```
import pandas as pd
from tqdm import tqdm
label_data = pd.read_csv('/content/drive/MyDrive/archive/HAM10000_metadata.csv')
from sklearn.model_selection import train_test_split
representative_data, _ = train_test_split(label_data, train_size=6000, stratify=label_data['dx'], random_state=0)
import os
import shutil
representative_data['image_path'] = ''
os.makedirs('representative_images', exist_ok=True)
for idx, row in tqdm(representative_data.iterrows(), total=len(representative_data)):
    image_id = row['image_id']
    image_file = f"{image_id}.jpg"  # Or use the correct file extension if different
    if os.path.exists(os.path.join('/content/drive/MyDrive/archive/HAM10000_images_part_1', image_file)):
        source_path = os.path.join('/content/drive/MyDrive/archive/HAM10000_images_part_1', image_file)
    elif os.path.exists(os.path.join('/content/drive/MyDrive/archive/HAM10000_images_part_2', image_file)):
        source_path = os.path.join('/content/drive/MyDrive/archive/HAM10000_images_part_2', image_file)
    else:
        continue
    dest_path = os.path.join('representative_images', image_file)
    shutil.copyfile(source_path, dest_path)
    representative_data.at[idx, 'image_path'] = dest_path
representative_data.to_csv('/content/drive/MyDrive/archive/representative_data.csv', index=False)
```

```
100%|██████████| 6000/6000 [13:17<00:00, 7.52it/s]
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
image_width, image_height = 224, 224  # or the dimensions you want to use
datagen = ImageDataGenerator(rescale=1./255, rotation_range=20,width_shift_range=0.2,
    height_shift_range=0.2,shear_range=0.2,zoom_range=0.2,horizontal_flip=True,fill_mode='nearest',
    validation_split=0.2  # Optional, if you want to split data into train and validation sets)
train_generator = datagen.flow_from_dataframe(
    dataframe=representative_data,x_col='image_path',y_col='dx',
    target_size=(image_width, image_height),class_mode='categorical',
    batch_size=32,shuffle=True,subset='training')
validation_generator = datagen.flow_from_dataframe(dataframe=representative_data,
    x_col='image_path',y_col='dx',target_size=(image_width, image_height),
    class_mode='categorical',batch_size=32,shuffle=True,subset='validation')
```

```
Found 4800 validated image filenames belonging to 7 classes.
Found 1200 validated image filenames belonging to 7 classes.
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization
import tensorflow.keras as keras
num_classes = len(representative_data['dx'].unique())
model = Sequential([
    Conv2D(64, (3, 3), activation='relu', input_shape=(image_width, image_height, 3)),
    BatchNormalization(),
    MaxPooling2D(pool_size=(3)),
    Conv2D(128, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D(pool_size=(3)),
    Conv2D(256, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D(pool_size=(3)),
    Conv2D(512, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D(pool_size=(3)),
    Conv2D(1024, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D(pool_size=(3)),
    Flatten(),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')])
model.compile(optimizer=keras.optimizers.Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(train_generator,steps_per_epoch=len(train_generator),
    epochs=30,validation_data=validation_generator,validation_steps=len(validation_generator))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-40-676fe2e784fc> in <cell line: 7>()
      5 num_classes = len(representative_data['dx'].unique())
      6
----> 7 model = Sequential([
      8     Conv2D(64, (3, 3), activation='relu', input_shape=(image_width, image_height,
3)),
      9     BatchNormalization(),

                                 2 frames
/usr/local/lib/python3.9/dist-packages/keras/layers/convolutional/base_conv.py in
compute_output_shape(self, input_shape)
    352
    353         except ValueError:
--> 354             raise ValueError(
    355                 "One of the dimensions in the output is <= 0 "
    356                 f"due to downsampling in {self.name}. Consider "

ValueError: One of the dimensions in the output is <= 0 due to downsampling in conv2d_20.
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
early_stopping = EarlyStopping(monitor='val_loss', patience=5, verbose=1)
reduce_lr_on_plateau = ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=3, verbose=1)
model.compile(optimizer=keras.optimizers.Adam(learning_rate=0.01), loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(
    train_generator,
    steps_per_epoch=len(train_generator),
    epochs=30,
    validation_data=validation_generator,
    validation_steps=len(validation_generator),
    callbacks=[early_stopping, reduce_lr_on_plateau])
```

```
Epoch 1/30
133/150 [=========================>....] - ETA: 5:58 - loss: 1.1091 - accuracy: 0.6647
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-41-28b3998e1767> in <cell line: 12>()
     10
     11 # Train the model
---> 12 history = model.fit(
     13     train_generator,
     14     steps_per_epoch=len(train_generator),

                                 8 frames
/usr/local/lib/python3.9/dist-packages/tensorflow/python/eager/execute.py in
quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
     50   try:
     51     ctx.ensure_initialized()
---> 52     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name,
     53                                         inputs, attrs, num_outputs)
     54   except core._NotOkStatusException as e:

KeyboardInterrupt:
```

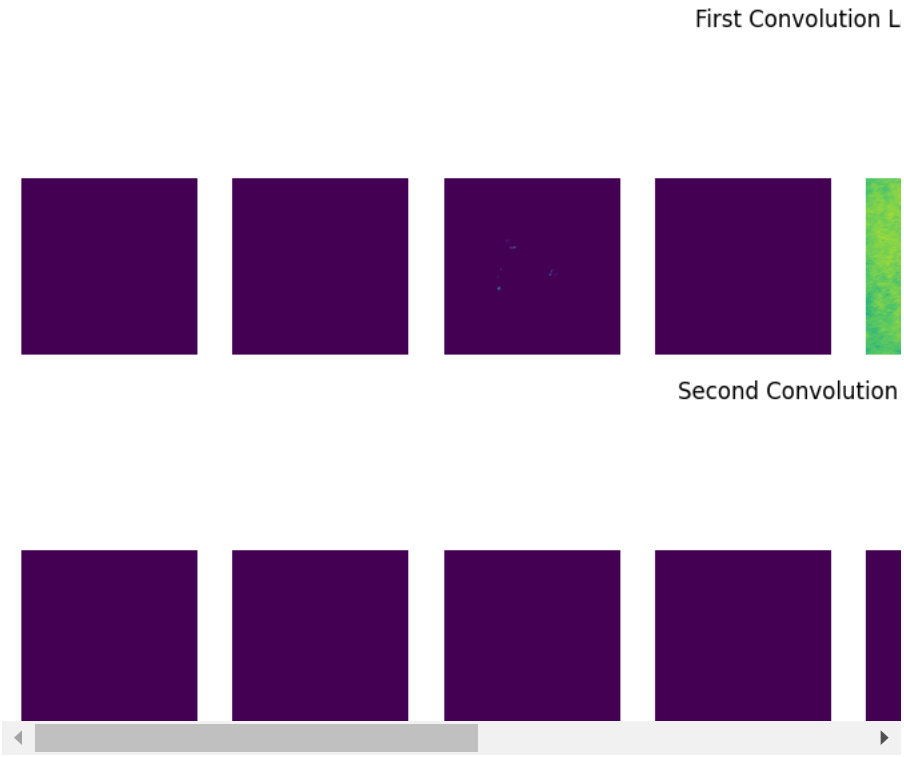
SEARCH STACK OVERFLOW

```
import tensorflow as tf
import numpy as np
import json
from datetime import datetime

timestamp = "22_08-32"

model.save(f'/content/drive/MyDrive/DVD/my_model_{timestamp}.h5')

history_dict = history.history
with open(f'/content/drive/MyDrive/DVD/history_{timestamp}.json', 'w') as file:
    json.dump(history_dict, file)



from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import os
import random
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.keras import Model
from tensorflow.keras.layers import Conv2D
model_path = '/content/drive/MyDrive/Colab Notebooks/my_model.h5'
model = load_model(model_path)
first_conv_layer, second_conv_layer = None, None
for layer in model.layers:
    if isinstance(layer, Conv2D):
        if first_conv_layer is None:
            first_conv_layer = layer
        elif second_conv_layer is None:
            second_conv_layer = layer
            break
folder_path = '/content/drive/MyDrive/archive/HAM10000_images_part_1'
image_name = random.choice(os.listdir(folder_path))
image_path = os.path.join(folder_path, image_name)
input_image = image.load_img(image_path, target_size=(224,224,3))
input_image_array = np.expand_dims(image.img_to_array(input_image), axis=0)
first_conv_layer_output = first_conv_layer.output
second_conv_layer_output = second_conv_layer.output
feature_map_model = Model(inputs=model.inputs, outputs=[first_conv_layer_output, second_conv_layer_output])
feature_maps = feature_map_model.predict(input_image_array)
def plot_feature_maps(feature_maps, layer_name, num_features=8):
    plt.figure(figsize=(16, 4))
    for i in range(num_features):
        plt.subplot(1, num_features, i+1)
        plt.imshow(feature_maps[0, :, :, i], cmap='viridis')
        plt.axis('off')
    plt.suptitle(layer_name)
    plt.show()
plot_feature_maps(feature_maps[0], 'First Convolution Layer')
plot_feature_maps(feature_maps[1], 'Second Convolution Layer')
```

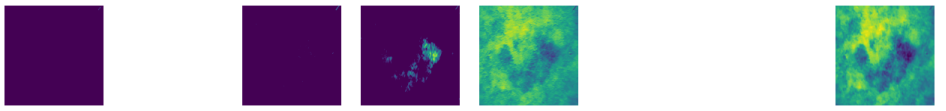
```
1/1 [==============================] - 0s 243ms/step
```



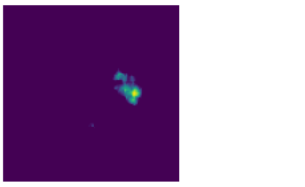

```
import os
import random
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
```

```
from tensorflow.keras import Model
from tensorflow.keras.layers import Conv2D
model_path = '/content/drive/MyDrive/Colab Notebooks/my_model.h5'
model = load_model(model_path)
first_conv_layer, second_conv_layer = None, None
for layer in model.layers:
    if isinstance(layer, Conv2D):
        if first_conv_layer is None:
            first_conv_layer = layer
        elif second_conv_layer is None:
            second_conv_layer = layer
            break
folder_path = '/content/drive/MyDrive/archive/HAM10000_images_part_1'
image_name = random.choice(os.listdir(folder_path))
image_path = os.path.join(folder_path, image_name)
input_image = image.load_img(image_path, target_size=(224,224))
input_image_array = np.expand_dims(image.img_to_array(input_image), axis=0)
first_conv_layer_output = first_conv_layer.output
second_conv_layer_output = second_conv_layer.output
feature_map_model = Model(inputs=model.inputs, outputs=[first_conv_layer_output, second_conv_layer_output])
feature_maps = feature_map_model.predict(input_image_array)
def normalize_feature_map(feature_map):
    return (feature_map - np.min(feature_map)) / (np.max(feature_map) - np.min(feature_map))
def plot_feature_maps(feature_maps, layer_name, num_features=8):
    plt.figure(figsize=(16, 4))
    for i in range(num_features):
        plt.subplot(1, num_features, i+1)
        normalized_feature_map = normalize_feature_map(feature_maps[0, :, :, i])
        plt.imshow(normalized_feature_map, cmap='viridis')
        plt.axis('off')
    plt.suptitle(layer_name)
    plt.show()
plot_feature_maps(feature_maps[0], 'First Convolution Layer')
plot_feature_maps(feature_maps[1], 'Second Convolution Layer')
```

```
1/1 [==============================] - 0s 216ms/step
<ipython-input-10-3ffaefd0f444>:41: RuntimeWarning: invalid value encountered in true_divide
  return (feature_map - np.min(feature_map)) / (np.max(feature_map) - np.min(feature_map))
```




```
<ipython-input-10-3ffaefd0f444>:41: RuntimeWarning: invalid value encountered in true_divide
  return (feature_map - np.min(feature_map)) / (np.max(feature_map) - np.min(feature_map))
```

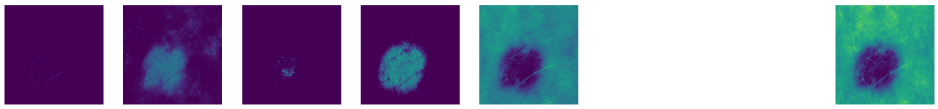



```
import os
import random
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.keras import Model
```

```
from tensorflow.keras import model
from tensorflow.keras.layers import Conv2D
model_path = '/content/drive/MyDrive/Colab Notebooks/my_model.h5'
model = load_model(model_path)
first_conv_layer, second_conv_layer = None, None
for layer in model.layers:
    if isinstance(layer, Conv2D):
        if first_conv_layer is None:
            first_conv_layer = layer
        elif second_conv_layer is None:
            second_conv_layer = layer
            break
folder_path = '/content/drive/MyDrive/archive/HAM10000_images_part_1'
image_name = random.choice(os.listdir(folder_path))
image_path = os.path.join(folder_path, image_name)
input_image = image.load_img(image_path, target_size=(224,224))
input_image_array = np.expand_dims(image.img_to_array(input_image), axis=0)
first_conv_layer_output = first_conv_layer.output
second_conv_layer_output = second_conv_layer.output
feature_map_model = Model(inputs=model.inputs, outputs=[first_conv_layer_output, second_conv_layer_output])
feature_maps = feature_map_model.predict(input_image_array)
def most_activated_filters(feature_map, top_k=8):
    avg_activation_per_filter = np.mean(feature_map, axis=(1, 2))
    sorted_indices = np.argsort(avg_activation_per_filter)[::-1]
    return sorted_indices[:top_k]
top_k = 8
most_activated_filters_first_layer = most_activated_filters(feature_maps[0][0], top_k)
most_activated_filters_second_layer = most_activated_filters(feature_maps[1][0], top_k)
def plot_most_activated_filters(feature_map, layer_name, filter_indices):
    plt.figure(figsize=(16, 4))
    for i, filter_index in enumerate(filter_indices):
        plt.subplot(1, len(filter_indices), i+1)
        normalized_feature_map = normalize_feature_map(feature_map[0, :, :, filter_index])
        plt.imshow(normalized_feature_map, cmap='viridis')
        plt.axis('off')
    plt.suptitle(layer_name)
    plt.show()
plot_most_activated_filters(feature_maps[0], 'Most Activated Filters in First Convolution Layer', most_activated_filters_first_layer)
plot_most_activated_filters(feature_maps[1], 'Most Activated Filters in Second Convolution Layer', most_activated_filters_second_layer)
```

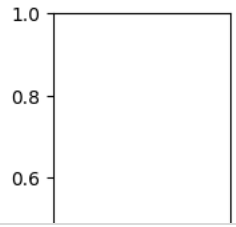
```
WARNING:tensorflow:5 out of the last 5 calls to <function Model.make_predict_function.<local
1/1 [==============================] - 0s 165ms/step
<ipython-input-10-3ffaefd0f444>:41: RuntimeWarning: invalid value encountered in true_divide
  return (feature_map - np.min(feature_map)) / (np.max(feature_map) - np.min(feature_map))
```

Most Activated Filters in First Convolution Layer



```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-11-008615ec5d6a> in <cell line: 62>()
     60
     61 plot_most_activated_filters(feature_maps[0], 'Most Activated Filters in First
Convolution Layer', most_activated_filters_first_layer)
---> 62 plot_most_activated_filters(feature_maps[1], 'Most Activated Filters in Second
Convolution Layer', most_activated_filters_second_layer)

<ipython-input-11-008615ec5d6a> in plot_most_activated_filters(feature_map, layer_name,
filter_indices)
     53     for i, filter_index in enumerate(filter_indices):
     54         plt.subplot(1, len(filter_indices), i+1)
---> 55         normalized_feature_map = normalize_feature_map(feature_map[0, :, :,
filter_index])
     56         plt.imshow(normalized_feature_map, cmap='viridis')
     57         plt.axis('off')

IndexError: index 65 is out of bounds for axis 3 with size 64
```

SEARCH STACK OVERFLOW



4秒 完成时间：12:56