zipfile
import os

# Replace this with the name of your uploaded zip file
zip_file = "dataset.zip"

with zipfile.ZipFile(zip_file, 'r') as zip_ref:
    zip_ref.extractall("unzipped")  # You can name this folder anything

# Check what's inside
os.listdir("unzipped")
```

```
['archive']
```

```
[5] import os
    os.listdir("unzipped")
```

```
['archive']
```

```
[8] os.listdir("unzipped/archive")
```

```
['Dataset']
```

```
[10] os.listdir("unzipped/archive/Dataset")
```

```
['Recyclable Images', 'Biodegradable Images', 'Trash Images']
```

[13]
```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)

train_generator = datagen.flow_from_directory(
    'unzipped/archive/Dataset',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='training'
)

val_generator = datagen.flow_from_directory(
    'unzipped/archive/Dataset',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='validation'
)
```

```
Found 312 images belonging to 3 classes.
Found 78 images belonging to 3 classes.
```

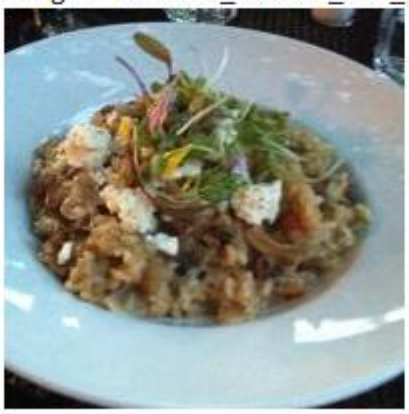
[15]
```python
import os
import shutil
import random

# Original dataset path
source_dir = "unzipped/archive/Dataset"
target_base = "dataset"
train_dir = os.path.join(target_base, "train")
val_dir = os.path.join(target_base, "val")

# Create destination directories
for folder in [train_dir, val_dir]:
    os.makedirs(folder, exist_ok=True)

# Split ratio
split_ratio = 0.8  # 80% train, 20% val

# Loop through class folders
for class_name in os.listdir(source_dir):
    class_path = os.path.join(source_dir, class_name)
    if not os.path.isdir(class_path):
        continue

    # Make class subfolders
    os.makedirs(os.path.join(train_dir, class_name), exist_ok=True)
    os.makedirs(os.path.join(val_dir, class_name), exist_ok=True)
```

```
        # Shuffle and split files
        list: val_files
        (26 items) ['TRAIN.4_NBIODEG_CCW_1895.jpg', 'TRAIN.4_NBIODEG_CCW_
        val_files = files[split_point:]

        # Move files to train
        for f in train_files:
            src = os.path.join(class_path, f)
            dst = os.path.join(train_dir, class_name, f)
            shutil.copy2(src, dst)

        # Move files to val
        for f in val_files:
            src = os.path.join(class_path, f)
            dst = os.path.join(val_dir, class_name, f)
            shutil.copy2(src, dst)

    print("✅ Dataset successfully split into 'dataset/train' and 'dataset/val'")
```

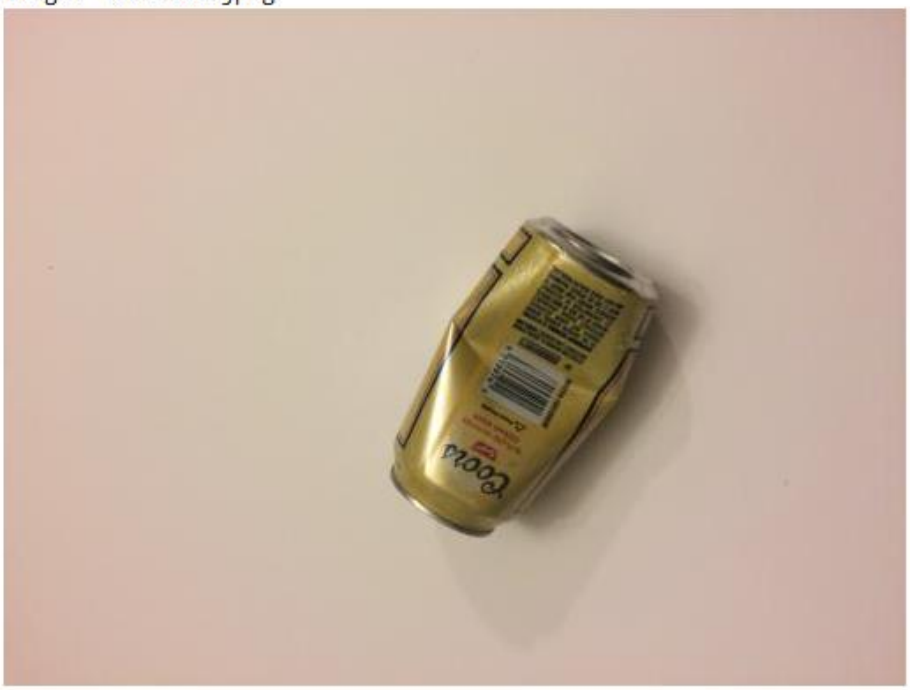
```
✅ Dataset successfully split into 'dataset/train' and 'dataset/val'
```

```
[19] import os
     import random
     from IPython.display import Image, display

     # Path to the folder containing images (adjust as needed)
     folder_path = "unzipped/archive/Dataset"

     # Choose a random class (subfolder)
     random_class = random.choice(os.listdir(folder_path))

     # Path to the chosen class folder
     class_path = os.path.join(folder_path, random_class)

     # Choose a random image from that class
     random_image = random.choice(os.listdir(class_path))

     # Full path to the image
     image_path = os.path.join(class_path, random_image)

     # Display info and the image
     print(f"Class: {random_class}")
     print(f"Image: {random_image}")
     display(Image(filename=image_path))
```

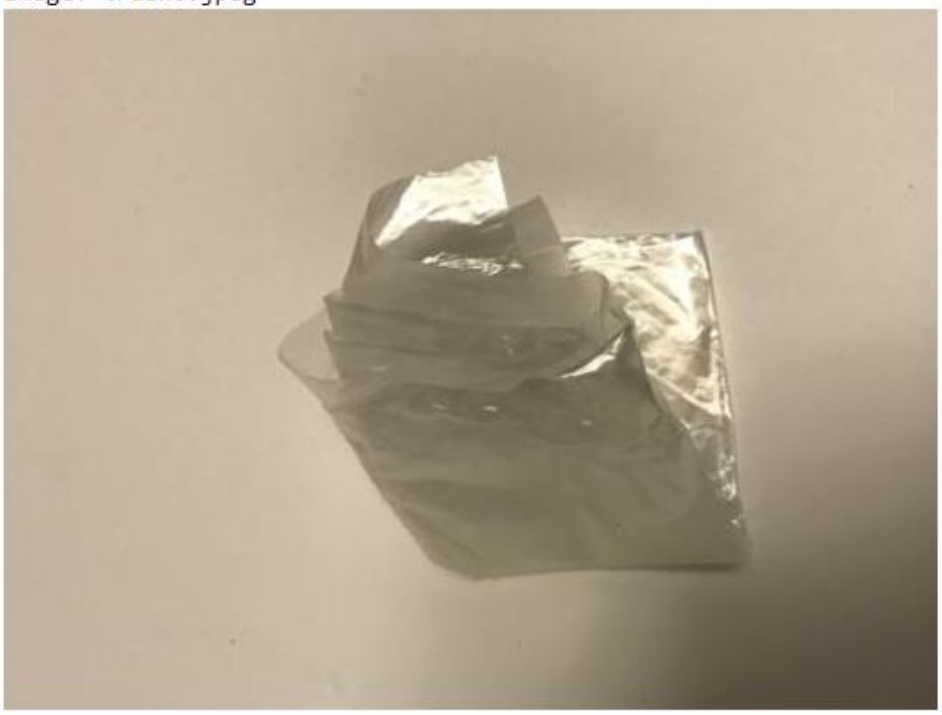
```
Class: Biodegradable Images
Image: TRAIN.2_BIODEG_ORI_11280.jpg
```



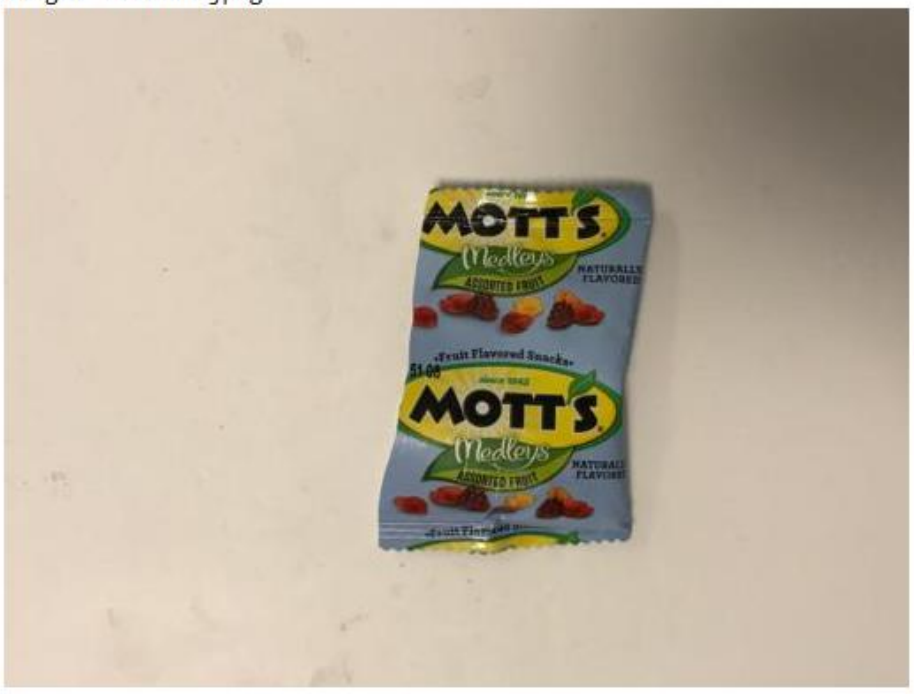
```
Class: Recyclable Images
Image: metal179.jpeg
```



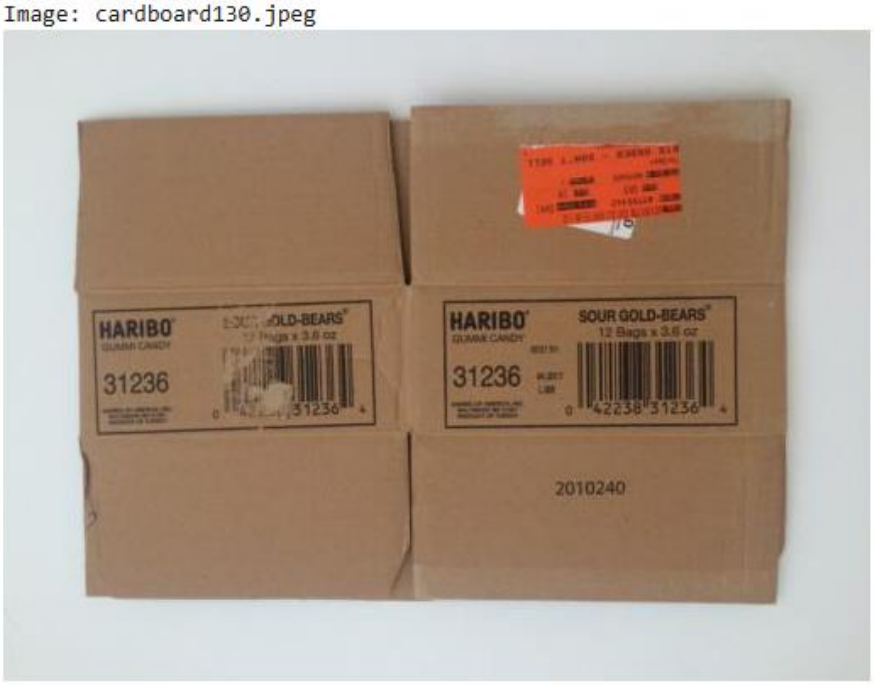
```
Class: Trash Images
Image: trash6.jpeg
```



```
Class: Trash Images
Image: trash68.jpeg
```



```
Class: Recyclable Images
Image: cardboard130.jpeg
```




```
[16] import tensorflow as tf
     from tensorflow.keras.preprocessing.image import ImageDataGenerator
     from tensorflow.keras.applications import ResNet50
     from tensorflow.keras.models import Model
     from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
     from tensorflow.keras.optimizers import Adam
     import matplotlib.pyplot as plt

     # Set up paths
     train_dir = 'dataset/train'    # e.g., dataset/train/eosinophil, dataset/train/lymphocyte...
     val_dir = 'dataset/val'        # e.g., dataset/val/...

     # Set up parameters
     img_size = (224, 224)
     batch_size = 32
     num_classes = 3  # Change this if you have a different number of classes

     # Data generators with augmentation
     train_datagen = ImageDataGenerator(
         rescale=1./255,
         zoom_range=0.2,
         horizontal_flip=True,
         rotation_range=20,
         shear_range=0.2
     )

     val_datagen = ImageDataGenerator(rescale=1./255)
```

[16]

```
train_gen = train_datagen.flow_from_directory(
    train_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical'
)

val_gen = val_datagen.flow_from_directory(
    val_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical'
)

# Load pre-trained ResNet50
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(*img_size, 3))

# Freeze base layers
for layer in base_model.layers:
    layer.trainable = False

# Add custom classification head
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dropout(0.5)(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.3)(x)
predictions = Dense(num_classes, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)
```

✓ 1m [16]

```
# Compile model
model.compile(optimizer=Adam(learning_rate=0.0001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Train model
history = model.fit(
    train_gen,
    validation_data=val_gen,
    epochs=10
)

# Save model
model.save('hematovision_model.h5')

# Plot training results
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Val Accuracy')
plt.title('Accuracy over epochs')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
Found 312 images belonging to 3 classes.
Found 78 images belonging to 3 classes.
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `supe
  self._warn_if_super_not_called()
Epoch 1/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 78s 7s/step - accuracy: 0.2977 - loss: 1.6325 - val_accuracy: 0.3333 - val_loss: 1.1350
Epoch 2/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 64s 7s/step - accuracy: 0.3067 - loss: 1.4125 - val_accuracy: 0.3462 - val_loss: 1.1525
Epoch 3/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 66s 7s/step - accuracy: 0.2772 - loss: 1.5921 - val_accuracy: 0.3333 - val_loss: 1.1403
Epoch 4/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 81s 7s/step - accuracy: 0.3586 - loss: 1.3374 - val_accuracy: 0.3846 - val_loss: 1.1207
Epoch 5/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 64s 6s/step - accuracy: 0.4007 - loss: 1.3496 - val_accuracy: 0.3333 - val_loss: 1.1150
Epoch 6/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 64s 6s/step - accuracy: 0.2988 - loss: 1.4098 - val_accuracy: 0.3462 - val_loss: 1.1090
Epoch 7/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 62s 6s/step - accuracy: 0.3241 - loss: 1.3120 - val_accuracy: 0.3462 - val_loss: 1.1099
Epoch 8/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 64s 6s/step - accuracy: 0.2909 - loss: 1.3417 - val_accuracy: 0.3718 - val_loss: 1.1011
Epoch 9/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 64s 6s/step - accuracy: 0.3822 - loss: 1.3315 - val_accuracy: 0.3333 - val_loss: 1.0971
Epoch 10/10
10/10 ━━━━━━━━━━━━━━━━━━━━ 82s 6s/step - accuracy: 0.3989 - loss: 1.2402 - val_accuracy: 0.3718 - val_loss: 1.0922
```