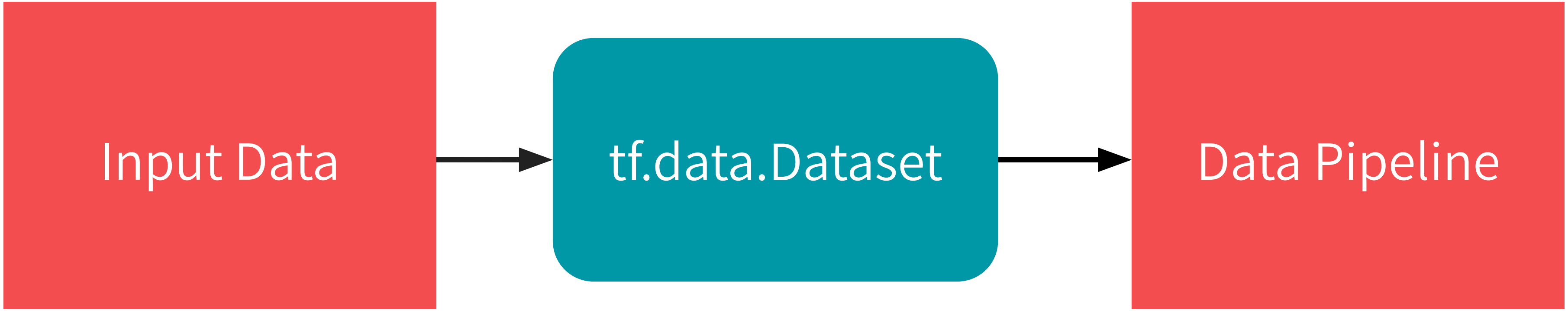
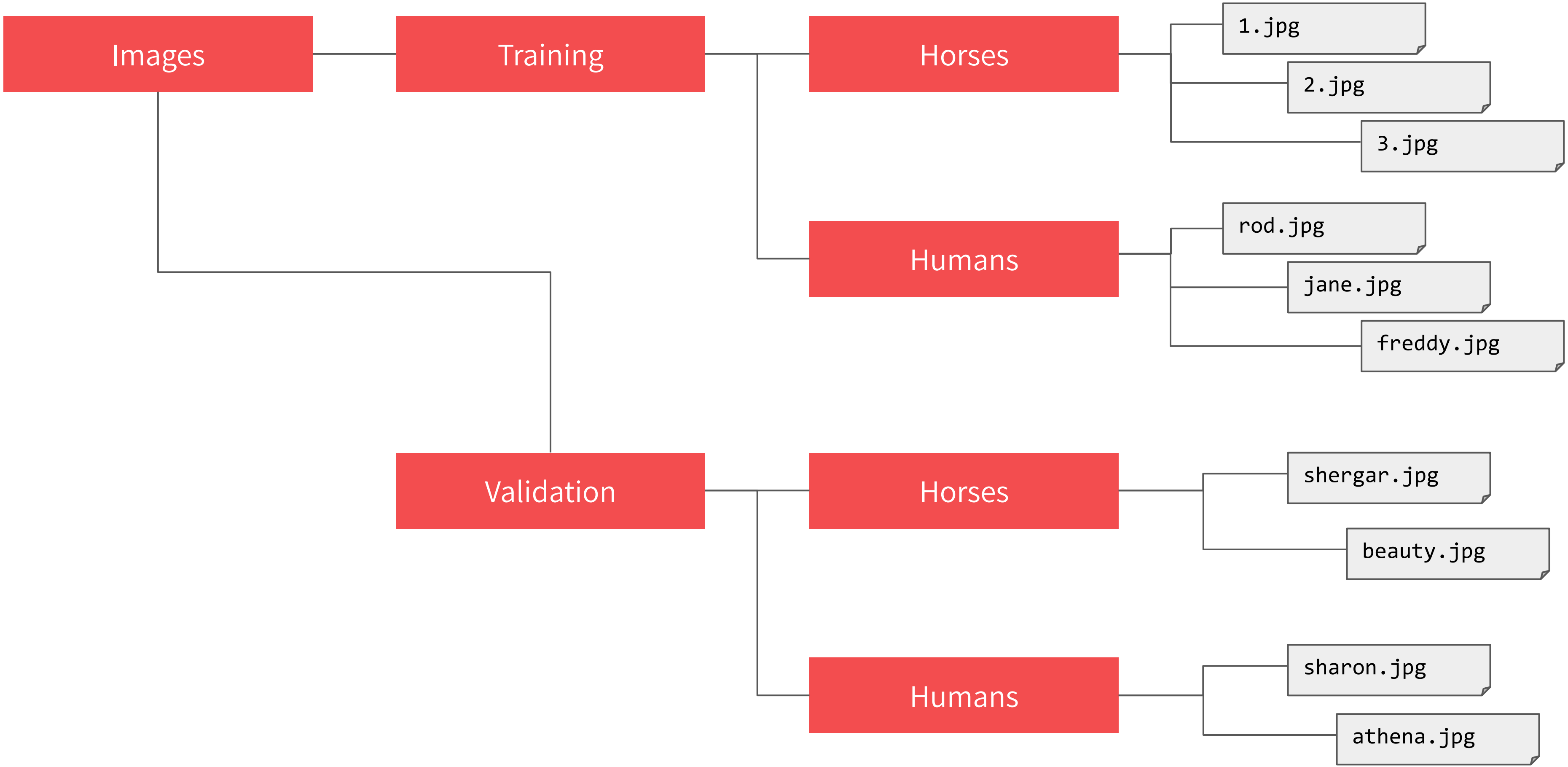
# Copyright Notice

# tf.data API



# tf.keras.utils.image_dataset_from_directory



Images

Training

Horses

1.jpg

2.jpg

3.jpg

Humans

rod.jpg

jane.jpg

freddy.jpg

Validation

Horses

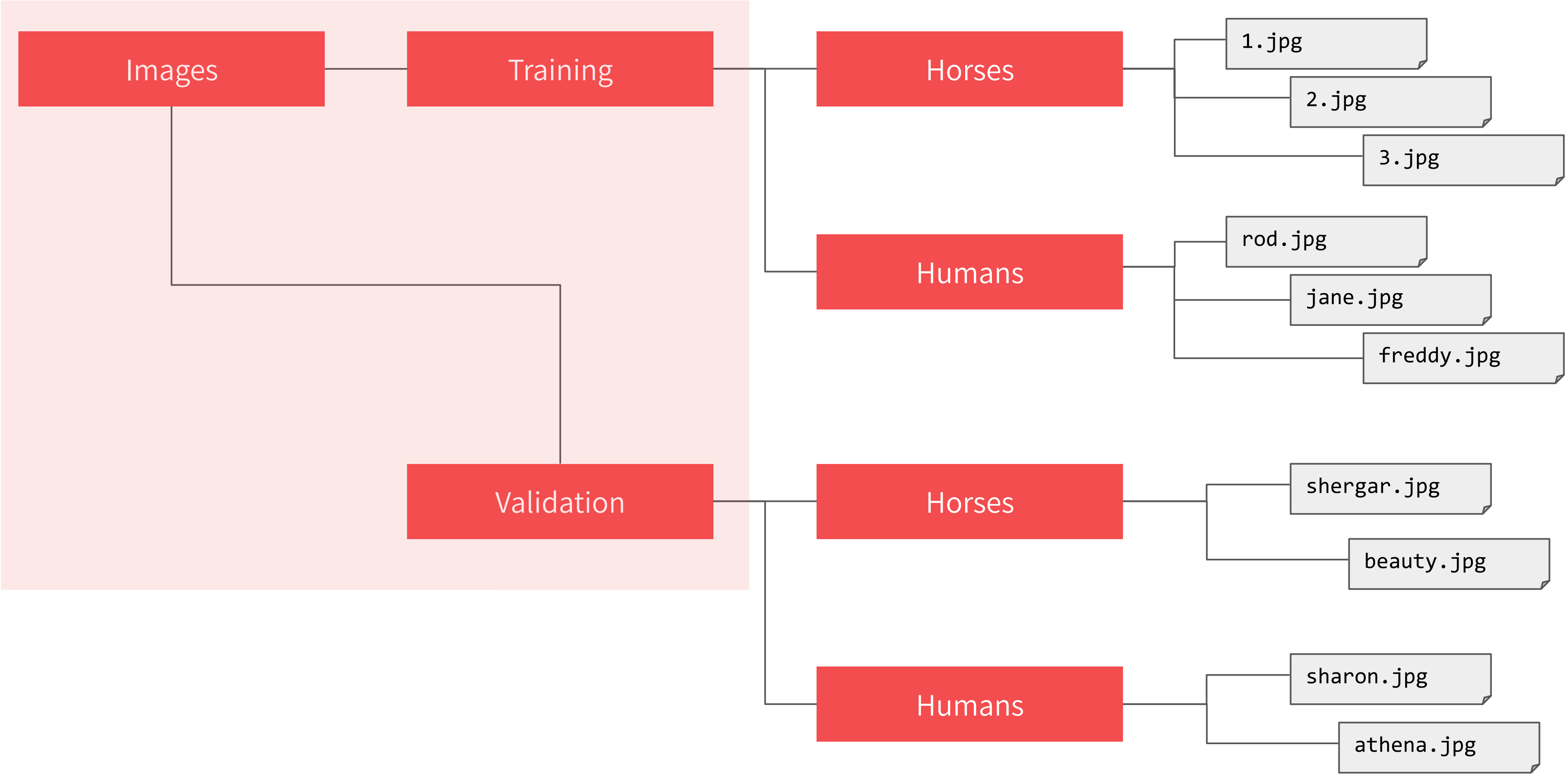
shergar.jpg

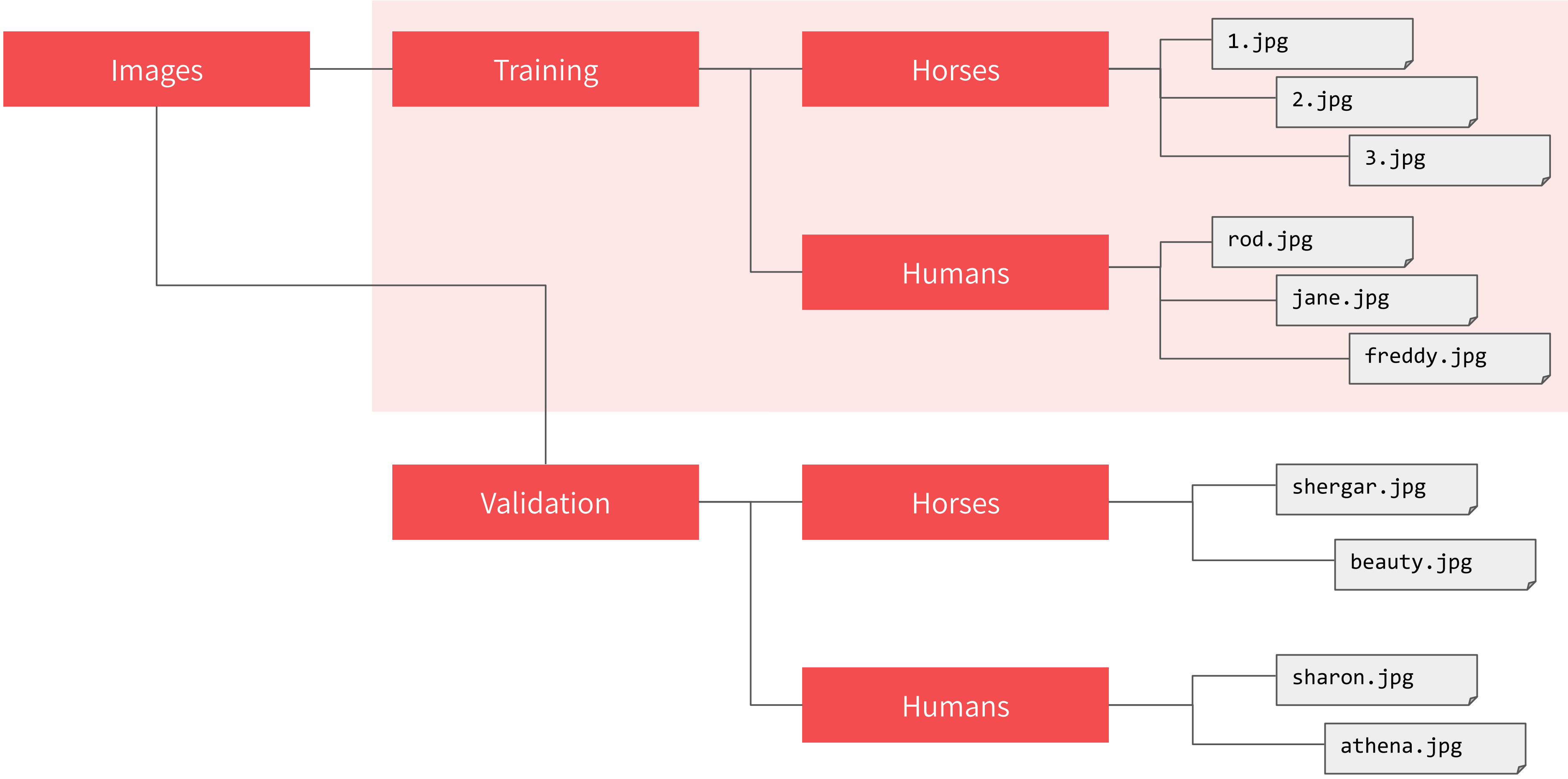
beauty.jpg

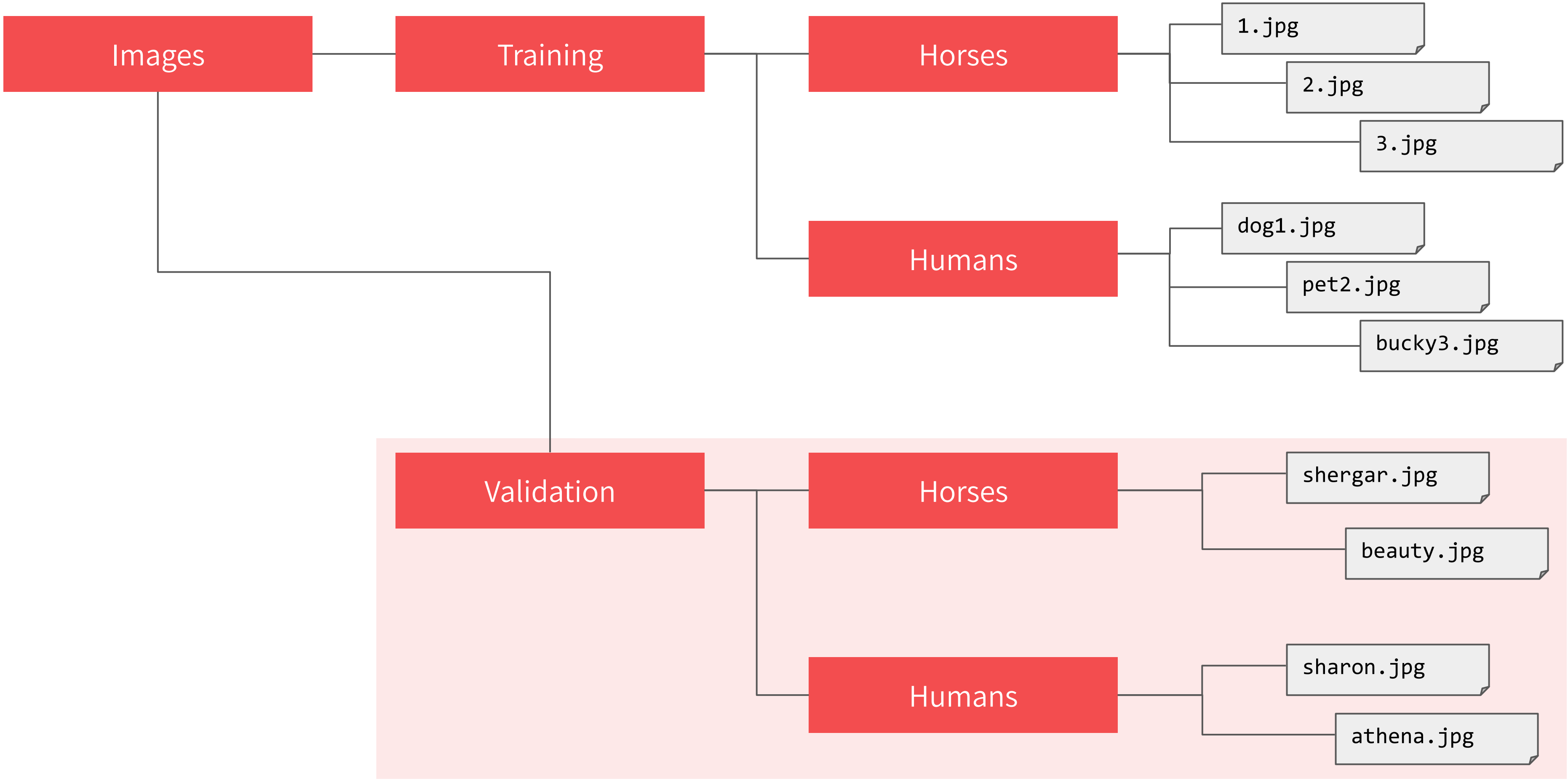
Humans

sharon.jpg

athena.jpg


Images
Training
Horses
1.jpg
2.jpg
3.jpg
Humans
rod.jpg
jane.jpg
freddy.jpg
Validation
Horses
shergar.jpg
beauty.jpg
Humans
sharon.jpg
athena.jpg


Images
Training
Horses
1.jpg
2.jpg
3.jpg
Humans
rod.jpg
jane.jpg
freddy.jpg
Validation
Horses
shergar.jpg
beauty.jpg
Humans
sharon.jpg
athena.jpg


Images
Training
Horses
1.jpg
2.jpg
3.jpg
Humans
dog1.jpg
pet2.jpg
bucky3.jpg
Validation
Horses
shergar.jpg
beauty.jpg
Humans
sharon.jpg
athena.jpg

```
train_dataset = tf.keras.utils.image_dataset_from_directory(
    TRAIN_DIR,
    image_size=(300, 300),
    batch_size=128,
    label_mode='binary'
    )
```

```python
train_dataset = tf.keras.utils.image_dataset_from_directory(
    TRAIN_DIR,
    image_size=(300, 300),
    batch_size=128,
    label_mode='binary'
    )
```

```
train_dataset = tf.keras.utils.image_dataset_from_directory(
    TRAIN_DIR,
    image_size=(300, 300),
    batch_size=128,
    label_mode='binary'
    )
```

```
train_dataset = tf.keras.utils.image_dataset_from_directory(
    TRAIN_DIR,
    image_size=(300, 300),
    batch_size=128,
    label_mode='binary'
    )
```

# tf.data.Dataset

https://www.tensorflow.org/api_docs/python/tf/data/Dataset

```
rescale_layer = tf.keras.layers.Rescaling(scale=1./255)

train_dataset_scaled = train_dataset.map(
    lambda image, label: (rescale_layer(image), label))
```

```
rescale_layer = tf.keras.layers.Rescaling(scale=1./255)

train_dataset_scaled = train_dataset.map(
    lambda image, label: (rescale_layer(image), label))
```

```
rescale_layer = tf.keras.layers.Rescaling(scale=1./255)

train_dataset_scaled = train_dataset.map(
    lambda image, label: (rescale_layer(image), label))
```

```
train_dataset_final = (train_dataset_scaled
                       .cache()
                       .shuffle(buffer_size=1000)
                       .prefetch(buffer_size=tf.data.AUTOTUNE))
```

```
train_dataset_final = (train_dataset_scaled
                       .cache()
                       .shuffle(buffer_size=1000)
                       .prefetch(buffer_size=tf.data.AUTOTUNE))
```

```
train_dataset_final = (train_dataset_scaled
                       .cache()
                       .shuffle(buffer_size=1000)
                       .prefetch(buffer_size=tf.data.AUTOTUNE))
```

```
train_dataset_final = (train_dataset_scaled
                       .cache()
                       .shuffle(buffer_size=1000)
                       .prefetch(buffer_size=tf.data.AUTOTUNE))
```

```
validation_dataset = tf.keras.utils.image_dataset_from_directory(
    VAL_DIR,
    image_size=(300, 300),
    batch_size=32,
    label_mode='binary'
    )

validation_dataset_scaled = validation_dataset.map(lambda image, label:
(rescale_layer(image), label))

# Configure the validation dataset
validation_dataset_final = (validation_dataset_scaled
                            .cache()
                            .prefetch(buffer_size=tf.data.AUTOTUNE))
```

```
validation_dataset = tf.keras.utils.image_dataset_from_directory(
    VAL_DIR,
    image_size=(300, 300),
    batch_size=128,
    label_mode='binary'
    )

validation_dataset_scaled = validation_dataset.map(lambda image, label:
(rescale_layer(image), label))

# Configure the validation dataset
validation_dataset_final = (validation_dataset_scaled
                             .cache()
                             .prefetch(buffer_size=tf.data.AUTOTUNE))
```

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Input(shape=(300, 300, 3)),
    tf.keras.layers.Conv2D(16, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Input(shape=(300, 300, 3)),
    tf.keras.layers.Conv2D(16, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

```python
model = tf.keras.models.Sequential([
    tf.keras.layers.Input(shape=(300, 300, 3)),
    tf.keras.layers.Conv2D(16, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Input(shape=(300, 300, 3)),
    tf.keras.layers.Conv2D(16, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 298, 298, 16)      448
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 149, 149, 16)      0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 147, 147, 32)      4640
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 73, 73, 32)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 71, 71, 64)        18496
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 35, 35, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 78400)             0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               40141312
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 513
=================================================================
Total params: 40,165,409
Trainable params: 40,165,409
Non-trainable params: 0
```

```
model.compile(loss='binary_crossentropy',
              optimizer=tf.keras.optimizers.RMSprop(learning_rate=0.001),
              metrics=['accuracy'])
```

https://youtu.be/zLRB4oupj6g

2.1.4 Gradient Descent in Practice II Learning Rate by Andrew Ng

```
history = model.fit(
      train_dataset_final,
      epochs=15,
      validation_data=validation_dataset_final,
      verbose=2)
```

```
history = model.fit(
      train_dataset_final,
      epochs=15,
      validation_data=validation_dataset_final,
      verbose=2)
```

```
history = model.fit(
      train_dataset_final,
      epochs=15,
      validation_data=validation_dataset_final,
      verbose=2)
```

```
history = model.fit(
      train_dataset_final,
      epochs=15,
      validation_data=validation_dataset_final,
      verbose=2)
```

```
history = model.fit(
      train_dataset_final,
      epochs=15,
      validation_data=validation_dataset_final,
      verbose=2)
```

```python
from google.colab import files

uploaded = files.upload()

for filename in uploaded.keys():

    # predicting images
    path = '/content/' + filename
    image = tf.keras.utils.load_img(path, target_size=(300, 300))
    image = tf.keras.utils.img_to_array(image)
    image = rescale_layer(image)
    image = np.expand_dims(image, axis=0)

    prediction = model.predict(image, verbose=0)[0][0]
    print(f'\nmodel output: {prediction}')

    if prediction > 0.5:
        print(filename + " is a human")
    else:
        print(filename + " is a horse")
```

```python
from google.colab import files

uploaded = files.upload()

for filename in uploaded.keys():

    # predicting images
    path = '/content/' + filename
    image = tf.keras.utils.load_img(path, target_size=(300, 300))
    image = tf.keras.utils.img_to_array(image)
    image = rescale_layer(image)
    image = np.expand_dims(image, axis=0)

    prediction = model.predict(image, verbose=0)[0][0]
    print(f'\nmodel output: {prediction}')

    if prediction > 0.5:
        print(filename + " is a human")
    else:
        print(filename + " is a horse")
```

```
from google.colab import files

uploaded = files.upload()

for filename in uploaded.keys():

    # predicting images
    path = '/content/' + filename
    image = tf.keras.utils.load_img(path, target_size=(300, 300))
    image = tf.keras.utils.img_to_array(image)
    image = rescale_layer(image)
    image = np.expand_dims(image, axis=0)

    prediction = model.predict(image, verbose=0)[0][0]
    print(f'\nmodel output: {prediction}')

    if prediction > 0.5:
        print(filename + " is a human")
    else:
        print(filename + " is a horse")
```

```
from google.colab import files

uploaded = files.upload()

for filename in uploaded.keys():

    # predicting images
    path = '/content/' + filename
    image = tf.keras.utils.load_img(path, target_size=(300, 300))
    image = tf.keras.utils.img_to_array(image)
    image = rescale_layer(image)
    image = np.expand_dims(image, axis=0)

    prediction = model.predict(image, verbose=0)[0][0]
    print(f'\nmodel output: {prediction}')

    if prediction > 0.5:
        print(filename + " is a human")
    else:
        print(filename + " is a horse")
```