# historical

May 15, 2024

The next two functions (`get_env_type` and `print_versions_and_GPU`) are used because the notebook was developed/run on different environments. They do not contribute to the actual exercise.

```
[ ]: def get_env_type() -> str:
         '''
         Get the environment type where the code is running.

         Returns:
         - 'kaggle' if running on Kaggle
         - 'google.colab' if running on Google Colab
         - 'local' if running on local environment
         '''
         import os, sys
         if 'KAGGLE_KERNEL_RUN_TYPE' in os.environ:
             return 'kaggle'
         elif 'google.colab' in sys.modules:
             if 'COLAB_TPU_ADDR' in os.environ:  # Google Colab w/ TPU
                 # Connect to TPU
                 import tensorflow
                 tpu = tensorflow.distribute.cluster_resolver.TPUClusterResolver()
                 tensorflow.config.experimental_connect_to_cluster(tpu)
                 tensorflow.tpu.experimental.initialize_tpu_system(tpu)
             # Connect to Drive
             from google.colab import drive
             drive.mount('/content/drive')
             return 'google.colab'
         else:   # Running on local environment
             return 'local'

     def print_versions_and_GPU() -> None:
         '''
         Prints version numbers for various modules and GPU information (if␣
      ↪available).
         '''
         import sys, tensorflow, sklearn
```

```
    print(f'Python: {sys.version_info.major}.{sys.version_info.minor}.{sys.
↪version_info.micro}')
    print(f'TensorFlow: {tensorflow.__version__}')
    try:
        print(f'Keras: {tensorflow.keras.version()}')
    except:
        print(f'Keras: Unknown version')
    print(f'Scikit-learn: {sklearn.__version__}')
    gpus = tensorflow.config.list_physical_devices('GPU')
    if gpus is None:
        gpus = tensorflow.test.gpu_device_name()
    print(f'GPUs: {gpus if gpus else "None"}')
```

[ ]:
```
import os
import numpy as np
import pandas as pd
from PIL import Image
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
from matplotlib.figure import Figure
from typing import List, Dict, Tuple
import sklearn
from sklearn.model_selection import train_test_split
import tensorflow
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Flatten, Dense, Dropout, BatchNormalization
from tensorflow.keras.optimizers import Adam
```

[ ]:
```
print_versions_and_GPU()
```

```
Python: 3.11.5
TensorFlow: 2.16.1
Keras: 3.1.1
Scikit-learn: 1.2.2
GPUs: None
```

[ ]:
```
# Determine the environment we're running on.
match get_env_type():
    case 'kaggle':
        raise ValueError('This notebook is not designed to run on Kaggle.')
    case 'google.colab':
        data_path = '/content/drive/MyDrive/data/
↪historical_structures_classification/data'
        models_path = '/content/drive/MyDrive/data/
↪historical_structures_classification/models'
        max_epochs = 200 # Should never reach that high
    case 'local':
        data_path = './data'
        max_epochs = 3
    case _:
        raise ValueError(f'Unknown environment type: {get_env_type()}')

print(f'Running on {get_env_type()}')
```

```
Running on local
```

## 1 Deep Learning

1. Plot the sample images (8–10) from each class or category to gain a better understanding of each class.

   **Hint**: You can use OpenCV open-source library for this task

   Because of familiarity with Pillow, this library will be used instead of OpenCV.

```
[ ]: def load_images(src_folder: str, file_extension: str, delete_if_bad: bool =␣
     ↪True) -> pd.DataFrame:
         images = []
         labels = []
         shapes = []
         channels = []
         for lbl in os.listdir(src_folder):
             lbl_dir = f'{src_folder}/{lbl}'
             for img in [img for img in os.listdir(lbl_dir) if img.
     ↪endswith(file_extension)]:
                 img_path = f'{lbl_dir}/{img}'
                 try:
                     temp = Image.open(img_path)
                     images.append(img_path)
                     labels.append(lbl)
                     shapes.append(temp.size)
                     channels.append(len(temp.getbands()))
                     temp.close()
                 except (OSError, IOError) as exc:
                     print(f'Error loading image: {img_path}: {exc}')
                     if delete_if_bad:
                         os.remove(img_path)
         return pd.DataFrame({'image': images, 'label': labels, 'shape': shapes,␣
     ↪'channels': channels})
```

```
[ ]: df_train = load_images(f'{data_path}/structures/train', '.jpg')
```

```
df_train['label'] = df_train['label'].astype('category')
```

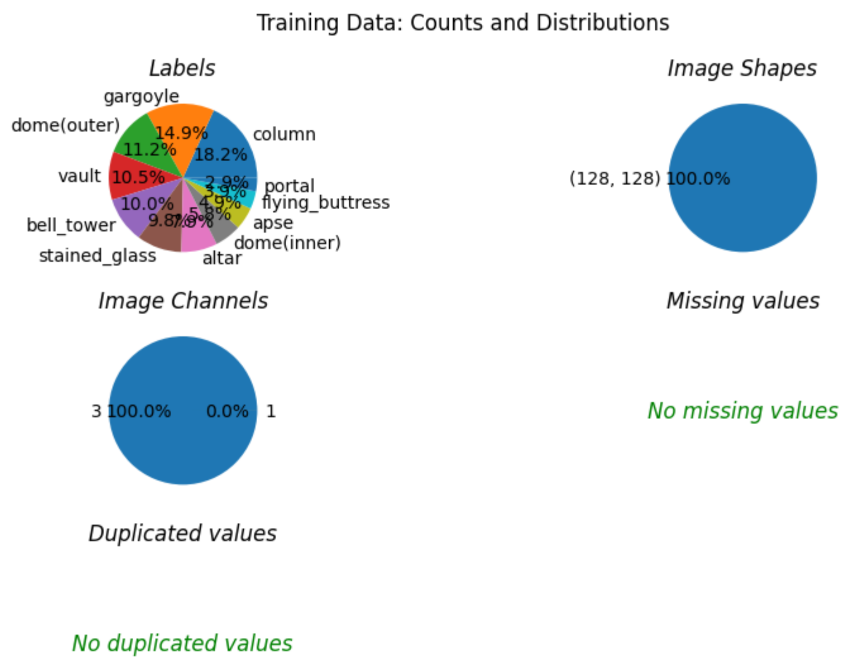
```
[ ]: display(list(dict(df_train['label'].value_counts()).keys()))
```

```
['column',
 'gargoyle',
 'dome(outer)',
 'vault',
 'bell_tower',
 'stained_glass',
 'altar',
 'dome(inner)',
 'apse',
 'flying_buttress',
 'portal']
```

```
[ ]: def counts_and_distributions(df: pd.DataFrame, title: str) -> None:
         plt.figure(figsize=(9, 6))
         plt.suptitle(f'{title}: Counts and Distributions')
         plt.subplot(3, 2, 1)
         plt.pie(df['label'].value_counts(), labels=list(dict(df['label'].
      ↪value_counts()).keys()), autopct='%1.1f%%')
         plt.title('Labels', fontstyle='italic')
         plt.subplot(3, 2, 2)
         plt.pie(df['shape'].value_counts(), labels=list(dict(df['shape'].
      ↪value_counts()).keys()), autopct='%1.1f%%')
         plt.title('Image Shapes', fontstyle='italic')
         plt.subplot(3, 2, 3)
         plt.pie(df['channels'].value_counts(), labels=list(dict(df['channels'].
      ↪value_counts()).keys()), autopct='%1.1f%%')
         plt.title('Image Channels', fontstyle='italic')
         plt.subplot(3, 2, 4)
         if (df.isna().sum().sum() > 0):
             plt.pie(df.isna().sum(), labels=df.isna().sum().index, autopct='%1.
      ↪1f%%')
         else:
             plt.text(0.5, 0.5, 'No missing values', horizontalalignment='center',␣
      ↪verticalalignment='center', fontsize=12, color='green', fontstyle='italic')
             plt.axis('off')
         plt.title('Missing values', fontstyle='italic')
         plt.subplot(3, 2, 5)
         if (df.duplicated().sum() > 0):
             plt.pie(df.duplicated().sum(), labels=['Duplicated'], autopct='%1.1f%%')
         else:
             plt.text(0.5, 0.5, 'No duplicated values',␣
      ↪horizontalalignment='center', verticalalignment='center', fontsize=12,␣
      ↪color='green', fontstyle='italic')
```

```
        plt.axis('off')
    plt.title('Duplicated values', fontstyle='italic')
    plt.tight_layout()
    plt.show()
```

[ ]:
```
counts_and_distributions(df_train, 'Training Data')
```



[ ]:
```
# There are some images with different channels, we will remove them
df_train = df_train[df_train['channels'] == 3]
```

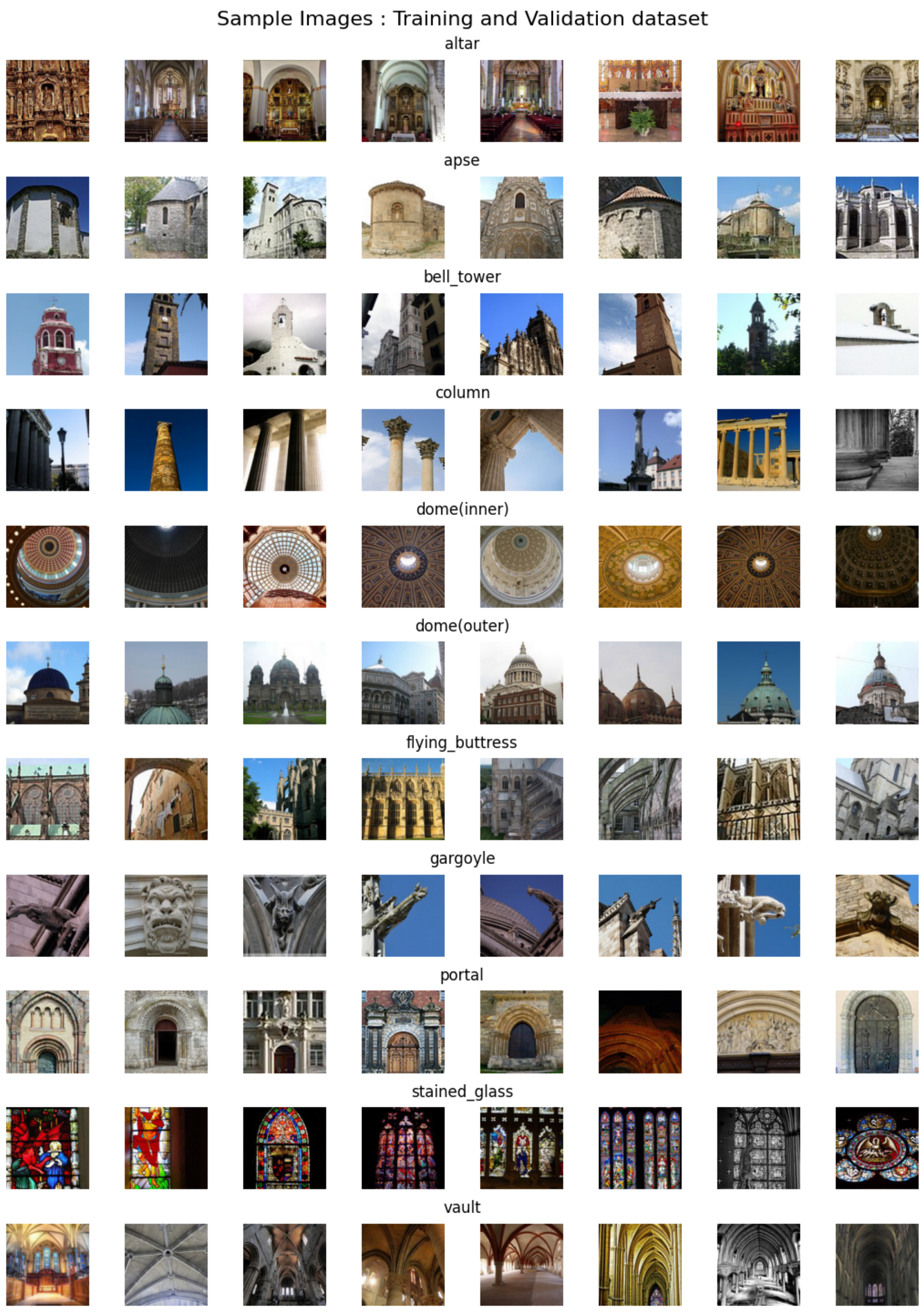
Consistent image size (128 x 128 with 3 channels), except for a very small number that only has 1 channel.

Very imbalanced classes. We should really use some kind of data augmentation to rebalance things.

No Missing or Duplicated data.

[ ]:
```
# Display n random images from each label from the dataframe
def display_images(df: pd.DataFrame, n: int, subtitle: str) -> None:
    labels = df['label'].unique()
    fig = plt.figure(figsize=(1.3*n, 1.3 * len(labels)), layout='constrained')
    fig.suptitle(f'Sample Images : {subtitle}', fontsize=16)
    subfigs = fig.subfigures(nrows=len(labels), ncols=1)
    for l, lbl in enumerate(labels):
        lbl_df = df[df['label'] == lbl]
        subfigs[l].suptitle(lbl, fontsize=12)
        for i, id in enumerate(np.random.choice(len(lbl_df), n, replace=False)):
            sp = subfigs[l].add_subplot(1, n, i+1)
            sp.imshow(Image.open(lbl_df.iloc[id]['image']))
            sp.axis('off')
    plt.show()
```

[ ]:
```
display_images(df_train, 8, 'Training and Validation dataset')
```

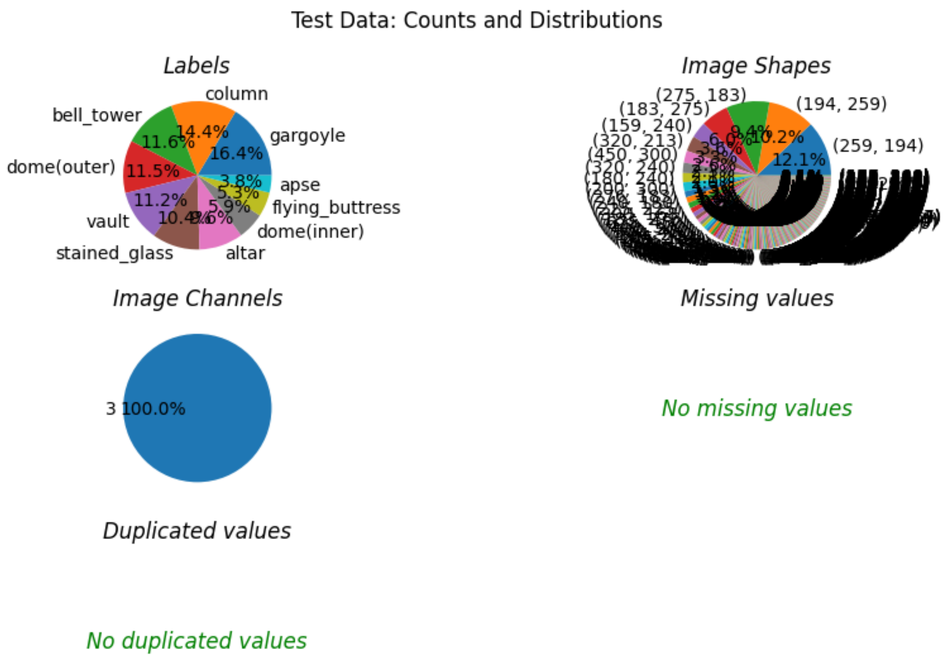

Sample Images : Training and Validation dataset
altar
apse
bell_tower
column
dome(inner)
dome(outer)
flying_buttress
gargoyle
portal
stained_glass
vault

```
[ ]: # Split the train dataset into train and validation datasets
     df2_train, df2_valid = train_test_split(df_train, test_size=0.2,␣
      ↪random_state=42, stratify=df_train['label'])

     # Create the image data generator without augmentation for the train and␣
      ↪validation datasets
     train_datagen = ImageDataGenerator(rescale=1./255)

     train_generator = train_datagen.flow_from_dataframe(
         dataframe=df2_train,
         x_col='image',
         y_col='label',
         target_size=(224, 224),
         class_mode='categorical',
         batch_size=128,
         shuffle=True,
         seed=42
     )

     valid_datagen = ImageDataGenerator(rescale=1./255)

     valid_generator = valid_datagen.flow_from_dataframe(
         dataframe=df2_valid,
         x_col='image',
         y_col='label',
         target_size=(224, 224),
         class_mode='categorical',
         batch_size=128,
         shuffle=False,
         seed=42
     )
```

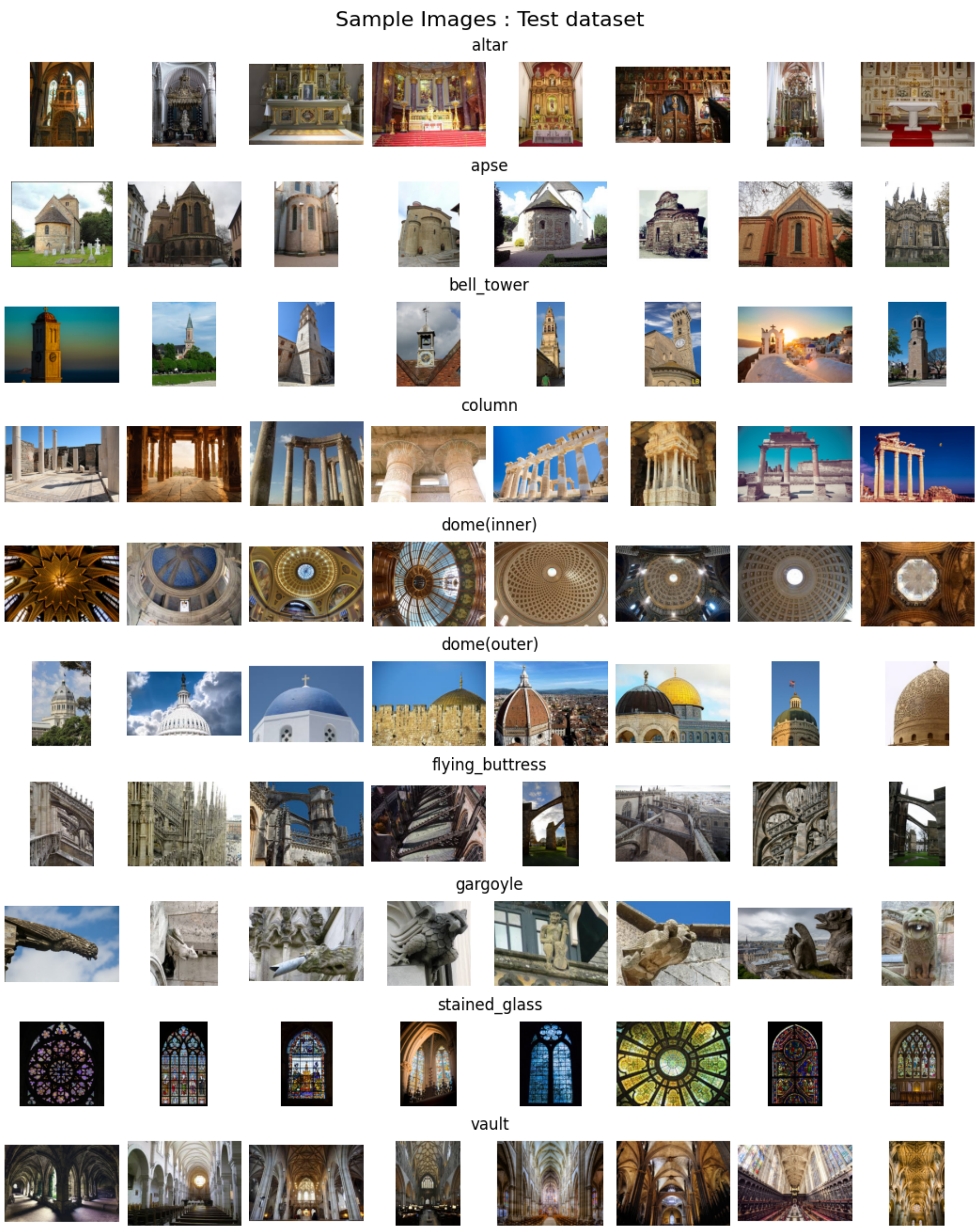
```
Found 8433 validated image filenames belonging to 11 classes.
Found 2109 validated image filenames belonging to 11 classes.
```

```
[ ]: df_test = load_images(f'{data_path}/structures/test', '.jpg')

     df_test['label'] = df_test['label'].astype('category')
```

```
[ ]: counts_and_distributions(df_test, 'Test Data')
```



```
[ ]: display_images(df_test, 8, 'Test dataset')
```



```
[ ]: test_datagen = ImageDataGenerator(rescale=1./255)

     test_generator = test_datagen.flow_from_dataframe(
         dataframe=df2_valid,
         x_col='image',
         y_col='label',
         target_size=(224, 224),
         class_mode='categorical',
         batch_size=128,
         shuffle=False,
         seed=42
     )
```

```
Found 2109 validated image filenames belonging to 11 classes.
```

2. Select an CNN architecture of your choice to train the CV model. Configure the architecture for transfer learning, set up a TensorFlow environment for the selected backbone architecture, and load pre-trained weights
   **Note**: Algorithm or architecture selection is an important step in the training of ML models, so select the one that performs the best on your dataset.
   >
   > To select the specific architecture, we'll run the training on several architecture: MobileNet V3 (Large and Small) as well as VGG16. Time and computing resources prevented further investigation for this exercise.
   >

3. Deep learning models tend to work well with large amounts (millions of images) of data, but we may not always have enough data to train a deep learning model from scratch. Transfer learning techniques allow us to train and fine-tune large deep learning architectures using limited data.
   **Hint**: For transfer learning, use pre-trained CNN weights and freeze all convolutional layers' weights.

4. As of now, CNN architecture has been configured for our model. Modify the top of this architecture to work with our dataset by:

   - Adding an appropriate number of dense layers with an activation function.

   - Using dropout for regularization.

   **Note**: It is important to understand that these parameters are hyperparameters that must be tuned.
   >
   > We'll use the `mish` activation function on trainable hidden layers, as it has been shown to perform significantly better during training than ReLU, leaky-ReLU, tanh, or signmoid functions.
   >

5. Compile the model with the right set of parameters like optimizer, loss function, and metric
   >
   > We'll use 3 metrics to track progress: "Top-1 accuracy", precision, and recall.
   >

[ ]:
```
import tensorflow
```

```
# Build or load a model if it was previously saved in {data_path}/models
def build_or_load_model(pre_trained_model: tensorflow.keras.Model,
                        num_classes: int,
                        name: str = None) -> tensorflow.keras.Model:
    try:
      model = tensorflow.keras.models.load_model(f"{models_path}/{name}.keras")
    except OSError:
      pre_trained_model.trainable = False
      model = Sequential([
        pre_trained_model,
        Flatten(),
        Dense(256, activation='mish'),
        Dropout(0.3),
        Dense(128, activation='mish'),
        Dropout(0.3),
        Dense(num_classes, activation='softmax')
      ], name=name)
      model.compile(
        optimizer=Adam(),
        loss='categorical_crossentropy',
        metrics=[     # Older versions of Keras don't define some metrics as␣
 ↪strings
          tensorflow.keras.metrics.TopKCategoricalAccuracy(k=1,␣
 ↪name='accuracy'),
          tensorflow.keras.metrics.Precision(name='precision'),
          tensorflow.keras.metrics.Recall(name='recall')]
      )
    model.summary()
    return model
```

6. Define your callback class to stop the training once validation accuracy reaches a certain number of your choice
   >
   > We use both `EarlyStopping` and `ReduceLROnPlateau` to control learning rate and training termination.
   >

[ ]:
```
# Define the callbacks for the model
import tensorflow
def callbacks(monitor: str = 'val_accuracy', es_patience=20, lr_patience=7) ->␣
 ↪List[tensorflow.keras.callbacks.Callback]:
    # Early stopping and learning rate reduction
    # if the monitor string is anything but 'loss' or 'val_loss', then the mode
    # should be 'max', otherwise 'min'
    if (monitor != 'loss' and monitor != 'val_loss'):
        mode = 'max'
    else:
        mode = 'min'
    return [
        tensorflow.keras.callbacks.EarlyStopping(
            monitor=monitor,
            patience=es_patience,
            restore_best_weights=True,
            verbose=2,
            mode=mode),
        tensorflow.keras.callbacks.ReduceLROnPlateau(
            monitor=monitor,
            factor=0.3,
            patience=lr_patience,
            verbose=2,
            min_lr=1e-6,
            mode=mode)]
```

[ ]:
```
# Build the model
model_mnv3l = build_or_load_model(
    tensorflow.keras.applications.MobileNetV3Large(include_top=False,␣
 ↪weights='imagenet', input_shape=(224, 224, 3)),
    len(df_train['label'].unique()),
    name='MobileNetV3Large')
```

```
Model: "MobileNetV3Large"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 MobilenetV3large (Function  (None, 7, 7, 960)         2996352
 al)

 flatten_4 (Flatten)         (None, 47040)             0

 dense_12 (Dense)            (None, 256)               12042496

 dropout_8 (Dropout)         (None, 256)               0

 dense_13 (Dense)            (None, 128)               32896

 dropout_9 (Dropout)         (None, 128)               0

 dense_14 (Dense)            (None, 11)                1419

=================================================================
Total params: 15073163 (57.50 MB)
Trainable params: 12076811 (46.07 MB)
Non-trainable params: 2996352 (11.43 MB)
_________________________________________________________________
```

7. Setup the train or test dataset directories and review the number of image samples for the train and test datasets for each class
   >
   > This was actually done earlier. It's interesting to note that the `portal` class is "missing" from the test dataset (but present in training/validation).
   >

8. Train the model without augmentation while continuously monitoring the validation accuracy

[ ]:
```
# Train the model and keep track and report compute time, and save the resulting
# model
def model_fit(model: tensorflow.keras.models.Model,
              train_generator: tensorflow.data.Dataset,
              valid_generator: tensorflow.data.Dataset,
              max_epochs: int,
              callbacks: List[tensorflow.keras.callbacks.Callback],
              **kwargs):
  import time
  start = time.time()
  history = model.fit(train_generator,
                      validation_data=valid_generator,
                      epochs=max_epochs,
                      callbacks=callbacks,
                      **kwargs)
  end = time.time()
  training_time = end - start
  print(f'{model.name} Training time: {training_time // 60} min {training_time␣
 ↪% 60:.2f} sec')
  model.save(f'{models_path}/{model.name}.keras')
  return history
```

[ ]:
```
history = model_fit(model_mnv3l, train_generator, valid_generator, max_epochs,␣
 ↪callbacks())
```

```
Epoch 1/200
66/66 [==============================] - 83s 1s/step - loss: 2.4973 - accuracy:
0.2356 - precision: 0.3273 - recall: 0.0427 - val_loss: 1.8431 - val_accuracy:
0.3812 - val_precision: 0.9189 - val_recall: 0.0806 - lr: 0.0010
Epoch 2/200
66/66 [==============================] - 77s 1s/step - loss: 1.8751 - accuracy:
0.3633 - precision: 0.7364 - recall: 0.1156 - val_loss: 1.6189 - val_accuracy:
0.4685 - val_precision: 0.8737 - val_recall: 0.1607 - lr: 0.0010
Epoch 3/200
66/66 [==============================] - 78s 1s/step - loss: 1.7811 - accuracy:
0.3893 - precision: 0.7153 - recall: 0.1546 - val_loss: 1.6087 - val_accuracy:
0.4528 - val_precision: 0.8481 - val_recall: 0.2172 - lr: 0.0010
Epoch 4/200
66/66 [==============================] - 78s 1s/step - loss: 1.7168 - accuracy:
0.4110 - precision: 0.7145 - recall: 0.1843 - val_loss: 1.5228 - val_accuracy:
```

```
0.4860 - val_precision: 0.8166 - val_recall: 0.2195 - lr: 0.0010
Epoch 5/200
66/66 [==============================] - 77s 1s/step - loss: 1.6873 - accuracy:
0.4211 - precision: 0.7255 - recall: 0.1946 - val_loss: 1.5181 - val_accuracy:
0.4751 - val_precision: 0.7781 - val_recall: 0.2461 - lr: 0.0010
Epoch 6/200
66/66 [==============================] - 77s 1s/step - loss: 1.6643 - accuracy:
0.4222 - precision: 0.7338 - recall: 0.1974 - val_loss: 1.4795 - val_accuracy:
0.5287 - val_precision: 0.8951 - val_recall: 0.1982 - lr: 0.0010
Epoch 7/200
66/66 [==============================] - 77s 1s/step - loss: 1.6301 - accuracy:
0.4331 - precision: 0.7314 - recall: 0.2140 - val_loss: 1.4485 - val_accuracy:
0.5301 - val_precision: 0.8456 - val_recall: 0.2442 - lr: 0.0010
Epoch 8/200
66/66 [==============================] - 76s 1s/step - loss: 1.5613 - accuracy:
0.4614 - precision: 0.7359 - recall: 0.2484 - val_loss: 1.5638 - val_accuracy:
0.5017 - val_precision: 0.8141 - val_recall: 0.2077 - lr: 0.0010
Epoch 9/200
66/66 [==============================] - 77s 1s/step - loss: 1.5728 - accuracy:
0.4584 - precision: 0.7498 - recall: 0.2363 - val_loss: 1.3722 - val_accuracy:
0.5481 - val_precision: 0.8292 - val_recall: 0.2831 - lr: 0.0010
Epoch 10/200
66/66 [==============================] - 77s 1s/step - loss: 1.5697 - accuracy:
0.4561 - precision: 0.7458 - recall: 0.2449 - val_loss: 1.3864 - val_accuracy:
0.5567 - val_precision: 0.8341 - val_recall: 0.2504 - lr: 0.0010
Epoch 11/200
66/66 [==============================] - 77s 1s/step - loss: 1.5682 - accuracy:
0.4552 - precision: 0.7476 - recall: 0.2410 - val_loss: 1.5309 - val_accuracy:
0.5031 - val_precision: 0.8240 - val_recall: 0.2707 - lr: 0.0010
Epoch 12/200
66/66 [==============================] - 76s 1s/step - loss: 1.5791 - accuracy:
0.4620 - precision: 0.7541 - recall: 0.2429 - val_loss: 1.3865 - val_accuracy:
0.5510 - val_precision: 0.8033 - val_recall: 0.2963 - lr: 0.0010
Epoch 13/200
66/66 [==============================] - 77s 1s/step - loss: 1.6298 - accuracy:
0.4307 - precision: 0.7281 - recall: 0.2239 - val_loss: 1.4199 - val_accuracy:
0.5287 - val_precision: 0.8723 - val_recall: 0.2266 - lr: 0.0010
Epoch 14/200
66/66 [==============================] - 76s 1s/step - loss: 1.6176 - accuracy:
0.4189 - precision: 0.7621 - recall: 0.2086 - val_loss: 1.4546 - val_accuracy:
0.5391 - val_precision: 0.8953 - val_recall: 0.1826 - lr: 0.0010
Epoch 15/200
66/66 [==============================] - 77s 1s/step - loss: 1.6549 - accuracy:
0.4054 - precision: 0.7383 - recall: 0.1991 - val_loss: 1.4957 - val_accuracy:
0.5225 - val_precision: 0.8783 - val_recall: 0.1745 - lr: 0.0010
Epoch 16/200
66/66 [==============================] - 77s 1s/step - loss: 1.5742 - accuracy:
0.4339 - precision: 0.7715 - recall: 0.2263 - val_loss: 1.4210 - val_accuracy:
```

```
0.5353 - val_precision: 0.7947 - val_recall: 0.2570 - lr: 0.0010
Epoch 17/200
66/66 [==============================] - 77s 1s/step - loss: 1.5238 - accuracy:
0.4565 - precision: 0.7491 - recall: 0.2507 - val_loss: 1.3656 - val_accuracy:
0.5633 - val_precision: 0.8558 - val_recall: 0.2560 - lr: 0.0010
Epoch 18/200
66/66 [==============================] - 77s 1s/step - loss: 1.4833 - accuracy:
0.4784 - precision: 0.7434 - recall: 0.2622 - val_loss: 1.3679 - val_accuracy:
0.5514 - val_precision: 0.7624 - val_recall: 0.3423 - lr: 0.0010
Epoch 19/200
66/66 [==============================] - 77s 1s/step - loss: 1.4701 - accuracy:
0.4859 - precision: 0.7613 - recall: 0.2738 - val_loss: 1.3381 - val_accuracy:
0.5747 - val_precision: 0.8386 - val_recall: 0.2907 - lr: 0.0010
Epoch 20/200
66/66 [==============================] - 77s 1s/step - loss: 1.5044 - accuracy:
0.4734 - precision: 0.7413 - recall: 0.2752 - val_loss: 1.3715 - val_accuracy:
0.5624 - val_precision: 0.8087 - val_recall: 0.3087 - lr: 0.0010
Epoch 21/200
66/66 [==============================] - 77s 1s/step - loss: 1.4717 - accuracy:
0.4844 - precision: 0.7526 - recall: 0.2857 - val_loss: 1.4280 - val_accuracy:
0.5277 - val_precision: 0.8014 - val_recall: 0.2774 - lr: 0.0010
Epoch 22/200
66/66 [==============================] - 78s 1s/step - loss: 1.4472 - accuracy:
0.4813 - precision: 0.7540 - recall: 0.2763 - val_loss: 1.3784 - val_accuracy:
0.5495 - val_precision: 0.8062 - val_recall: 0.2840 - lr: 0.0010
Epoch 23/200
66/66 [==============================] - 76s 1s/step - loss: 1.4342 - accuracy:
0.4920 - precision: 0.7600 - recall: 0.2906 - val_loss: 1.3179 - val_accuracy:
0.5638 - val_precision: 0.8147 - val_recall: 0.3210 - lr: 0.0010
Epoch 24/200
66/66 [==============================] - 77s 1s/step - loss: 1.4067 - accuracy:
0.5053 - precision: 0.7659 - recall: 0.3061 - val_loss: 1.4132 - val_accuracy:
0.5334 - val_precision: 0.8069 - val_recall: 0.2537 - lr: 0.0010
Epoch 25/200
66/66 [==============================] - 77s 1s/step - loss: 1.4871 - accuracy:
0.4822 - precision: 0.7479 - recall: 0.2695 - val_loss: 1.3417 - val_accuracy:
0.5671 - val_precision: 0.8019 - val_recall: 0.3148 - lr: 0.0010
Epoch 26/200
66/66 [==============================] - 77s 1s/step - loss: 1.4146 - accuracy:
0.5007 - precision: 0.7499 - recall: 0.2941 - val_loss: 1.3078 - val_accuracy:
0.5756 - val_precision: 0.8319 - val_recall: 0.3167 - lr: 0.0010
Epoch 27/200
66/66 [==============================] - 76s 1s/step - loss: 1.3962 - accuracy:
0.5011 - precision: 0.7654 - recall: 0.3076 - val_loss: 1.4257 - val_accuracy:
0.5287 - val_precision: 0.8704 - val_recall: 0.2262 - lr: 0.0010
Epoch 28/200
66/66 [==============================] - 77s 1s/step - loss: 1.3884 - accuracy:
0.5081 - precision: 0.7712 - recall: 0.3070 - val_loss: 1.3160 - val_accuracy:
```

```
0.5709 - val_precision: 0.7627 - val_recall: 0.3917 - lr: 0.0010
Epoch 29/200
66/66 [==============================] - 78s 1s/step - loss: 1.3333 - accuracy:
0.5272 - precision: 0.7820 - recall: 0.3323 - val_loss: 1.2652 - val_accuracy:
0.5903 - val_precision: 0.8107 - val_recall: 0.3532 - lr: 0.0010
Epoch 30/200
66/66 [==============================] - 77s 1s/step - loss: 1.3507 - accuracy:
0.5177 - precision: 0.7621 - recall: 0.3275 - val_loss: 1.2580 - val_accuracy:
0.6074 - val_precision: 0.7822 - val_recall: 0.3917 - lr: 0.0010
Epoch 31/200
66/66 [==============================] - 77s 1s/step - loss: 1.3556 - accuracy:
0.5192 - precision: 0.7590 - recall: 0.3293 - val_loss: 1.2923 - val_accuracy:
0.5799 - val_precision: 0.7493 - val_recall: 0.4068 - lr: 0.0010
Epoch 32/200
66/66 [==============================] - 77s 1s/step - loss: 1.2940 - accuracy:
0.5464 - precision: 0.7852 - recall: 0.3485 - val_loss: 1.2821 - val_accuracy:
0.5823 - val_precision: 0.7922 - val_recall: 0.3850 - lr: 0.0010
Epoch 33/200
66/66 [==============================] - 76s 1s/step - loss: 1.3067 - accuracy:
0.5366 - precision: 0.7686 - recall: 0.3546 - val_loss: 1.2913 - val_accuracy:
0.5903 - val_precision: 0.7899 - val_recall: 0.3637 - lr: 0.0010
Epoch 34/200
66/66 [==============================] - 76s 1s/step - loss: 1.3088 - accuracy:
0.5282 - precision: 0.7705 - recall: 0.3520 - val_loss: 1.2848 - val_accuracy:
0.5842 - val_precision: 0.7898 - val_recall: 0.3741 - lr: 0.0010
Epoch 35/200
66/66 [==============================] - 77s 1s/step - loss: 1.2528 - accuracy:
0.5545 - precision: 0.7762 - recall: 0.3644 - val_loss: 1.2630 - val_accuracy:
0.5908 - val_precision: 0.7618 - val_recall: 0.3898 - lr: 0.0010
Epoch 36/200
66/66 [==============================] - 77s 1s/step - loss: 1.2637 - accuracy:
0.5461 - precision: 0.7769 - recall: 0.3750 - val_loss: 1.2741 - val_accuracy:
0.5870 - val_precision: 0.7533 - val_recall: 0.4272 - lr: 0.0010
Epoch 37/200
66/66 [==============================] - ETA: 0s - loss: 1.2370 - accuracy:
0.5642 - precision: 0.7798 - recall: 0.3847
Epoch 37: ReduceLROnPlateau reducing learning rate to 0.0003000000142492354.
66/66 [==============================] - 77s 1s/step - loss: 1.2370 - accuracy:
0.5642 - precision: 0.7798 - recall: 0.3847 - val_loss: 1.2628 - val_accuracy:
0.5903 - val_precision: 0.7581 - val_recall: 0.4220 - lr: 0.0010
Epoch 38/200
66/66 [==============================] - 77s 1s/step - loss: 1.1550 - accuracy:
0.5843 - precision: 0.7987 - recall: 0.4136 - val_loss: 1.2020 - val_accuracy:
0.6169 - val_precision: 0.7932 - val_recall: 0.4201 - lr: 3.0000e-04
Epoch 39/200
66/66 [==============================] - 77s 1s/step - loss: 1.1365 - accuracy:
0.5953 - precision: 0.8121 - recall: 0.4227 - val_loss: 1.1922 - val_accuracy:
0.6193 - val_precision: 0.7915 - val_recall: 0.4139 - lr: 3.0000e-04
```

```
Epoch 40/200
66/66 [==============================] - 77s 1s/step - loss: 1.1359 - accuracy:
0.5987 - precision: 0.8079 - recall: 0.4255 - val_loss: 1.2141 - val_accuracy:
0.6226 - val_precision: 0.7767 - val_recall: 0.4272 - lr: 3.0000e-04
Epoch 41/200
66/66 [==============================] - 77s 1s/step - loss: 1.1185 - accuracy:
0.6046 - precision: 0.8006 - recall: 0.4323 - val_loss: 1.2097 - val_accuracy:
0.6093 - val_precision: 0.7746 - val_recall: 0.4367 - lr: 3.0000e-04
Epoch 42/200
66/66 [==============================] - 77s 1s/step - loss: 1.1488 - accuracy:
0.5950 - precision: 0.7964 - recall: 0.4226 - val_loss: 1.1996 - val_accuracy:
0.6240 - val_precision: 0.7959 - val_recall: 0.4234 - lr: 3.0000e-04
Epoch 43/200
66/66 [==============================] - 77s 1s/step - loss: 1.1173 - accuracy:
0.6022 - precision: 0.8152 - recall: 0.4332 - val_loss: 1.1896 - val_accuracy:
0.6249 - val_precision: 0.7782 - val_recall: 0.4443 - lr: 3.0000e-04
Epoch 44/200
66/66 [==============================] - 77s 1s/step - loss: 1.1060 - accuracy:
0.6124 - precision: 0.8134 - recall: 0.4450 - val_loss: 1.1967 - val_accuracy:
0.6221 - val_precision: 0.7913 - val_recall: 0.4154 - lr: 3.0000e-04
Epoch 45/200
66/66 [==============================] - 76s 1s/step - loss: 1.0991 - accuracy:
0.6089 - precision: 0.8082 - recall: 0.4473 - val_loss: 1.1938 - val_accuracy:
0.6249 - val_precision: 0.7740 - val_recall: 0.4580 - lr: 3.0000e-04
Epoch 46/200
66/66 [==============================] - 77s 1s/step - loss: 1.0872 - accuracy:
0.6095 - precision: 0.8071 - recall: 0.4486 - val_loss: 1.1817 - val_accuracy:
0.6283 - val_precision: 0.7693 - val_recall: 0.4822 - lr: 3.0000e-04
Epoch 47/200
66/66 [==============================] - 76s 1s/step - loss: 1.1062 - accuracy:
0.6043 - precision: 0.7997 - recall: 0.4473 - val_loss: 1.2031 - val_accuracy:
0.6164 - val_precision: 0.7646 - val_recall: 0.4775 - lr: 3.0000e-04
Epoch 48/200
66/66 [==============================] - 77s 1s/step - loss: 1.1022 - accuracy:
0.6095 - precision: 0.8077 - recall: 0.4497 - val_loss: 1.1702 - val_accuracy:
0.6339 - val_precision: 0.7950 - val_recall: 0.4523 - lr: 3.0000e-04
Epoch 49/200
66/66 [==============================] - 76s 1s/step - loss: 1.0881 - accuracy:
0.6182 - precision: 0.8104 - recall: 0.4612 - val_loss: 1.1786 - val_accuracy:
0.6330 - val_precision: 0.7855 - val_recall: 0.4481 - lr: 3.0000e-04
Epoch 50/200
66/66 [==============================] - 76s 1s/step - loss: 1.0847 - accuracy:
0.6202 - precision: 0.8077 - recall: 0.4602 - val_loss: 1.1850 - val_accuracy:
0.6249 - val_precision: 0.7582 - val_recall: 0.4950 - lr: 3.0000e-04
Epoch 51/200
66/66 [==============================] - 76s 1s/step - loss: 1.0794 - accuracy:
0.6153 - precision: 0.7974 - recall: 0.4577 - val_loss: 1.2100 - val_accuracy:
0.6178 - val_precision: 0.7789 - val_recall: 0.4694 - lr: 3.0000e-04
```

```
Epoch 52/200
66/66 [==============================] - 77s 1s/step - loss: 1.0511 - accuracy:
0.6252 - precision: 0.8069 - recall: 0.4711 - val_loss: 1.1855 - val_accuracy:
0.6321 - val_precision: 0.7677 - val_recall: 0.4936 - lr: 3.0000e-04
Epoch 53/200
66/66 [==============================] - 76s 1s/step - loss: 1.0415 - accuracy:
0.6303 - precision: 0.8146 - recall: 0.4731 - val_loss: 1.1749 - val_accuracy:
0.6358 - val_precision: 0.7662 - val_recall: 0.4988 - lr: 3.0000e-04
Epoch 54/200
66/66 [==============================] - 77s 1s/step - loss: 1.0361 - accuracy:
0.6346 - precision: 0.8152 - recall: 0.4813 - val_loss: 1.1772 - val_accuracy:
0.6358 - val_precision: 0.7674 - val_recall: 0.4974 - lr: 3.0000e-04
Epoch 55/200
66/66 [==============================] - 77s 1s/step - loss: 1.0503 - accuracy:
0.6242 - precision: 0.8100 - recall: 0.4767 - val_loss: 1.1599 - val_accuracy:
0.6401 - val_precision: 0.7679 - val_recall: 0.4974 - lr: 3.0000e-04
Epoch 56/200
66/66 [==============================] - 77s 1s/step - loss: 1.0302 - accuracy:
0.6373 - precision: 0.8050 - recall: 0.4804 - val_loss: 1.1615 - val_accuracy:
0.6278 - val_precision: 0.7884 - val_recall: 0.4841 - lr: 3.0000e-04
Epoch 57/200
66/66 [==============================] - 77s 1s/step - loss: 1.0252 - accuracy:
0.6358 - precision: 0.8145 - recall: 0.4890 - val_loss: 1.2270 - val_accuracy:
0.6140 - val_precision: 0.7599 - val_recall: 0.4742 - lr: 3.0000e-04
Epoch 58/200
66/66 [==============================] - 77s 1s/step - loss: 1.0509 - accuracy:
0.6284 - precision: 0.7991 - recall: 0.4749 - val_loss: 1.1761 - val_accuracy:
0.6302 - val_precision: 0.7653 - val_recall: 0.4931 - lr: 3.0000e-04
Epoch 59/200
66/66 [==============================] - 77s 1s/step - loss: 1.0355 - accuracy:
0.6305 - precision: 0.8083 - recall: 0.4786 - val_loss: 1.1478 - val_accuracy:
0.6354 - val_precision: 0.7869 - val_recall: 0.4798 - lr: 3.0000e-04
Epoch 60/200
66/66 [==============================] - 77s 1s/step - loss: 1.0012 - accuracy:
0.6488 - precision: 0.8195 - recall: 0.4985 - val_loss: 1.1387 - val_accuracy:
0.6453 - val_precision: 0.7888 - val_recall: 0.4922 - lr: 3.0000e-04
Epoch 61/200
66/66 [==============================] - 78s 1s/step - loss: 1.0051 - accuracy:
0.6502 - precision: 0.8097 - recall: 0.4884 - val_loss: 1.1522 - val_accuracy:
0.6392 - val_precision: 0.7903 - val_recall: 0.4860 - lr: 3.0000e-04
Epoch 62/200
66/66 [==============================] - 77s 1s/step - loss: 0.9892 - accuracy:
0.6541 - precision: 0.8180 - recall: 0.5042 - val_loss: 1.1964 - val_accuracy:
0.6354 - val_precision: 0.7569 - val_recall: 0.5107 - lr: 3.0000e-04
Epoch 63/200
66/66 [==============================] - 77s 1s/step - loss: 0.9865 - accuracy:
0.6478 - precision: 0.8147 - recall: 0.5074 - val_loss: 1.1589 - val_accuracy:
0.6406 - val_precision: 0.7659 - val_recall: 0.5211 - lr: 3.0000e-04
```

```
Epoch 64/200
66/66 [==============================] - 77s 1s/step - loss: 0.9800 - accuracy:
0.6554 - precision: 0.8168 - recall: 0.5112 - val_loss: 1.1849 - val_accuracy:
0.6249 - val_precision: 0.7697 - val_recall: 0.4770 - lr: 3.0000e-04
Epoch 65/200
66/66 [==============================] - 77s 1s/step - loss: 0.9804 - accuracy:
0.6507 - precision: 0.8096 - recall: 0.5099 - val_loss: 1.1442 - val_accuracy:
0.6420 - val_precision: 0.7700 - val_recall: 0.5192 - lr: 3.0000e-04
Epoch 66/200
66/66 [==============================] - 77s 1s/step - loss: 0.9576 - accuracy:
0.6635 - precision: 0.8212 - recall: 0.5157 - val_loss: 1.1833 - val_accuracy:
0.6297 - val_precision: 0.7568 - val_recall: 0.5149 - lr: 3.0000e-04
Epoch 67/200
66/66 [==============================] - 77s 1s/step - loss: 0.9644 - accuracy:
0.6580 - precision: 0.8161 - recall: 0.5143 - val_loss: 1.1492 - val_accuracy:
0.6496 - val_precision: 0.7813 - val_recall: 0.4964 - lr: 3.0000e-04
Epoch 68/200
66/66 [==============================] - 77s 1s/step - loss: 0.9688 - accuracy:
0.6574 - precision: 0.8246 - recall: 0.5157 - val_loss: 1.2543 - val_accuracy:
0.6254 - val_precision: 0.7240 - val_recall: 0.5386 - lr: 3.0000e-04
Epoch 69/200
66/66 [==============================] - 77s 1s/step - loss: 0.9714 - accuracy:
0.6503 - precision: 0.7994 - recall: 0.5112 - val_loss: 1.1651 - val_accuracy:
0.6458 - val_precision: 0.7963 - val_recall: 0.4931 - lr: 3.0000e-04
Epoch 70/200
66/66 [==============================] - 77s 1s/step - loss: 0.9542 - accuracy:
0.6637 - precision: 0.8132 - recall: 0.5200 - val_loss: 1.1491 - val_accuracy:
0.6463 - val_precision: 0.7765 - val_recall: 0.4974 - lr: 3.0000e-04
Epoch 71/200
66/66 [==============================] - 77s 1s/step - loss: 0.9474 - accuracy:
0.6639 - precision: 0.8145 - recall: 0.5244 - val_loss: 1.1751 - val_accuracy:
0.6468 - val_precision: 0.7741 - val_recall: 0.5102 - lr: 3.0000e-04
Epoch 72/200
66/66 [==============================] - 77s 1s/step - loss: 0.9330 - accuracy:
0.6695 - precision: 0.8194 - recall: 0.5391 - val_loss: 1.2123 - val_accuracy:
0.6387 - val_precision: 0.7486 - val_recall: 0.5239 - lr: 3.0000e-04
Epoch 73/200
66/66 [==============================] - 77s 1s/step - loss: 0.9337 - accuracy:
0.6711 - precision: 0.8200 - recall: 0.5330 - val_loss: 1.1474 - val_accuracy:
0.6515 - val_precision: 0.7745 - val_recall: 0.5211 - lr: 3.0000e-04
Epoch 74/200
66/66 [==============================] - 77s 1s/step - loss: 0.9151 - accuracy:
0.6783 - precision: 0.8276 - recall: 0.5403 - val_loss: 1.2079 - val_accuracy:
0.6382 - val_precision: 0.7509 - val_recall: 0.5088 - lr: 3.0000e-04
Epoch 75/200
66/66 [==============================] - 77s 1s/step - loss: 0.9504 - accuracy:
0.6642 - precision: 0.8164 - recall: 0.5285 - val_loss: 1.3126 - val_accuracy:
0.6046 - val_precision: 0.7376 - val_recall: 0.4784 - lr: 3.0000e-04
```

```
Epoch 76/200
66/66 [==============================] - 77s 1s/step - loss: 0.9390 - accuracy:
0.6695 - precision: 0.8145 - recall: 0.5382 - val_loss: 1.1451 - val_accuracy:
0.6543 - val_precision: 0.7736 - val_recall: 0.5410 - lr: 3.0000e-04
Epoch 77/200
66/66 [==============================] - 78s 1s/step - loss: 0.9183 - accuracy:
0.6731 - precision: 0.8211 - recall: 0.5410 - val_loss: 1.1544 - val_accuracy:
0.6562 - val_precision: 0.7685 - val_recall: 0.5510 - lr: 3.0000e-04
Epoch 78/200
66/66 [==============================] - 77s 1s/step - loss: 0.9077 - accuracy:
0.6764 - precision: 0.8255 - recall: 0.5463 - val_loss: 1.1537 - val_accuracy:
0.6449 - val_precision: 0.7878 - val_recall: 0.5159 - lr: 3.0000e-04
Epoch 79/200
66/66 [==============================] - 76s 1s/step - loss: 0.8951 - accuracy:
0.6805 - precision: 0.8255 - recall: 0.5482 - val_loss: 1.1387 - val_accuracy:
0.6548 - val_precision: 0.7899 - val_recall: 0.5135 - lr: 3.0000e-04
Epoch 80/200
66/66 [==============================] - 77s 1s/step - loss: 0.9005 - accuracy:
0.6854 - precision: 0.8220 - recall: 0.5486 - val_loss: 1.2198 - val_accuracy:
0.6358 - val_precision: 0.7527 - val_recall: 0.5557 - lr: 3.0000e-04
Epoch 81/200
66/66 [==============================] - 77s 1s/step - loss: 0.8962 - accuracy:
0.6821 - precision: 0.8208 - recall: 0.5531 - val_loss: 1.3025 - val_accuracy:
0.6003 - val_precision: 0.7287 - val_recall: 0.4699 - lr: 3.0000e-04
Epoch 82/200
66/66 [==============================] - 77s 1s/step - loss: 0.8944 - accuracy:
0.6861 - precision: 0.8226 - recall: 0.5646 - val_loss: 1.1457 - val_accuracy:
0.6553 - val_precision: 0.7772 - val_recall: 0.5325 - lr: 3.0000e-04
Epoch 83/200
66/66 [==============================] - 77s 1s/step - loss: 0.8946 - accuracy:
0.6853 - precision: 0.8148 - recall: 0.5566 - val_loss: 1.2499 - val_accuracy:
0.6311 - val_precision: 0.7351 - val_recall: 0.5026 - lr: 3.0000e-04
Epoch 84/200
66/66 [==============================] - ETA: 0s - loss: 0.8740 - accuracy:
0.6919 - precision: 0.8293 - recall: 0.5647
Epoch 84: ReduceLROnPlateau reducing learning rate to 9.000000427477062e-05.
66/66 [==============================] - 77s 1s/step - loss: 0.8740 - accuracy:
0.6919 - precision: 0.8293 - recall: 0.5647 - val_loss: 1.1538 - val_accuracy:
0.6496 - val_precision: 0.7799 - val_recall: 0.5543 - lr: 3.0000e-04
Epoch 85/200
66/66 [==============================] - 76s 1s/step - loss: 0.8314 - accuracy:
0.7085 - precision: 0.8410 - recall: 0.5789 - val_loss: 1.1731 - val_accuracy:
0.6572 - val_precision: 0.7531 - val_recall: 0.5756 - lr: 9.0000e-05
Epoch 86/200
66/66 [==============================] - 77s 1s/step - loss: 0.8278 - accuracy:
0.7143 - precision: 0.8442 - recall: 0.5789 - val_loss: 1.1355 - val_accuracy:
0.6638 - val_precision: 0.7749 - val_recall: 0.5581 - lr: 9.0000e-05
Epoch 87/200
```

```
66/66 [==============================] - 78s 1s/step - loss: 0.8256 - accuracy:
0.7172 - precision: 0.8473 - recall: 0.5864 - val_loss: 1.1263 - val_accuracy:
0.6586 - val_precision: 0.7904 - val_recall: 0.5543 - lr: 9.0000e-05
Epoch 88/200
66/66 [==============================] - 77s 1s/step - loss: 0.8343 - accuracy:
0.7058 - precision: 0.8372 - recall: 0.5757 - val_loss: 1.1467 - val_accuracy:
0.6633 - val_precision: 0.7672 - val_recall: 0.5671 - lr: 9.0000e-05
Epoch 89/200
66/66 [==============================] - 76s 1s/step - loss: 0.8318 - accuracy:
0.7094 - precision: 0.8382 - recall: 0.5859 - val_loss: 1.1397 - val_accuracy:
0.6539 - val_precision: 0.7758 - val_recall: 0.5481 - lr: 9.0000e-05
Epoch 90/200
66/66 [==============================] - 78s 1s/step - loss: 0.8291 - accuracy:
0.7086 - precision: 0.8351 - recall: 0.5902 - val_loss: 1.1567 - val_accuracy:
0.6534 - val_precision: 0.7586 - val_recall: 0.5633 - lr: 9.0000e-05
Epoch 91/200
66/66 [==============================] - 77s 1s/step - loss: 0.8109 - accuracy:
0.7187 - precision: 0.8411 - recall: 0.5911 - val_loss: 1.1688 - val_accuracy:
0.6586 - val_precision: 0.7598 - val_recall: 0.5624 - lr: 9.0000e-05
Epoch 92/200
66/66 [==============================] - 78s 1s/step - loss: 0.8311 - accuracy:
0.7039 - precision: 0.8320 - recall: 0.5783 - val_loss: 1.1400 - val_accuracy:
0.6619 - val_precision: 0.7797 - val_recall: 0.5657 - lr: 9.0000e-05
Epoch 93/200
66/66 [==============================] - ETA: 0s - loss: 0.8267 - accuracy:
0.7110 - precision: 0.8385 - recall: 0.5831
Epoch 93: ReduceLROnPlateau reducing learning rate to 2.700000040931627e-05.
66/66 [==============================] - 78s 1s/step - loss: 0.8267 - accuracy:
0.7110 - precision: 0.8385 - recall: 0.5831 - val_loss: 1.1486 - val_accuracy:
0.6577 - val_precision: 0.7676 - val_recall: 0.5467 - lr: 9.0000e-05
Epoch 94/200
66/66 [==============================] - 77s 1s/step - loss: 0.8167 - accuracy:
0.7154 - precision: 0.8441 - recall: 0.5941 - val_loss: 1.1465 - val_accuracy:
0.6619 - val_precision: 0.7708 - val_recall: 0.5676 - lr: 2.7000e-05
Epoch 95/200
66/66 [==============================] - 78s 1s/step - loss: 0.7941 - accuracy:
0.7222 - precision: 0.8429 - recall: 0.5968 - val_loss: 1.1424 - val_accuracy:
0.6610 - val_precision: 0.7804 - val_recall: 0.5728 - lr: 2.7000e-05
Epoch 96/200
66/66 [==============================] - 78s 1s/step - loss: 0.8141 - accuracy:
0.7140 - precision: 0.8463 - recall: 0.5948 - val_loss: 1.1338 - val_accuracy:
0.6605 - val_precision: 0.7682 - val_recall: 0.5657 - lr: 2.7000e-05
Epoch 97/200
66/66 [==============================] - 77s 1s/step - loss: 0.7898 - accuracy:
0.7220 - precision: 0.8453 - recall: 0.6067 - val_loss: 1.1450 - val_accuracy:
0.6676 - val_precision: 0.7716 - val_recall: 0.5766 - lr: 2.7000e-05
Epoch 98/200
66/66 [==============================] - 77s 1s/step - loss: 0.8060 - accuracy:
```

```
0.7196 - precision: 0.8452 - recall: 0.5924 - val_loss: 1.1306 - val_accuracy:
0.6629 - val_precision: 0.7737 - val_recall: 0.5642 - lr: 2.7000e-05
Epoch 99/200
66/66 [==============================] - 77s 1s/step - loss: 0.8030 - accuracy:
0.7210 - precision: 0.8508 - recall: 0.5972 - val_loss: 1.1357 - val_accuracy:
0.6619 - val_precision: 0.7725 - val_recall: 0.5666 - lr: 2.7000e-05
Epoch 100/200
66/66 [==============================] - 77s 1s/step - loss: 0.8043 - accuracy:
0.7242 - precision: 0.8462 - recall: 0.5963 - val_loss: 1.1420 - val_accuracy:
0.6596 - val_precision: 0.7755 - val_recall: 0.5633 - lr: 2.7000e-05
Epoch 101/200
66/66 [==============================] - 78s 1s/step - loss: 0.8055 - accuracy:
0.7139 - precision: 0.8480 - recall: 0.5961 - val_loss: 1.1327 - val_accuracy:
0.6652 - val_precision: 0.7726 - val_recall: 0.5718 - lr: 2.7000e-05
Epoch 102/200
66/66 [==============================] - 78s 1s/step - loss: 0.8057 - accuracy:
0.7184 - precision: 0.8477 - recall: 0.5883 - val_loss: 1.1455 - val_accuracy:
0.6577 - val_precision: 0.7700 - val_recall: 0.5652 - lr: 2.7000e-05
Epoch 103/200
66/66 [==============================] - 77s 1s/step - loss: 0.8030 - accuracy:
0.7212 - precision: 0.8453 - recall: 0.5966 - val_loss: 1.1279 - val_accuracy:
0.6633 - val_precision: 0.7778 - val_recall: 0.5695 - lr: 2.7000e-05
Epoch 104/200
66/66 [==============================] - ETA: 0s - loss: 0.7922 - accuracy:
0.7211 - precision: 0.8459 - recall: 0.6014
Epoch 104: ReduceLROnPlateau reducing learning rate to 8.100000013655517e-06.
66/66 [==============================] - 78s 1s/step - loss: 0.7922 - accuracy:
0.7211 - precision: 0.8459 - recall: 0.6014 - val_loss: 1.1447 - val_accuracy:
0.6596 - val_precision: 0.7662 - val_recall: 0.5704 - lr: 2.7000e-05
Epoch 105/200
66/66 [==============================] - 77s 1s/step - loss: 0.7936 - accuracy:
0.7238 - precision: 0.8546 - recall: 0.6042 - val_loss: 1.1341 - val_accuracy:
0.6581 - val_precision: 0.7730 - val_recall: 0.5699 - lr: 8.1000e-06
Epoch 106/200
66/66 [==============================] - 78s 1s/step - loss: 0.7899 - accuracy:
0.7216 - precision: 0.8527 - recall: 0.6025 - val_loss: 1.1398 - val_accuracy:
0.6643 - val_precision: 0.7731 - val_recall: 0.5718 - lr: 8.1000e-06
Epoch 107/200
66/66 [==============================] - 78s 1s/step - loss: 0.7952 - accuracy:
0.7200 - precision: 0.8429 - recall: 0.5993 - val_loss: 1.1320 - val_accuracy:
0.6619 - val_precision: 0.7771 - val_recall: 0.5718 - lr: 8.1000e-06
Epoch 108/200
66/66 [==============================] - 77s 1s/step - loss: 0.8015 - accuracy:
0.7166 - precision: 0.8430 - recall: 0.5958 - val_loss: 1.1338 - val_accuracy:
0.6605 - val_precision: 0.7761 - val_recall: 0.5737 - lr: 8.1000e-06
Epoch 109/200
66/66 [==============================] - 77s 1s/step - loss: 0.7862 - accuracy:
0.7224 - precision: 0.8483 - recall: 0.6033 - val_loss: 1.1386 - val_accuracy:
```

```
0.6577 - val_precision: 0.7766 - val_recall: 0.5704 - lr: 8.1000e-06
Epoch 110/200
66/66 [==============================] - 78s 1s/step - loss: 0.7902 - accuracy:
0.7217 - precision: 0.8479 - recall: 0.6017 - val_loss: 1.1405 - val_accuracy:
0.6600 - val_precision: 0.7698 - val_recall: 0.5709 - lr: 8.1000e-06
Epoch 111/200
66/66 [==============================] - ETA: 0s - loss: 0.7872 - accuracy:
0.7235 - precision: 0.8449 - recall: 0.6031
Epoch 111: ReduceLROnPlateau reducing learning rate to 2.429999949526973e-06.
66/66 [==============================] - 78s 1s/step - loss: 0.7872 - accuracy:
0.7235 - precision: 0.8449 - recall: 0.6031 - val_loss: 1.1344 - val_accuracy:
0.6577 - val_precision: 0.7709 - val_recall: 0.5728 - lr: 8.1000e-06
Epoch 112/200
66/66 [==============================] - 77s 1s/step - loss: 0.7953 - accuracy:
0.7187 - precision: 0.8494 - recall: 0.5992 - val_loss: 1.1375 - val_accuracy:
0.6619 - val_precision: 0.7743 - val_recall: 0.5742 - lr: 2.4300e-06
Epoch 113/200
66/66 [==============================] - 77s 1s/step - loss: 0.7815 - accuracy:
0.7303 - precision: 0.8517 - recall: 0.6102 - val_loss: 1.1384 - val_accuracy:
0.6619 - val_precision: 0.7748 - val_recall: 0.5742 - lr: 2.4300e-06
Epoch 114/200
66/66 [==============================] - 77s 1s/step - loss: 0.7911 - accuracy:
0.7218 - precision: 0.8461 - recall: 0.5985 - val_loss: 1.1383 - val_accuracy:
0.6596 - val_precision: 0.7733 - val_recall: 0.5742 - lr: 2.4300e-06
Epoch 115/200
66/66 [==============================] - 77s 1s/step - loss: 0.8008 - accuracy:
0.7191 - precision: 0.8390 - recall: 0.6012 - val_loss: 1.1375 - val_accuracy:
0.6610 - val_precision: 0.7712 - val_recall: 0.5723 - lr: 2.4300e-06
Epoch 116/200
66/66 [==============================] - 77s 1s/step - loss: 0.7898 - accuracy:
0.7213 - precision: 0.8466 - recall: 0.5958 - val_loss: 1.1352 - val_accuracy:
0.6615 - val_precision: 0.7745 - val_recall: 0.5733 - lr: 2.4300e-06
Epoch 117/200
66/66 [==============================] - ETA: 0s - loss: 0.7999 - accuracy:
0.7199 - precision: 0.8450 - recall: 0.5979Restoring model weights from the end
of the best epoch: 97.
66/66 [==============================] - 77s 1s/step - loss: 0.7999 - accuracy:
0.7199 - precision: 0.8450 - recall: 0.5979 - val_loss: 1.1352 - val_accuracy:
0.6596 - val_precision: 0.7742 - val_recall: 0.5737 - lr: 2.4300e-06
Epoch 117: early stopping
MobileNetV3Large Training time: 150.0 min 21.22 sec
```

[ ]:
```
def f1(prec_list: List[float], rec_list: List[float]) -> List[float]:
    return [
        2 * (prec * rec) / (prec + rec) if prec + rec != 0.0 else 0.0
        for (prec, rec) in zip(prec_list, rec_list)
    ]
```

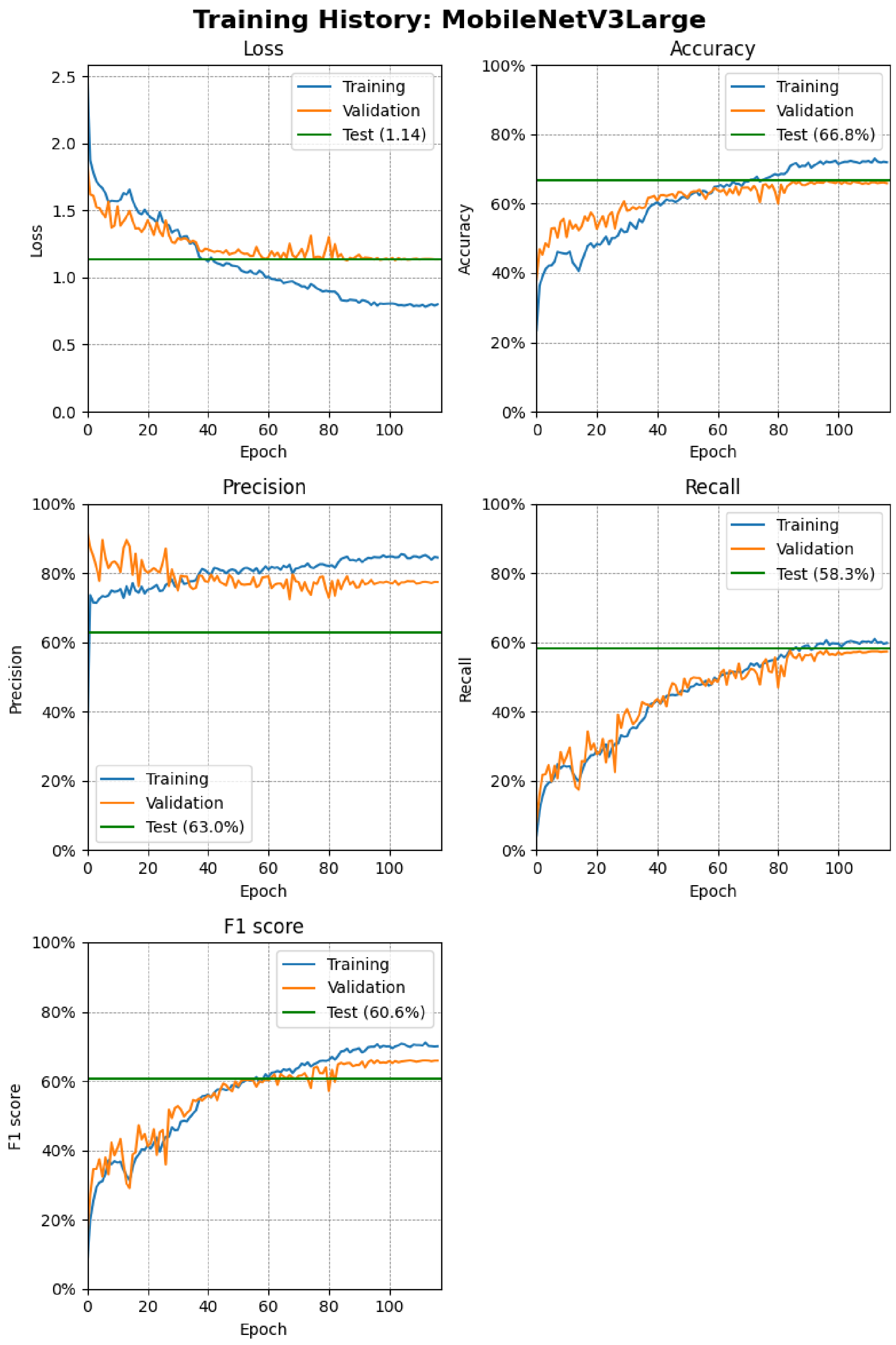
```
def get_predictions(model: Model, generator: ImageDataGenerator) -> Tuple[np.
 ↪ndarray, np.ndarray]:
    pred = model.predict(generator)
    pred_labels = np.argmax(pred, axis=1)   # Turn "one-hot encoding"-like␣
 ↪probabilities
                                            # into integer encoding for easier␣
 ↪comparison
                                            # with true labels
    return pred, pred_labels

def quality_scores(pred_labels: np.ndarray, predictions: np.ndarray, generator:␣
 ↪ImageDataGenerator) -> Dict[str, float]:
    loss_func = tensorflow.keras.losses.SparseCategoricalCrossentropy()
    retval = {}
    retval['accuracy'] = sklearn.metrics.accuracy_score(generator.labels,␣
 ↪pred_labels)
    retval['precision'] = sklearn.metrics.precision_score(generator.labels,␣
 ↪pred_labels, average='macro', zero_division=0)
    retval['recall'] = sklearn.metrics.recall_score(generator.labels,␣
 ↪pred_labels, average='macro')
    retval['f1_score'] = 2 * (retval['precision'] * retval['recall']) /␣
 ↪(retval['precision'] + retval['recall'])
    retval['loss'] = loss_func(np.array(generator.labels), predictions)
    return retval

def training_subplot(hist, metric: str, plotnum: int, lim = None, test_val =␣
 ↪None):
    sp = plt.subplot(3, 2, plotnum)
    metric_nm = metric.replace('_',' ').capitalize()
    plt.plot(hist.history[metric], label='Training')
    plt.plot(hist.history['val_' + metric], label='Validation')
    if (test_val is not None):
        test_lbl = 'Test ' + (f'({test_val:.1%})' if (lim == 1) else␣
 ↪f'({test_val:.2f})')
        plt.axhline(y=test_val, label=test_lbl, color='green', linestyle='-')
    plt.xlabel('Epoch')
    plt.ylabel(metric_nm)
    plt.ylim(0, lim)
    plt.xlim(0, len(hist.history['loss']))
    if (lim == 1):
        sp.yaxis.set_major_formatter(ticker.PercentFormatter(xmax=1.0))
    plt.legend()
    plt.grid(visible=True, which='both', axis='both', linestyle='--',␣
 ↪linewidth=0.5, color='grey')
    plt.title(metric_nm)
```

```
def training_plot(hist, predicted_labels: np.ndarray, predictions: np.ndarray,␣
↪generator: ImageDataGenerator, title='Training History'):
    quality = quality_scores(predicted_labels, predictions, generator)
    # Compute the f1 for training and validation
    hist.history['f1_score'] = f1(hist.history['precision'], hist.
↪history['recall'])
    hist.history['val_f1_score'] = f1(hist.history['val_precision'], hist.
↪history['val_recall'])
    # Plot the training history (loss and accuracy) in two subplots
    plt.figure(figsize=(8, 12))
    plt.suptitle(title, fontsize=16, fontweight='bold')
    training_subplot(hist, 'loss', 1, test_val=quality['loss'])
    training_subplot(hist, 'accuracy', 2, 1, test_val=quality['accuracy'])
    training_subplot(hist, 'precision', 3, 1, test_val=quality['precision'])
    training_subplot(hist, 'recall', 4, 1, test_val=quality['recall'])
    training_subplot(hist, 'f1_score', 5, 1, test_val=quality['f1_score'])
    plt.tight_layout()
    plt.show()
```

[ ]:
```
predictions, predicted_labels = get_predictions(model_mnv3l, test_generator)
```

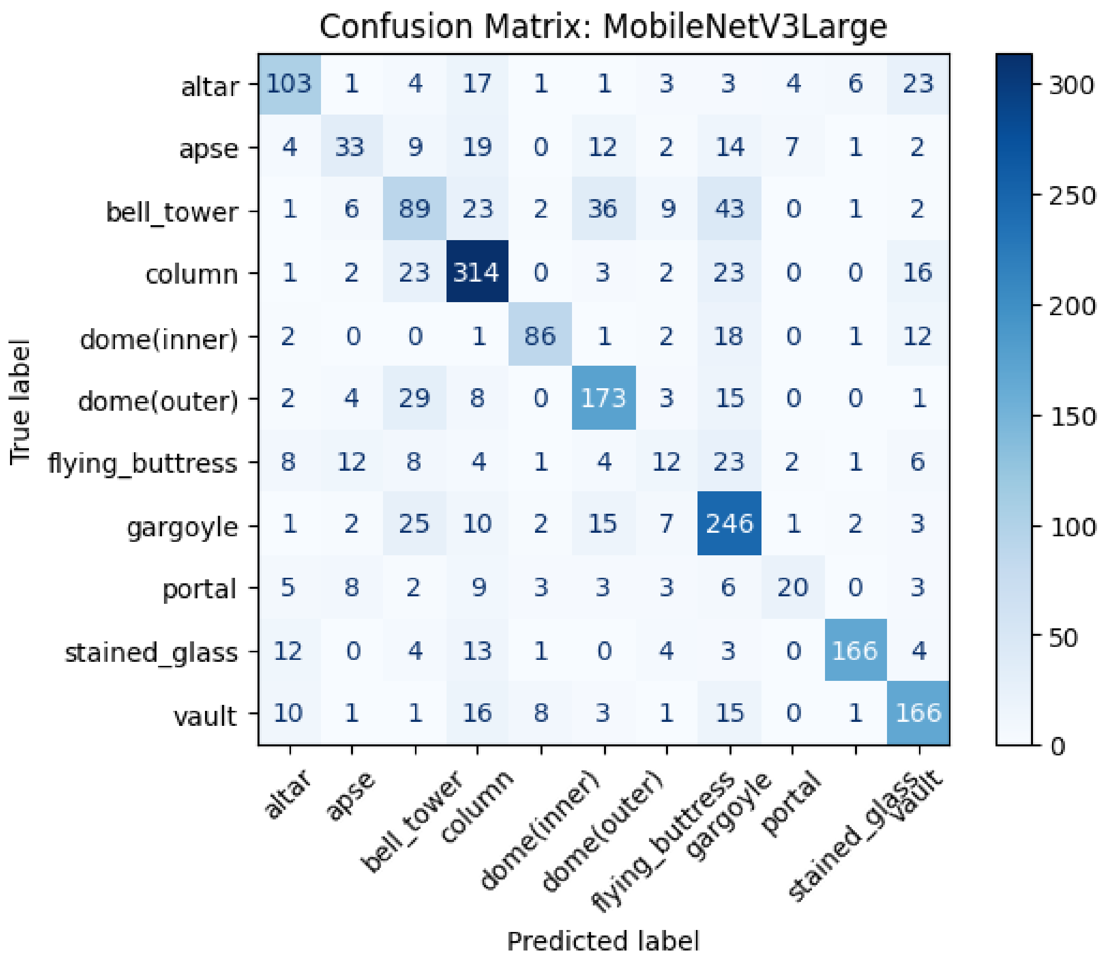
```
17/17 [==============================] - 16s 864ms/step
```

[ ]:
```
training_plot(history, predicted_labels, predictions, test_generator, "Training␣
↪History: MobileNetV3Large")
```


Training History: MobileNetV3Large
Loss
Training
Validation
Test (1.14)
Loss
Epoch
Accuracy
Training
Validation
Test (66.8%)
Accuracy
Epoch
Precision
Training
Validation
Test (63.0%)
Precision
Epoch
Recall
Training
Validation
Test (58.3%)
Recall
Epoch
F1 score
Training
Validation
Test (60.6%)
F1 score
Epoch

```
[ ]: from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,␣
     ↪classification_report

     def confusion_matrix_and_classification_report(predicted_labels: np.ndarray,␣
     ↪generator: tensorflow.keras.preprocessing.image.DirectoryIterator, title:␣
     ↪str):
         class_names = generator.class_indices.keys()
         # Compute the confusion matrix
         cm = confusion_matrix(generator.labels, predicted_labels)
         # Plot the confusion matrix
         cm_display = ConfusionMatrixDisplay(cm, display_labels=class_names)
         cm_display.plot(cmap='Blues', values_format='d', xticks_rotation=45)
         plt.title('Confusion Matrix: ' + title)
         plt.show()
         # Display the classification report
         print(classification_report(generator.labels, predicted_labels,␣
     ↪target_names=class_names, zero_division=0))
```

```
[ ]: confusion_matrix_and_classification_report(predicted_labels, test_generator,␣
     ↪'MobileNetV3Large')
```



```
                 precision    recall  f1-score   support

          altar       0.69      0.62      0.65       166
           apse       0.48      0.32      0.38       103
     bell_tower       0.46      0.42      0.44       212
         column       0.72      0.82      0.77       384
    dome(inner)       0.83      0.70      0.76       123
    dome(outer)       0.69      0.74      0.71       235
flying_buttress       0.25      0.15      0.19        81
       gargoyle       0.60      0.78      0.68       314
         portal       0.59      0.32      0.42        62
  stained_glass       0.93      0.80      0.86       207
          vault       0.70      0.75      0.72       222

       accuracy                           0.67      2109
      macro avg       0.63      0.58      0.60      2109
   weighted avg       0.66      0.67      0.66      2109
```

```
[ ]: model_mnv3s = build_or_load_model(
         tensorflow.keras.applications.MobileNetV3Small(include_top=False,␣
      ↪weights='imagenet', input_shape=(224, 224, 3)),
         len(df_train['label'].unique()),
         name='MobileNetV3Small')
```

```
Model: "MobileNetV3Small"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 MobilenetV3small (Function  (None, 7, 7, 576)         939120
 al)

 flatten_5 (Flatten)         (None, 28224)             0

 dense_15 (Dense)            (None, 256)               7225600

 dropout_10 (Dropout)        (None, 256)               0

 dense_16 (Dense)            (None, 128)               32896

 dropout_11 (Dropout)        (None, 128)               0

 dense_17 (Dense)            (None, 11)                1419

=================================================================
Total params: 8199035 (31.28 MB)
Trainable params: 7259915 (27.69 MB)
Non-trainable params: 939120 (3.58 MB)
_________________________________________________________________
```

```
[ ]: history_mnv3s = model_fit(model_mnv3s, train_generator, valid_generator,␣
      ↪max_epochs, callbacks())
```

```
Epoch 1/200
66/66 [==============================] - 58s 827ms/step - loss: 2.7858 -
accuracy: 0.1902 - precision: 0.1430 - recall: 0.0164 - val_loss: 2.0914 -
val_accuracy: 0.2764 - val_precision: 0.6667 - val_recall: 9.4832e-04 - lr:
0.0010
Epoch 2/200
66/66 [==============================] - 52s 785ms/step - loss: 2.0783 -
accuracy: 0.2826 - precision: 0.7198 - recall: 0.0439 - val_loss: 1.9664 -
val_accuracy: 0.3395 - val_precision: 0.9167 - val_recall: 0.0522 - lr: 0.0010
Epoch 3/200
66/66 [==============================] - 52s 789ms/step - loss: 1.9901 -
accuracy: 0.3094 - precision: 0.7309 - recall: 0.0828 - val_loss: 1.8600 -
val_accuracy: 0.3656 - val_precision: 0.8750 - val_recall: 0.1128 - lr: 0.0010
```

```
Epoch 4/200
66/66 [==============================] - 52s 792ms/step - loss: 1.9597 -
accuracy: 0.3284 - precision: 0.7239 - recall: 0.1029 - val_loss: 1.7928 -
val_accuracy: 0.3902 - val_precision: 0.8679 - val_recall: 0.1527 - lr: 0.0010
Epoch 5/200
66/66 [==============================] - 52s 789ms/step - loss: 1.9429 -
accuracy: 0.3242 - precision: 0.7290 - recall: 0.1062 - val_loss: 1.7791 -
val_accuracy: 0.4139 - val_precision: 0.8855 - val_recall: 0.1100 - lr: 0.0010
Epoch 6/200
66/66 [==============================] - 53s 797ms/step - loss: 1.9157 -
accuracy: 0.3329 - precision: 0.7005 - recall: 0.1245 - val_loss: 1.7199 -
val_accuracy: 0.4414 - val_precision: 0.9130 - val_recall: 0.1195 - lr: 0.0010
Epoch 7/200
66/66 [==============================] - 53s 796ms/step - loss: 1.8599 -
accuracy: 0.3623 - precision: 0.7333 - recall: 0.1383 - val_loss: 1.7423 -
val_accuracy: 0.4462 - val_precision: 0.8883 - val_recall: 0.1735 - lr: 0.0010
Epoch 8/200
66/66 [==============================] - 52s 792ms/step - loss: 1.8474 -
accuracy: 0.3651 - precision: 0.7194 - recall: 0.1450 - val_loss: 1.6762 -
val_accuracy: 0.4623 - val_precision: 0.8698 - val_recall: 0.1679 - lr: 0.0010
Epoch 9/200
66/66 [==============================] - 52s 790ms/step - loss: 1.8453 -
accuracy: 0.3640 - precision: 0.7322 - recall: 0.1559 - val_loss: 1.6829 -
val_accuracy: 0.4922 - val_precision: 0.9373 - val_recall: 0.1275 - lr: 0.0010
Epoch 10/200
66/66 [==============================] - 52s 788ms/step - loss: 1.7991 -
accuracy: 0.3799 - precision: 0.7232 - recall: 0.1608 - val_loss: 1.5863 -
val_accuracy: 0.5064 - val_precision: 0.9022 - val_recall: 0.1750 - lr: 0.0010
Epoch 11/200
66/66 [==============================] - 52s 790ms/step - loss: 1.7861 -
accuracy: 0.3828 - precision: 0.7495 - recall: 0.1721 - val_loss: 1.6402 -
val_accuracy: 0.4836 - val_precision: 0.8932 - val_recall: 0.1745 - lr: 0.0010
Epoch 12/200
66/66 [==============================] - 52s 785ms/step - loss: 1.7855 -
accuracy: 0.3867 - precision: 0.7214 - recall: 0.1781 - val_loss: 1.5629 -
val_accuracy: 0.5254 - val_precision: 0.9056 - val_recall: 0.1683 - lr: 0.0010
Epoch 13/200
66/66 [==============================] - 52s 792ms/step - loss: 1.7940 -
accuracy: 0.3820 - precision: 0.7299 - recall: 0.1698 - val_loss: 1.8369 -
val_accuracy: 0.4111 - val_precision: 0.8416 - val_recall: 0.1361 - lr: 0.0010
Epoch 14/200
66/66 [==============================] - 52s 793ms/step - loss: 1.9700 -
accuracy: 0.3131 - precision: 0.7080 - recall: 0.1191 - val_loss: 1.8348 -
val_accuracy: 0.3736 - val_precision: 0.9198 - val_recall: 0.0925 - lr: 0.0010
Epoch 15/200
66/66 [==============================] - 52s 796ms/step - loss: 1.9146 -
accuracy: 0.3158 - precision: 0.7268 - recall: 0.1262 - val_loss: 1.7847 -
val_accuracy: 0.4144 - val_precision: 0.9065 - val_recall: 0.1057 - lr: 0.0010
```

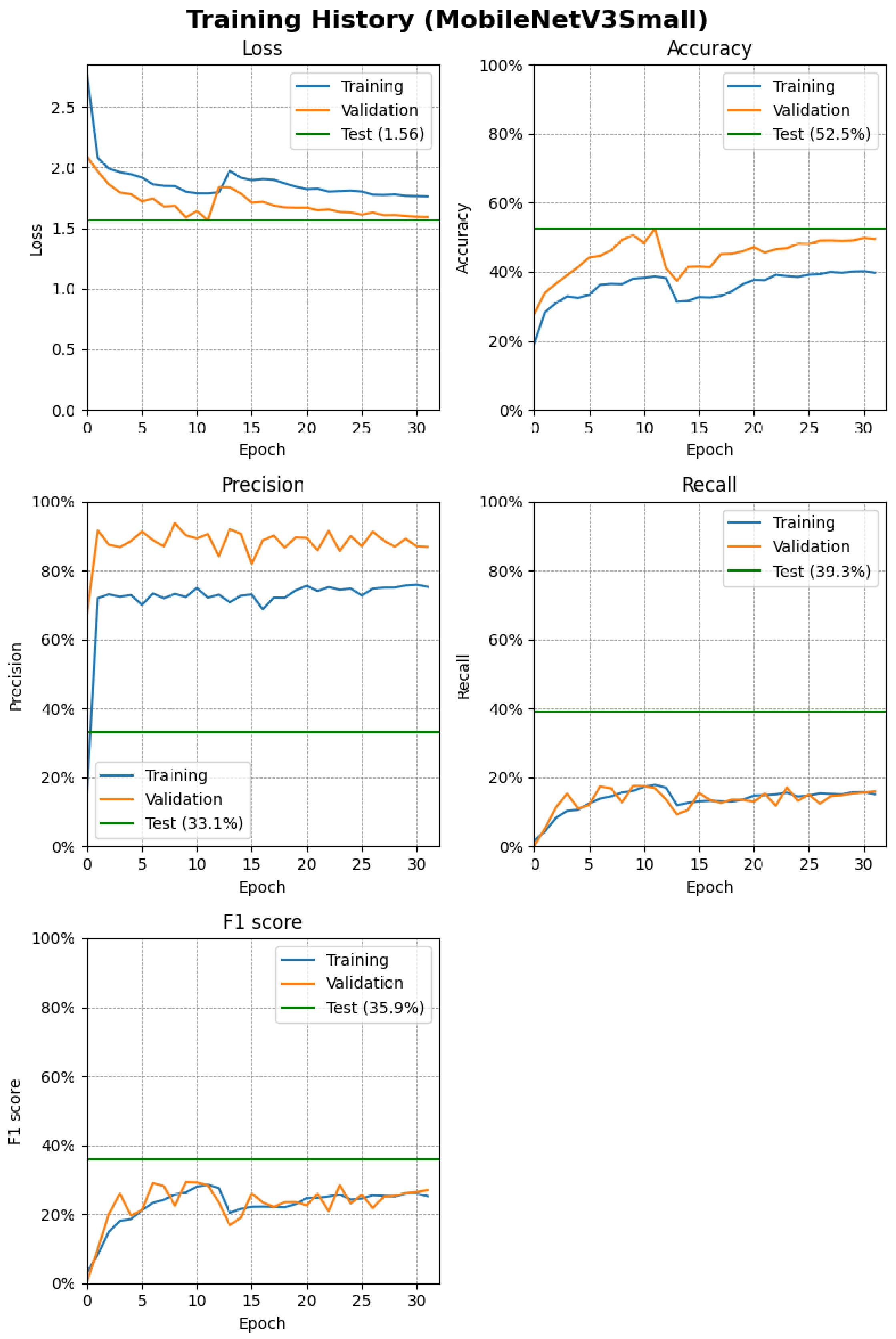
```
Epoch 16/200
66/66 [==============================] - 52s 784ms/step - loss: 1.8939 -
accuracy: 0.3265 - precision: 0.7307 - recall: 0.1303 - val_loss: 1.7093 -
val_accuracy: 0.4154 - val_precision: 0.8191 - val_recall: 0.1546 - lr: 0.0010
Epoch 17/200
66/66 [==============================] - 52s 791ms/step - loss: 1.9035 -
accuracy: 0.3253 - precision: 0.6877 - recall: 0.1321 - val_loss: 1.7162 -
val_accuracy: 0.4135 - val_precision: 0.8875 - val_recall: 0.1347 - lr: 0.0010
Epoch 18/200
66/66 [==============================] - 52s 792ms/step - loss: 1.8971 -
accuracy: 0.3297 - precision: 0.7213 - recall: 0.1304 - val_loss: 1.6851 -
val_accuracy: 0.4509 - val_precision: 0.9014 - val_recall: 0.1257 - lr: 0.0010
Epoch 19/200
66/66 [==============================] - ETA: 0s - loss: 1.8678 - accuracy:
0.3434 - precision: 0.7212 - recall: 0.1297
Epoch 19: ReduceLROnPlateau reducing learning rate to 0.0003000000142492354.
66/66 [==============================] - 52s 790ms/step - loss: 1.8678 -
accuracy: 0.3434 - precision: 0.7212 - recall: 0.1297 - val_loss: 1.6698 -
val_accuracy: 0.4523 - val_precision: 0.8667 - val_recall: 0.1356 - lr: 0.0010
Epoch 20/200
66/66 [==============================] - 52s 786ms/step - loss: 1.8406 -
accuracy: 0.3640 - precision: 0.7425 - recall: 0.1354 - val_loss: 1.6672 -
val_accuracy: 0.4595 - val_precision: 0.8962 - val_recall: 0.1351 - lr:
3.0000e-04
Epoch 21/200
66/66 [==============================] - 52s 794ms/step - loss: 1.8204 -
accuracy: 0.3765 - precision: 0.7556 - recall: 0.1470 - val_loss: 1.6676 -
val_accuracy: 0.4718 - val_precision: 0.8947 - val_recall: 0.1290 - lr:
3.0000e-04
Epoch 22/200
66/66 [==============================] - 52s 786ms/step - loss: 1.8242 -
accuracy: 0.3758 - precision: 0.7405 - recall: 0.1482 - val_loss: 1.6469 -
val_accuracy: 0.4557 - val_precision: 0.8587 - val_recall: 0.1527 - lr:
3.0000e-04
Epoch 23/200
66/66 [==============================] - 52s 788ms/step - loss: 1.7999 -
accuracy: 0.3913 - precision: 0.7521 - recall: 0.1507 - val_loss: 1.6536 -
val_accuracy: 0.4656 - val_precision: 0.9151 - val_recall: 0.1176 - lr:
3.0000e-04
Epoch 24/200
66/66 [==============================] - 52s 787ms/step - loss: 1.8044 -
accuracy: 0.3876 - precision: 0.7442 - recall: 0.1556 - val_loss: 1.6323 -
val_accuracy: 0.4685 - val_precision: 0.8568 - val_recall: 0.1702 - lr:
3.0000e-04
Epoch 25/200
66/66 [==============================] - 52s 789ms/step - loss: 1.8080 -
accuracy: 0.3853 - precision: 0.7477 - recall: 0.1444 - val_loss: 1.6282 -
val_accuracy: 0.4817 - val_precision: 0.9000 - val_recall: 0.1323 - lr:
```

```
3.0000e-04
Epoch 26/200
66/66 [==============================] - ETA: 0s - loss: 1.8004 - accuracy:
0.3917 - precision: 0.7276 - recall: 0.1473
Epoch 26: ReduceLROnPlateau reducing learning rate to 9.000000427477062e-05.
66/66 [==============================] - 52s 789ms/step - loss: 1.8004 -
accuracy: 0.3917 - precision: 0.7276 - recall: 0.1473 - val_loss: 1.6100 -
val_accuracy: 0.4808 - val_precision: 0.8709 - val_recall: 0.1503 - lr:
3.0000e-04
Epoch 27/200
66/66 [==============================] - 52s 790ms/step - loss: 1.7748 -
accuracy: 0.3933 - precision: 0.7474 - recall: 0.1537 - val_loss: 1.6277 -
val_accuracy: 0.4903 - val_precision: 0.9126 - val_recall: 0.1238 - lr:
9.0000e-05
Epoch 28/200
66/66 [==============================] - 52s 785ms/step - loss: 1.7726 -
accuracy: 0.3996 - precision: 0.7504 - recall: 0.1519 - val_loss: 1.6053 -
val_accuracy: 0.4908 - val_precision: 0.8873 - val_recall: 0.1456 - lr:
9.0000e-05
Epoch 29/200
66/66 [==============================] - 52s 792ms/step - loss: 1.7770 -
accuracy: 0.3971 - precision: 0.7506 - recall: 0.1506 - val_loss: 1.6069 -
val_accuracy: 0.4893 - val_precision: 0.8691 - val_recall: 0.1479 - lr:
9.0000e-05
Epoch 30/200
66/66 [==============================] - 52s 790ms/step - loss: 1.7654 -
accuracy: 0.4008 - precision: 0.7567 - recall: 0.1568 - val_loss: 1.5985 -
val_accuracy: 0.4908 - val_precision: 0.8923 - val_recall: 0.1532 - lr:
9.0000e-05
Epoch 31/200
66/66 [==============================] - 52s 794ms/step - loss: 1.7627 -
accuracy: 0.4018 - precision: 0.7589 - recall: 0.1571 - val_loss: 1.5919 -
val_accuracy: 0.4979 - val_precision: 0.8704 - val_recall: 0.1560 - lr:
9.0000e-05
Epoch 32/200
66/66 [==============================] - ETA: 0s - loss: 1.7603 - accuracy:
0.3974 - precision: 0.7528 - recall: 0.1513Restoring model weights from the end
of the best epoch: 12.
66/66 [==============================] - 52s 782ms/step - loss: 1.7603 -
accuracy: 0.3974 - precision: 0.7528 - recall: 0.1513 - val_loss: 1.5902 -
val_accuracy: 0.4950 - val_precision: 0.8686 - val_recall: 0.1598 - lr:
9.0000e-05
Epoch 32: early stopping
MobileNetV3Small Training time: 27.0 min 54.10 sec
```

[ ]:
```
predictions_mnv3s, predicted_labels_mnv3s = get_predictions(model_mnv3s,␣
↪test_generator)
```

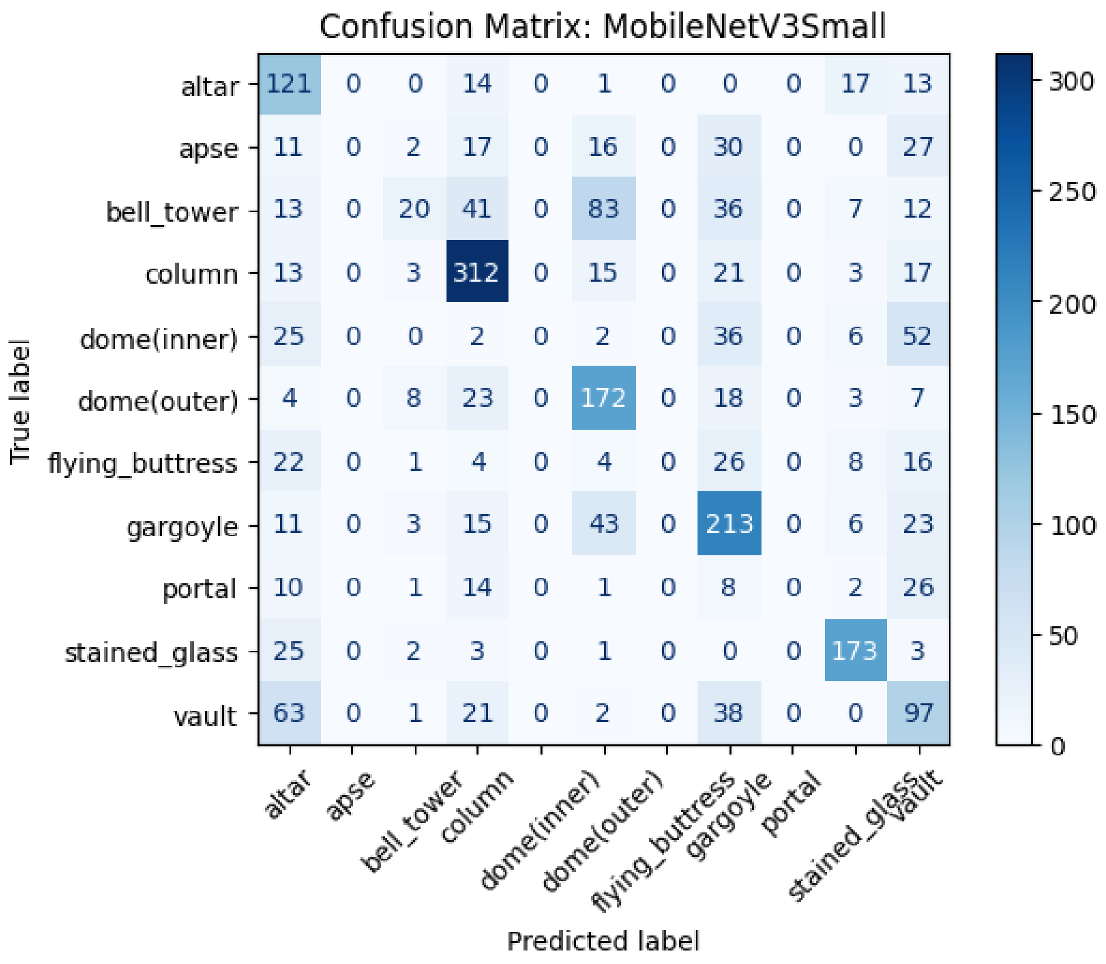
```
17/17 [==============================] - 11s 588ms/step
```

[ ]:
```
training_plot(history_mnv3s, predicted_labels_mnv3s, predictions_mnv3s,␣
↪test_generator, "Training History (MobileNetV3Small)")
```



Training History (MobileNetV3Small)
Loss
Accuracy
Precision
Recall
F1 score
Training
Validation
Test (1.56)
Test (52.5%)
Test (33.1%)
Test (39.3%)
Test (35.9%)
Epoch

```
[ ]: confusion_matrix_and_classification_report(predicted_labels_mnv3s,␣
     ↪test_generator, 'MobileNetV3Small')
```



```
                 precision    recall  f1-score   support

          altar       0.38      0.73      0.50       166
           apse       0.00      0.00      0.00       103
     bell_tower       0.49      0.09      0.16       212
         column       0.67      0.81      0.73       384
    dome(inner)       0.00      0.00      0.00       123
    dome(outer)       0.51      0.73      0.60       235
flying_buttress       0.00      0.00      0.00        81
       gargoyle       0.50      0.68      0.58       314
         portal       0.00      0.00      0.00        62
  stained_glass       0.77      0.84      0.80       207
```

```
       vault       0.33      0.44      0.38       222

    accuracy                           0.53      2109
   macro avg       0.33      0.39      0.34      2109
weighted avg       0.44      0.53      0.46      2109
```

[ ]:
```
model_vgg16 = build_or_load_model(
    tensorflow.keras.applications.VGG16(include_top=False, weights='imagenet',␣
 ↪input_shape=(224, 224, 3)),
    len(df_train['label'].unique()),
    name='VGG16')
```

```
Model: "VGG16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 flatten_6 (Flatten)         (None, 25088)             0

 dense_18 (Dense)            (None, 256)               6422784

 dropout_12 (Dropout)        (None, 256)               0

 dense_19 (Dense)            (None, 128)               32896

 dropout_13 (Dropout)        (None, 128)               0

 dense_20 (Dense)            (None, 11)                1419

=================================================================
Total params: 21171787 (80.76 MB)
Trainable params: 6457099 (24.63 MB)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
```

[ ]:
```
history_vgg16 = model_fit(model_vgg16, train_generator, valid_generator,␣
 ↪max_epochs, callbacks())
```

```
Epoch 1/200
66/66 [==============================] - 181s 3s/step - loss: 0.9449 - accuracy:
0.7172 - precision: 0.8212 - recall: 0.6309 - val_loss: 0.3174 - val_accuracy:
0.9099 - val_precision: 0.9449 - val_recall: 0.8620 - lr: 0.0010
Epoch 2/200
66/66 [==============================] - 181s 3s/step - loss: 0.3573 - accuracy:
0.8914 - precision: 0.9258 - recall: 0.8534 - val_loss: 0.2751 - val_accuracy:
0.9208 - val_precision: 0.9392 - val_recall: 0.8938 - lr: 0.0010
Epoch 3/200
```

```
66/66 [==============================] - 181s 3s/step - loss: 0.2352 - accuracy:
0.9261 - precision: 0.9471 - recall: 0.9042 - val_loss: 0.2353 - val_accuracy:
0.9303 - val_precision: 0.9433 - val_recall: 0.9151 - lr: 0.0010
Epoch 4/200
66/66 [==============================] - 182s 3s/step - loss: 0.1700 - accuracy:
0.9464 - precision: 0.9622 - recall: 0.9331 - val_loss: 0.2181 - val_accuracy:
0.9374 - val_precision: 0.9494 - val_recall: 0.9260 - lr: 0.0010
Epoch 5/200
66/66 [==============================] - 183s 3s/step - loss: 0.1334 - accuracy:
0.9553 - precision: 0.9673 - recall: 0.9460 - val_loss: 0.2419 - val_accuracy:
0.9237 - val_precision: 0.9378 - val_recall: 0.9151 - lr: 0.0010
Epoch 6/200
66/66 [==============================] - 184s 3s/step - loss: 0.1072 - accuracy:
0.9640 - precision: 0.9723 - recall: 0.9574 - val_loss: 0.2516 - val_accuracy:
0.9341 - val_precision: 0.9394 - val_recall: 0.9265 - lr: 0.0010
Epoch 7/200
66/66 [==============================] - 184s 3s/step - loss: 0.1064 - accuracy:
0.9664 - precision: 0.9735 - recall: 0.9597 - val_loss: 0.2352 - val_accuracy:
0.9412 - val_precision: 0.9476 - val_recall: 0.9355 - lr: 0.0010
Epoch 8/200
66/66 [==============================] - 184s 3s/step - loss: 0.0754 - accuracy:
0.9768 - precision: 0.9803 - recall: 0.9717 - val_loss: 0.2854 - val_accuracy:
0.9237 - val_precision: 0.9304 - val_recall: 0.9189 - lr: 0.0010
Epoch 9/200
66/66 [==============================] - 184s 3s/step - loss: 0.0697 - accuracy:
0.9781 - precision: 0.9827 - recall: 0.9721 - val_loss: 0.2854 - val_accuracy:
0.9298 - val_precision: 0.9356 - val_recall: 0.9232 - lr: 0.0010
Epoch 10/200
66/66 [==============================] - 184s 3s/step - loss: 0.0700 - accuracy:
0.9763 - precision: 0.9792 - recall: 0.9721 - val_loss: 0.2590 - val_accuracy:
0.9350 - val_precision: 0.9421 - val_recall: 0.9331 - lr: 0.0010
Epoch 11/200
66/66 [==============================] - 184s 3s/step - loss: 0.0744 - accuracy:
0.9746 - precision: 0.9798 - recall: 0.9711 - val_loss: 0.3145 - val_accuracy:
0.9322 - val_precision: 0.9354 - val_recall: 0.9270 - lr: 0.0010
Epoch 12/200
66/66 [==============================] - 184s 3s/step - loss: 0.0681 - accuracy:
0.9774 - precision: 0.9816 - recall: 0.9728 - val_loss: 0.3680 - val_accuracy:
0.9199 - val_precision: 0.9250 - val_recall: 0.9180 - lr: 0.0010
Epoch 13/200
66/66 [==============================] - 187s 3s/step - loss: 0.0890 - accuracy:
0.9695 - precision: 0.9752 - recall: 0.9645 - val_loss: 0.2783 - val_accuracy:
0.9331 - val_precision: 0.9413 - val_recall: 0.9270 - lr: 0.0010
Epoch 14/200
66/66 [==============================] - ETA: 0s - loss: 0.0939 - accuracy:
0.9689 - precision: 0.9743 - recall: 0.9631
Epoch 14: ReduceLROnPlateau reducing learning rate to 0.0003000000142492354.
66/66 [==============================] - 187s 3s/step - loss: 0.0939 - accuracy:
```

```
0.9689 - precision: 0.9743 - recall: 0.9631 - val_loss: 0.2861 - val_accuracy:
0.9327 - val_precision: 0.9403 - val_recall: 0.9260 - lr: 0.0010
Epoch 15/200
66/66 [==============================] - 187s 3s/step - loss: 0.0395 - accuracy:
0.9868 - precision: 0.9905 - recall: 0.9843 - val_loss: 0.2848 - val_accuracy:
0.9374 - val_precision: 0.9420 - val_recall: 0.9327 - lr: 3.0000e-04
Epoch 16/200
66/66 [==============================] - 187s 3s/step - loss: 0.0306 - accuracy:
0.9899 - precision: 0.9920 - recall: 0.9877 - val_loss: 0.3013 - val_accuracy:
0.9388 - val_precision: 0.9431 - val_recall: 0.9360 - lr: 3.0000e-04
Epoch 17/200
66/66 [==============================] - 187s 3s/step - loss: 0.0312 - accuracy:
0.9892 - precision: 0.9907 - recall: 0.9877 - val_loss: 0.2934 - val_accuracy:
0.9426 - val_precision: 0.9451 - val_recall: 0.9388 - lr: 3.0000e-04
Epoch 18/200
66/66 [==============================] - 187s 3s/step - loss: 0.0266 - accuracy:
0.9917 - precision: 0.9932 - recall: 0.9897 - val_loss: 0.3166 - val_accuracy:
0.9360 - val_precision: 0.9438 - val_recall: 0.9322 - lr: 3.0000e-04
Epoch 19/200
66/66 [==============================] - 187s 3s/step - loss: 0.0217 - accuracy:
0.9925 - precision: 0.9933 - recall: 0.9910 - val_loss: 0.3314 - val_accuracy:
0.9336 - val_precision: 0.9371 - val_recall: 0.9327 - lr: 3.0000e-04
Epoch 20/200
66/66 [==============================] - 188s 3s/step - loss: 0.0225 - accuracy:
0.9925 - precision: 0.9931 - recall: 0.9915 - val_loss: 0.3090 - val_accuracy:
0.9388 - val_precision: 0.9433 - val_recall: 0.9379 - lr: 3.0000e-04
Epoch 21/200
66/66 [==============================] - 188s 3s/step - loss: 0.0213 - accuracy:
0.9938 - precision: 0.9954 - recall: 0.9928 - val_loss: 0.3094 - val_accuracy:
0.9407 - val_precision: 0.9438 - val_recall: 0.9393 - lr: 3.0000e-04
Epoch 22/200
66/66 [==============================] - 188s 3s/step - loss: 0.0214 - accuracy:
0.9926 - precision: 0.9944 - recall: 0.9915 - val_loss: 0.3094 - val_accuracy:
0.9388 - val_precision: 0.9437 - val_recall: 0.9374 - lr: 3.0000e-04
Epoch 23/200
66/66 [==============================] - 188s 3s/step - loss: 0.0185 - accuracy:
0.9943 - precision: 0.9956 - recall: 0.9929 - val_loss: 0.3215 - val_accuracy:
0.9398 - val_precision: 0.9414 - val_recall: 0.9365 - lr: 3.0000e-04
Epoch 24/200
66/66 [==============================] - ETA: 0s - loss: 0.0182 - accuracy:
0.9944 - precision: 0.9954 - recall: 0.9934
Epoch 24: ReduceLROnPlateau reducing learning rate to 9.000000427477062e-05.
66/66 [==============================] - 187s 3s/step - loss: 0.0182 - accuracy:
0.9944 - precision: 0.9954 - recall: 0.9934 - val_loss: 0.3249 - val_accuracy:
0.9379 - val_precision: 0.9408 - val_recall: 0.9346 - lr: 3.0000e-04
Epoch 25/200
66/66 [==============================] - 190s 3s/step - loss: 0.0164 - accuracy:
0.9948 - precision: 0.9951 - recall: 0.9942 - val_loss: 0.3196 - val_accuracy:
```

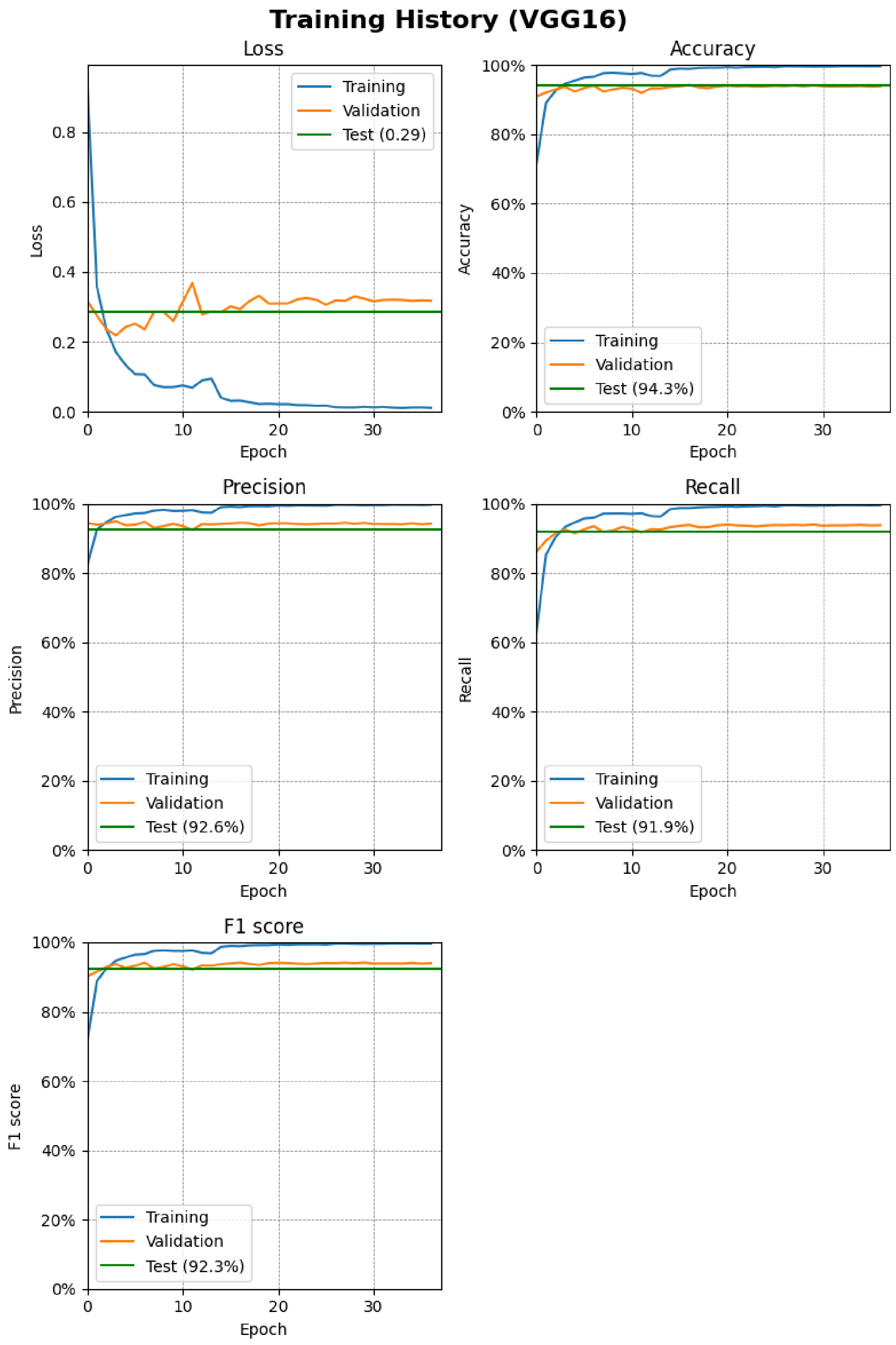
```
0.9388 - val_precision: 0.9418 - val_recall: 0.9369 - lr: 9.0000e-05
Epoch 26/200
66/66 [==============================] - 192s 3s/step - loss: 0.0170 - accuracy:
0.9937 - precision: 0.9948 - recall: 0.9926 - val_loss: 0.3056 - val_accuracy:
0.9403 - val_precision: 0.9433 - val_recall: 0.9384 - lr: 9.0000e-05
Epoch 27/200
66/66 [==============================] - 192s 3s/step - loss: 0.0125 - accuracy:
0.9967 - precision: 0.9973 - recall: 0.9957 - val_loss: 0.3183 - val_accuracy:
0.9393 - val_precision: 0.9428 - val_recall: 0.9379 - lr: 9.0000e-05
Epoch 28/200
66/66 [==============================] - 192s 3s/step - loss: 0.0119 - accuracy:
0.9963 - precision: 0.9973 - recall: 0.9957 - val_loss: 0.3171 - val_accuracy:
0.9412 - val_precision: 0.9451 - val_recall: 0.9388 - lr: 9.0000e-05
Epoch 29/200
66/66 [==============================] - 193s 3s/step - loss: 0.0118 - accuracy:
0.9962 - precision: 0.9969 - recall: 0.9949 - val_loss: 0.3294 - val_accuracy:
0.9388 - val_precision: 0.9424 - val_recall: 0.9379 - lr: 9.0000e-05
Epoch 30/200
66/66 [==============================] - 193s 3s/step - loss: 0.0138 - accuracy:
0.9955 - precision: 0.9962 - recall: 0.9947 - val_loss: 0.3229 - val_accuracy:
0.9417 - val_precision: 0.9447 - val_recall: 0.9398 - lr: 9.0000e-05
Epoch 31/200
66/66 [==============================] - ETA: 0s - loss: 0.0120 - accuracy:
0.9958 - precision: 0.9966 - recall: 0.9953
Epoch 31: ReduceLROnPlateau reducing learning rate to 2.700000040931627e-05.
66/66 [==============================] - 192s 3s/step - loss: 0.0120 - accuracy:
0.9958 - precision: 0.9966 - recall: 0.9953 - val_loss: 0.3156 - val_accuracy:
0.9388 - val_precision: 0.9418 - val_recall: 0.9365 - lr: 9.0000e-05
Epoch 32/200
66/66 [==============================] - 192s 3s/step - loss: 0.0134 - accuracy:
0.9960 - precision: 0.9967 - recall: 0.9954 - val_loss: 0.3195 - val_accuracy:
0.9384 - val_precision: 0.9414 - val_recall: 0.9374 - lr: 2.7000e-05
Epoch 33/200
66/66 [==============================] - 193s 3s/step - loss: 0.0112 - accuracy:
0.9969 - precision: 0.9977 - recall: 0.9963 - val_loss: 0.3203 - val_accuracy:
0.9388 - val_precision: 0.9414 - val_recall: 0.9374 - lr: 2.7000e-05
Epoch 34/200
66/66 [==============================] - 193s 3s/step - loss: 0.0103 - accuracy:
0.9968 - precision: 0.9974 - recall: 0.9962 - val_loss: 0.3195 - val_accuracy:
0.9388 - val_precision: 0.9406 - val_recall: 0.9379 - lr: 2.7000e-05
Epoch 35/200
66/66 [==============================] - 192s 3s/step - loss: 0.0116 - accuracy:
0.9968 - precision: 0.9975 - recall: 0.9961 - val_loss: 0.3169 - val_accuracy:
0.9398 - val_precision: 0.9438 - val_recall: 0.9388 - lr: 2.7000e-05
Epoch 36/200
66/66 [==============================] - 192s 3s/step - loss: 0.0119 - accuracy:
0.9966 - precision: 0.9969 - recall: 0.9957 - val_loss: 0.3183 - val_accuracy:
0.9379 - val_precision: 0.9405 - val_recall: 0.9374 - lr: 2.7000e-05
```

```
Epoch 37/200
66/66 [==============================] - ETA: 0s - loss: 0.0105 - accuracy:
0.9968 - precision: 0.9976 - recall: 0.9961Restoring model weights from the end
of the best epoch: 17.
66/66 [==============================] - 192s 3s/step - loss: 0.0105 - accuracy:
0.9968 - precision: 0.9976 - recall: 0.9961 - val_loss: 0.3173 - val_accuracy:
0.9388 - val_precision: 0.9428 - val_recall: 0.9379 - lr: 2.7000e-05
Epoch 37: early stopping
VGG16 Training time: 115.0 min 42.25 sec
```

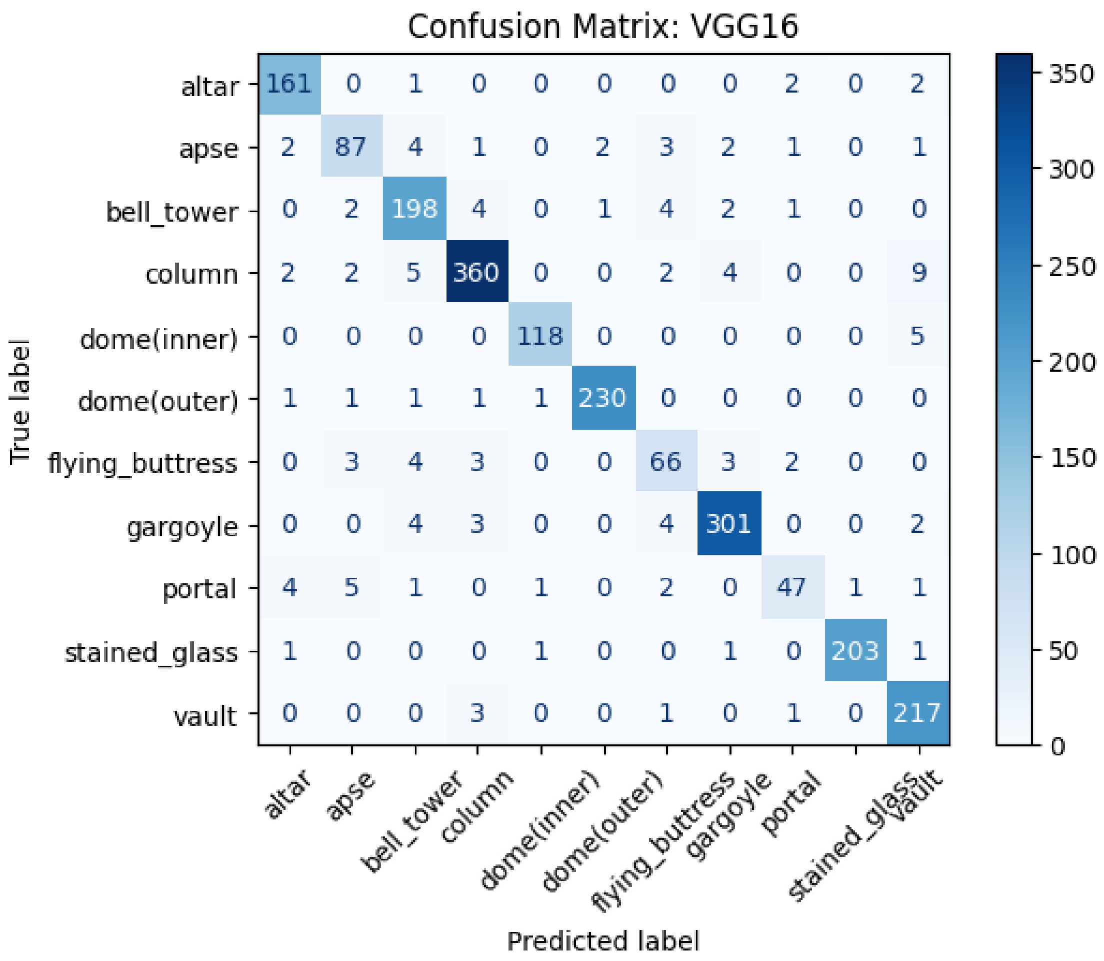
[ ]:
```
predictions_vgg16, predicted_labels_vgg16 = get_predictions(model_vgg16,␣
  ↪test_generator)
```

```
17/17 [==============================] - 38s 2s/step
```

[ ]:
```
training_plot(history_vgg16, predicted_labels_vgg16, predictions_vgg16,␣
  ↪test_generator, "Training History (VGG16)")
```


Training History (VGG16)
Loss
Accuracy
Precision
Recall
F1 score
Epoch
Training
Validation
Test (0.29)
Test (94.3%)
Test (92.6%)
Test (91.9%)
Test (92.3%)

```
[ ]: confusion_matrix_and_classification_report(predicted_labels_vgg16,␣
      ↪test_generator, 'VGG16')
```



| | altar | apse | bell_tower | column | dome(inner) | dome(outer) | flying_buttress | gargoyle | portal | stained_glass | vault |
|---|---|---|---|---|---|---|---|---|---|---|---|
| altar | 161 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 2 |
| apse | 2 | 87 | 4 | 1 | 0 | 2 | 3 | 2 | 1 | 0 | 1 |
| bell_tower | 0 | 2 | 198 | 4 | 0 | 1 | 4 | 2 | 1 | 0 | 0 |
| column | 2 | 2 | 5 | 360 | 0 | 0 | 2 | 4 | 0 | 0 | 9 |
| dome(inner) | 0 | 0 | 0 | 0 | 118 | 0 | 0 | 0 | 0 | 0 | 5 |
| dome(outer) | 1 | 1 | 1 | 1 | 1 | 230 | 0 | 0 | 0 | 0 | 0 |
| flying_buttress | 0 | 3 | 4 | 3 | 0 | 0 | 66 | 3 | 2 | 0 | 0 |
| gargoyle | 0 | 0 | 4 | 3 | 0 | 0 | 4 | 301 | 0 | 0 | 2 |
| portal | 4 | 5 | 1 | 0 | 1 | 0 | 2 | 0 | 47 | 1 | 1 |
| stained_glass | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 203 | 1 |
| vault | 0 | 0 | 0 | 3 | 0 | 0 | 1 | 0 | 1 | 0 | 217 |

```
                 precision    recall  f1-score   support

          altar       0.94      0.97      0.96       166
           apse       0.87      0.84      0.86       103
     bell_tower       0.91      0.93      0.92       212
         column       0.96      0.94      0.95       384
    dome(inner)       0.98      0.96      0.97       123
    dome(outer)       0.99      0.98      0.98       235
flying_buttress       0.80      0.81      0.81        81
       gargoyle       0.96      0.96      0.96       314
         portal       0.87      0.76      0.81        62
  stained_glass       1.00      0.98      0.99       207
```

```
       vault       0.91      0.98      0.94       222

    accuracy                           0.94      2109
   macro avg       0.93      0.92      0.92      2109
weighted avg       0.94      0.94      0.94      2109
```

10. Train the model with augmentation and keep monitoring validation accuracy
    **Note**: Choose carefully the number of epochs, steps per epoch, and validation steps based on your computer configuration
    >
    > We'll only do augmentation for the VGG16 pre-trained network as it clearly outperforms the other architectures we've tried.
    >

11. Visualize training and validation accuracy on the y-axis against each epoch on the x-axis to see if the model overfits after a certain epoch
    >
    > This was done several times earlier, showing accuracy, precision, recall, loss, and F1 score. In addition, we produced confusion matrices for each tested pre-trained model. In each case, the combination of early stopping and reducing learning rate on plateau seems to have eliminated overfitting.
    >

[ ]:
```
# Build a train/validate generator pair from the train folder using data
# augmentation for the train data set and also balanced weights for the
# classes

aug_train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    zoom_range=0.2,
    brightness_range=[0.5, 1.5],
    horizontal_flip=True,
    fill_mode='nearest'
)

aug_train_generator = aug_train_datagen.flow_from_dataframe(
    dataframe=df2_train,
    x_col='image',
    y_col='label',
    target_size=(224, 224),
    class_mode='categorical',
    batch_size=128,
    shuffle=True,
    seed=42
)

aug_valid_datagen = ImageDataGenerator(rescale=1./255)

aug_valid_generator = aug_valid_datagen.flow_from_dataframe(
    dataframe=df2_valid,
    x_col='image',
    y_col='label',
    target_size=(224, 224),
    class_mode='categorical',
    batch_size= 128,
    shuffle=False,
    seed=42
)
```

```
Found 8433 validated image filenames belonging to 11 classes.
Found 2109 validated image filenames belonging to 11 classes.
```

[ ]:
```
# Compute class weights based on df_train
class_counts = df_train['label'].value_counts().to_dict()
class_weights = {i: 1 / v for i, (k, v) in enumerate(class_counts.items())}
sum_weights = sum(class_weights.values())
class_weights = {k: v / sum_weights for k, v in class_weights.items()}
```

[ ]:
```
model_vgg16_aug = build_or_load_model(
    tensorflow.keras.applications.VGG16(include_top=False, weights='imagenet',␣
 ↪input_shape=(224, 224, 3)),
    len(df_train['label'].unique()),
    name='VGG16_aug'
)
```

```
Model: "VGG16_aug"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 flatten_7 (Flatten)         (None, 25088)             0

 dense_21 (Dense)            (None, 256)               6422784

 dropout_14 (Dropout)        (None, 256)               0

 dense_22 (Dense)            (None, 128)               32896

 dropout_15 (Dropout)        (None, 128)               0

 dense_23 (Dense)            (None, 11)                1419
```

```
=================================================================
Total params: 21171787 (80.76 MB)
Trainable params: 6457099 (24.63 MB)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
```

[ ]:
```
history_vgg16_aug = model_fit(
    model_vgg16_aug,
    aug_train_generator,
    aug_valid_generator,
    max_epochs,
    callbacks(),
    class_weight=class_weights
)
```

```
Epoch 1/200
66/66 [==============================] - 244s 4s/step - loss: 0.1264 - accuracy:
0.5289 - precision: 0.6673 - recall: 0.4031 - val_loss: 0.6337 - val_accuracy:
0.8094 - val_precision: 0.8896 - val_recall: 0.6880 - lr: 0.0010
Epoch 2/200
66/66 [==============================] - 242s 4s/step - loss: 0.0595 - accuracy:
0.7366 - precision: 0.8516 - recall: 0.6337 - val_loss: 0.4048 - val_accuracy:
0.8843 - val_precision: 0.9349 - val_recall: 0.8108 - lr: 0.0010
Epoch 3/200
66/66 [==============================] - 242s 4s/step - loss: 0.0486 - accuracy:
0.7783 - precision: 0.8665 - recall: 0.6975 - val_loss: 0.3372 - val_accuracy:
0.9066 - val_precision: 0.9422 - val_recall: 0.8582 - lr: 0.0010
Epoch 4/200
66/66 [==============================] - 241s 4s/step - loss: 0.0475 - accuracy:
0.7888 - precision: 0.8635 - recall: 0.7184 - val_loss: 0.3284 - val_accuracy:
0.9033 - val_precision: 0.9369 - val_recall: 0.8663 - lr: 0.0010
Epoch 5/200
66/66 [==============================] - 242s 4s/step - loss: 0.0458 - accuracy:
0.7981 - precision: 0.8744 - recall: 0.7306 - val_loss: 0.3661 - val_accuracy:
0.8971 - val_precision: 0.9319 - val_recall: 0.8497 - lr: 0.0010
Epoch 6/200
66/66 [==============================] - 242s 4s/step - loss: 0.0447 - accuracy:
0.8016 - precision: 0.8763 - recall: 0.7324 - val_loss: 0.3888 - val_accuracy:
0.8881 - val_precision: 0.9238 - val_recall: 0.8450 - lr: 0.0010
Epoch 7/200
66/66 [==============================] - 241s 4s/step - loss: 0.0418 - accuracy:
0.8128 - precision: 0.8828 - recall: 0.7550 - val_loss: 0.3220 - val_accuracy:
0.9075 - val_precision: 0.9361 - val_recall: 0.8819 - lr: 0.0010
Epoch 8/200
66/66 [==============================] - 241s 4s/step - loss: 0.0420 - accuracy:
0.8167 - precision: 0.8828 - recall: 0.7556 - val_loss: 0.2845 - val_accuracy:
0.9170 - val_precision: 0.9427 - val_recall: 0.8886 - lr: 0.0010
```

```
Epoch 9/200
66/66 [==============================] - 241s 4s/step - loss: 0.0386 - accuracy:
0.8268 - precision: 0.8860 - recall: 0.7753 - val_loss: 0.3107 - val_accuracy:
0.9090 - val_precision: 0.9374 - val_recall: 0.8805 - lr: 0.0010
Epoch 10/200
66/66 [==============================] - 242s 4s/step - loss: 0.0373 - accuracy:
0.8279 - precision: 0.8834 - recall: 0.7760 - val_loss: 0.2840 - val_accuracy:
0.9194 - val_precision: 0.9366 - val_recall: 0.8966 - lr: 0.0010
Epoch 11/200
66/66 [==============================] - 242s 4s/step - loss: 0.0387 - accuracy:
0.8278 - precision: 0.8867 - recall: 0.7767 - val_loss: 0.3102 - val_accuracy:
0.9071 - val_precision: 0.9365 - val_recall: 0.8876 - lr: 0.0010
Epoch 12/200
66/66 [==============================] - 242s 4s/step - loss: 0.0392 - accuracy:
0.8252 - precision: 0.8851 - recall: 0.7707 - val_loss: 0.2858 - val_accuracy:
0.9104 - val_precision: 0.9429 - val_recall: 0.8843 - lr: 0.0010
Epoch 13/200
66/66 [==============================] - 242s 4s/step - loss: 0.0378 - accuracy:
0.8286 - precision: 0.8838 - recall: 0.7791 - val_loss: 0.2650 - val_accuracy:
0.9189 - val_precision: 0.9427 - val_recall: 0.9047 - lr: 0.0010
Epoch 14/200
66/66 [==============================] - 242s 4s/step - loss: 0.0352 - accuracy:
0.8418 - precision: 0.8918 - recall: 0.8017 - val_loss: 0.2869 - val_accuracy:
0.9175 - val_precision: 0.9407 - val_recall: 0.8952 - lr: 0.0010
Epoch 15/200
66/66 [==============================] - 242s 4s/step - loss: 0.0342 - accuracy:
0.8438 - precision: 0.8912 - recall: 0.8014 - val_loss: 0.2779 - val_accuracy:
0.9161 - val_precision: 0.9400 - val_recall: 0.8990 - lr: 0.0010
Epoch 16/200
66/66 [==============================] - 242s 4s/step - loss: 0.0347 - accuracy:
0.8418 - precision: 0.8925 - recall: 0.7971 - val_loss: 0.2643 - val_accuracy:
0.9289 - val_precision: 0.9485 - val_recall: 0.8990 - lr: 0.0010
Epoch 17/200
66/66 [==============================] - 242s 4s/step - loss: 0.0332 - accuracy:
0.8462 - precision: 0.8987 - recall: 0.8035 - val_loss: 0.2541 - val_accuracy:
0.9227 - val_precision: 0.9479 - val_recall: 0.9056 - lr: 0.0010
Epoch 18/200
66/66 [==============================] - 241s 4s/step - loss: 0.0337 - accuracy:
0.8492 - precision: 0.8979 - recall: 0.8083 - val_loss: 0.3008 - val_accuracy:
0.9180 - val_precision: 0.9390 - val_recall: 0.8971 - lr: 0.0010
Epoch 19/200
66/66 [==============================] - 242s 4s/step - loss: 0.0332 - accuracy:
0.8509 - precision: 0.9006 - recall: 0.8086 - val_loss: 0.2602 - val_accuracy:
0.9265 - val_precision: 0.9411 - val_recall: 0.9170 - lr: 0.0010
Epoch 20/200
66/66 [==============================] - 241s 4s/step - loss: 0.0324 - accuracy:
0.8544 - precision: 0.8986 - recall: 0.8147 - val_loss: 0.2806 - val_accuracy:
0.9213 - val_precision: 0.9389 - val_recall: 0.9037 - lr: 0.0010
```

```
Epoch 21/200
66/66 [==============================] - 242s 4s/step - loss: 0.0305 - accuracy:
0.8571 - precision: 0.9021 - recall: 0.8224 - val_loss: 0.2620 - val_accuracy:
0.9222 - val_precision: 0.9382 - val_recall: 0.9071 - lr: 0.0010
Epoch 22/200
66/66 [==============================] - 242s 4s/step - loss: 0.0317 - accuracy:
0.8538 - precision: 0.9040 - recall: 0.8156 - val_loss: 0.2528 - val_accuracy:
0.9260 - val_precision: 0.9466 - val_recall: 0.9075 - lr: 0.0010
Epoch 23/200
66/66 [==============================] - ETA: 0s - loss: 0.0321 - accuracy:
0.8502 - precision: 0.8978 - recall: 0.8117
Epoch 23: ReduceLROnPlateau reducing learning rate to 0.0003000000142492354.
66/66 [==============================] - 242s 4s/step - loss: 0.0321 - accuracy:
0.8502 - precision: 0.8978 - recall: 0.8117 - val_loss: 0.2627 - val_accuracy:
0.9256 - val_precision: 0.9413 - val_recall: 0.9123 - lr: 0.0010
Epoch 24/200
66/66 [==============================] - 242s 4s/step - loss: 0.0266 - accuracy:
0.8748 - precision: 0.9130 - recall: 0.8367 - val_loss: 0.2365 - val_accuracy:
0.9308 - val_precision: 0.9452 - val_recall: 0.9151 - lr: 3.0000e-04
Epoch 25/200
66/66 [==============================] - 242s 4s/step - loss: 0.0265 - accuracy:
0.8780 - precision: 0.9171 - recall: 0.8434 - val_loss: 0.2511 - val_accuracy:
0.9251 - val_precision: 0.9418 - val_recall: 0.9132 - lr: 3.0000e-04
Epoch 26/200
66/66 [==============================] - 242s 4s/step - loss: 0.0262 - accuracy:
0.8820 - precision: 0.9196 - recall: 0.8468 - val_loss: 0.2426 - val_accuracy:
0.9279 - val_precision: 0.9447 - val_recall: 0.9161 - lr: 3.0000e-04
Epoch 27/200
66/66 [==============================] - 242s 4s/step - loss: 0.0258 - accuracy:
0.8801 - precision: 0.9199 - recall: 0.8468 - val_loss: 0.2367 - val_accuracy:
0.9303 - val_precision: 0.9456 - val_recall: 0.9147 - lr: 3.0000e-04
Epoch 28/200
66/66 [==============================] - 242s 4s/step - loss: 0.0248 - accuracy:
0.8809 - precision: 0.9184 - recall: 0.8530 - val_loss: 0.2294 - val_accuracy:
0.9327 - val_precision: 0.9461 - val_recall: 0.9237 - lr: 3.0000e-04
Epoch 29/200
66/66 [==============================] - 242s 4s/step - loss: 0.0249 - accuracy:
0.8850 - precision: 0.9197 - recall: 0.8534 - val_loss: 0.2306 - val_accuracy:
0.9322 - val_precision: 0.9438 - val_recall: 0.9165 - lr: 3.0000e-04
Epoch 30/200
66/66 [==============================] - 242s 4s/step - loss: 0.0249 - accuracy:
0.8831 - precision: 0.9197 - recall: 0.8530 - val_loss: 0.2338 - val_accuracy:
0.9350 - val_precision: 0.9455 - val_recall: 0.9213 - lr: 3.0000e-04
Epoch 31/200
66/66 [==============================] - 242s 4s/step - loss: 0.0242 - accuracy:
0.8868 - precision: 0.9212 - recall: 0.8572 - val_loss: 0.2381 - val_accuracy:
0.9312 - val_precision: 0.9431 - val_recall: 0.9199 - lr: 3.0000e-04
Epoch 32/200
```

```
66/66 [==============================] - 241s 4s/step - loss: 0.0245 - accuracy:
0.8822 - precision: 0.9169 - recall: 0.8541 - val_loss: 0.2352 - val_accuracy:
0.9331 - val_precision: 0.9468 - val_recall: 0.9203 - lr: 3.0000e-04
Epoch 33/200
66/66 [==============================] - 242s 4s/step - loss: 0.0234 - accuracy:
0.8915 - precision: 0.9235 - recall: 0.8632 - val_loss: 0.2442 - val_accuracy:
0.9270 - val_precision: 0.9433 - val_recall: 0.9147 - lr: 3.0000e-04
Epoch 34/200
66/66 [==============================] - 242s 4s/step - loss: 0.0239 - accuracy:
0.8876 - precision: 0.9211 - recall: 0.8583 - val_loss: 0.2376 - val_accuracy:
0.9308 - val_precision: 0.9441 - val_recall: 0.9203 - lr: 3.0000e-04
Epoch 35/200
66/66 [==============================] - 242s 4s/step - loss: 0.0229 - accuracy:
0.8917 - precision: 0.9258 - recall: 0.8645 - val_loss: 0.2303 - val_accuracy:
0.9350 - val_precision: 0.9473 - val_recall: 0.9203 - lr: 3.0000e-04
Epoch 36/200
66/66 [==============================] - 242s 4s/step - loss: 0.0238 - accuracy:
0.8888 - precision: 0.9218 - recall: 0.8578 - val_loss: 0.2395 - val_accuracy:
0.9331 - val_precision: 0.9483 - val_recall: 0.9227 - lr: 3.0000e-04
Epoch 37/200
66/66 [==============================] - ETA: 0s - loss: 0.0232 - accuracy:
0.8877 - precision: 0.9230 - recall: 0.8588
Epoch 37: ReduceLROnPlateau reducing learning rate to 9.000000427477062e-05.
66/66 [==============================] - 242s 4s/step - loss: 0.0232 - accuracy:
0.8877 - precision: 0.9230 - recall: 0.8588 - val_loss: 0.2261 - val_accuracy:
0.9341 - val_precision: 0.9454 - val_recall: 0.9279 - lr: 3.0000e-04
Epoch 38/200
66/66 [==============================] - 242s 4s/step - loss: 0.0232 - accuracy:
0.8908 - precision: 0.9213 - recall: 0.8658 - val_loss: 0.2236 - val_accuracy:
0.9369 - val_precision: 0.9476 - val_recall: 0.9260 - lr: 9.0000e-05
Epoch 39/200
66/66 [==============================] - 242s 4s/step - loss: 0.0216 - accuracy:
0.8992 - precision: 0.9271 - recall: 0.8719 - val_loss: 0.2276 - val_accuracy:
0.9350 - val_precision: 0.9470 - val_recall: 0.9227 - lr: 9.0000e-05
Epoch 40/200
66/66 [==============================] - 242s 4s/step - loss: 0.0219 - accuracy:
0.8984 - precision: 0.9294 - recall: 0.8684 - val_loss: 0.2272 - val_accuracy:
0.9346 - val_precision: 0.9453 - val_recall: 0.9256 - lr: 9.0000e-05
Epoch 41/200
66/66 [==============================] - 242s 4s/step - loss: 0.0216 - accuracy:
0.8984 - precision: 0.9269 - recall: 0.8687 - val_loss: 0.2215 - val_accuracy:
0.9374 - val_precision: 0.9485 - val_recall: 0.9256 - lr: 9.0000e-05
Epoch 42/200
66/66 [==============================] - 242s 4s/step - loss: 0.0206 - accuracy:
0.9018 - precision: 0.9315 - recall: 0.8777 - val_loss: 0.2213 - val_accuracy:
0.9360 - val_precision: 0.9486 - val_recall: 0.9275 - lr: 9.0000e-05
Epoch 43/200
66/66 [==============================] - 242s 4s/step - loss: 0.0225 - accuracy:
```

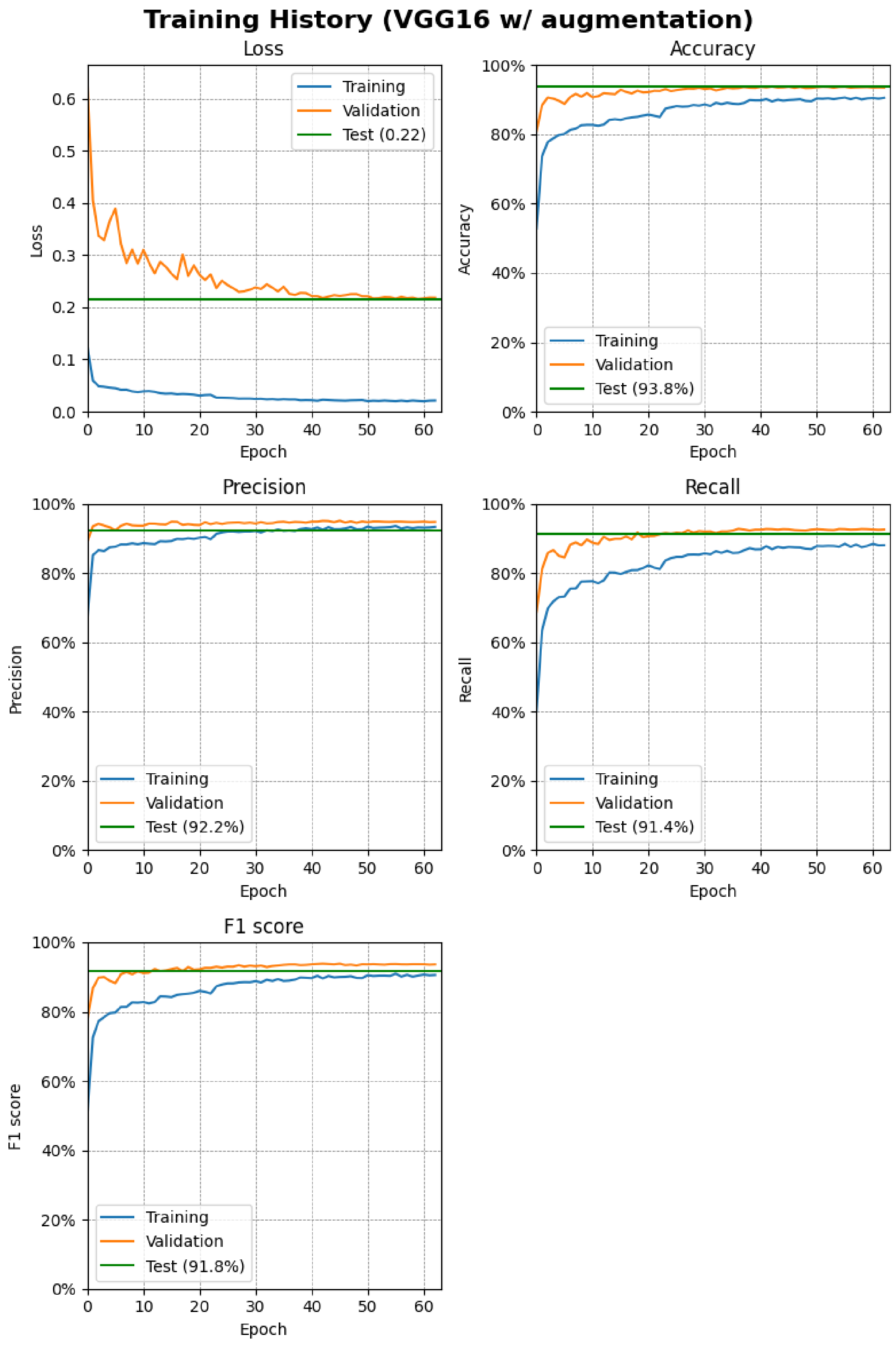
```
0.8951 - precision: 0.9255 - recall: 0.8686 - val_loss: 0.2175 - val_accuracy:
0.9384 - val_precision: 0.9509 - val_recall: 0.9270 - lr: 9.0000e-05
Epoch 44/200
66/66 [==============================] - 242s 4s/step - loss: 0.0220 - accuracy:
0.9002 - precision: 0.9326 - recall: 0.8758 - val_loss: 0.2203 - val_accuracy:
0.9355 - val_precision: 0.9503 - val_recall: 0.9256 - lr: 9.0000e-05
Epoch 45/200
66/66 [==============================] - 242s 4s/step - loss: 0.0214 - accuracy:
0.8975 - precision: 0.9268 - recall: 0.8728 - val_loss: 0.2233 - val_accuracy:
0.9360 - val_precision: 0.9467 - val_recall: 0.9270 - lr: 9.0000e-05
Epoch 46/200
66/66 [==============================] - 242s 4s/step - loss: 0.0210 - accuracy:
0.8992 - precision: 0.9268 - recall: 0.8755 - val_loss: 0.2218 - val_accuracy:
0.9379 - val_precision: 0.9513 - val_recall: 0.9265 - lr: 9.0000e-05
Epoch 47/200
66/66 [==============================] - 242s 4s/step - loss: 0.0207 - accuracy:
0.9002 - precision: 0.9289 - recall: 0.8744 - val_loss: 0.2232 - val_accuracy:
0.9346 - val_precision: 0.9457 - val_recall: 0.9241 - lr: 9.0000e-05
Epoch 48/200
66/66 [==============================] - 242s 4s/step - loss: 0.0214 - accuracy:
0.9016 - precision: 0.9330 - recall: 0.8738 - val_loss: 0.2252 - val_accuracy:
0.9374 - val_precision: 0.9493 - val_recall: 0.9232 - lr: 9.0000e-05
Epoch 49/200
66/66 [==============================] - 242s 4s/step - loss: 0.0216 - accuracy:
0.8967 - precision: 0.9264 - recall: 0.8703 - val_loss: 0.2253 - val_accuracy:
0.9341 - val_precision: 0.9447 - val_recall: 0.9227 - lr: 9.0000e-05
Epoch 50/200
66/66 [==============================] - ETA: 0s - loss: 0.0219 - accuracy:
0.8958 - precision: 0.9268 - recall: 0.8692
Epoch 50: ReduceLROnPlateau reducing learning rate to 2.700000040931627e-05.
66/66 [==============================] - 243s 4s/step - loss: 0.0219 - accuracy:
0.8958 - precision: 0.9268 - recall: 0.8692 - val_loss: 0.2213 - val_accuracy:
0.9346 - val_precision: 0.9490 - val_recall: 0.9256 - lr: 9.0000e-05
Epoch 51/200
66/66 [==============================] - 242s 4s/step - loss: 0.0199 - accuracy:
0.9037 - precision: 0.9340 - recall: 0.8783 - val_loss: 0.2211 - val_accuracy:
0.9365 - val_precision: 0.9467 - val_recall: 0.9270 - lr: 2.7000e-05
Epoch 52/200
66/66 [==============================] - 242s 4s/step - loss: 0.0206 - accuracy:
0.9030 - precision: 0.9300 - recall: 0.8779 - val_loss: 0.2167 - val_accuracy:
0.9379 - val_precision: 0.9490 - val_recall: 0.9260 - lr: 2.7000e-05
Epoch 53/200
66/66 [==============================] - 241s 4s/step - loss: 0.0202 - accuracy:
0.9043 - precision: 0.9311 - recall: 0.8787 - val_loss: 0.2172 - val_accuracy:
0.9374 - val_precision: 0.9489 - val_recall: 0.9246 - lr: 2.7000e-05
Epoch 54/200
66/66 [==============================] - 241s 4s/step - loss: 0.0210 - accuracy:
0.9018 - precision: 0.9317 - recall: 0.8783 - val_loss: 0.2193 - val_accuracy:
```

```
0.9350 - val_precision: 0.9480 - val_recall: 0.9241 - lr: 2.7000e-05
Epoch 55/200
66/66 [==============================] - 242s 4s/step - loss: 0.0203 - accuracy:
0.9045 - precision: 0.9327 - recall: 0.8762 - val_loss: 0.2189 - val_accuracy:
0.9374 - val_precision: 0.9477 - val_recall: 0.9275 - lr: 2.7000e-05
Epoch 56/200
66/66 [==============================] - 242s 4s/step - loss: 0.0198 - accuracy:
0.9066 - precision: 0.9358 - recall: 0.8850 - val_loss: 0.2163 - val_accuracy:
0.9379 - val_precision: 0.9486 - val_recall: 0.9270 - lr: 2.7000e-05
Epoch 57/200
66/66 [==============================] - ETA: 0s - loss: 0.0208 - accuracy:
0.9017 - precision: 0.9285 - recall: 0.8760
Epoch 57: ReduceLROnPlateau reducing learning rate to 8.100000013655517e-06.
66/66 [==============================] - 242s 4s/step - loss: 0.0208 - accuracy:
0.9017 - precision: 0.9285 - recall: 0.8760 - val_loss: 0.2199 - val_accuracy:
0.9350 - val_precision: 0.9485 - val_recall: 0.9256 - lr: 2.7000e-05
Epoch 58/200
66/66 [==============================] - 241s 4s/step - loss: 0.0197 - accuracy:
0.9060 - precision: 0.9322 - recall: 0.8822 - val_loss: 0.2172 - val_accuracy:
0.9360 - val_precision: 0.9476 - val_recall: 0.9260 - lr: 8.1000e-06
Epoch 59/200
66/66 [==============================] - 241s 4s/step - loss: 0.0210 - accuracy:
0.9009 - precision: 0.9296 - recall: 0.8749 - val_loss: 0.2182 - val_accuracy:
0.9365 - val_precision: 0.9472 - val_recall: 0.9275 - lr: 8.1000e-06
Epoch 60/200
66/66 [==============================] - 242s 4s/step - loss: 0.0202 - accuracy:
0.9042 - precision: 0.9317 - recall: 0.8786 - val_loss: 0.2157 - val_accuracy:
0.9369 - val_precision: 0.9476 - val_recall: 0.9270 - lr: 8.1000e-06
Epoch 61/200
66/66 [==============================] - 242s 4s/step - loss: 0.0196 - accuracy:
0.9053 - precision: 0.9311 - recall: 0.8846 - val_loss: 0.2169 - val_accuracy:
0.9355 - val_precision: 0.9485 - val_recall: 0.9260 - lr: 8.1000e-06
Epoch 62/200
66/66 [==============================] - 242s 4s/step - loss: 0.0207 - accuracy:
0.9032 - precision: 0.9317 - recall: 0.8800 - val_loss: 0.2183 - val_accuracy:
0.9360 - val_precision: 0.9471 - val_recall: 0.9251 - lr: 8.1000e-06
Epoch 63/200
66/66 [==============================] - ETA: 0s - loss: 0.0210 - accuracy:
0.9057 - precision: 0.9332 - recall: 0.8804Restoring model weights from the end
of the best epoch: 43.
66/66 [==============================] - 242s 4s/step - loss: 0.0210 - accuracy:
0.9057 - precision: 0.9332 - recall: 0.8804 - val_loss: 0.2181 - val_accuracy:
0.9355 - val_precision: 0.9476 - val_recall: 0.9260 - lr: 8.1000e-06
Epoch 63: early stopping
VGG16_aug Training time: 253.0 min 57.49 sec
```

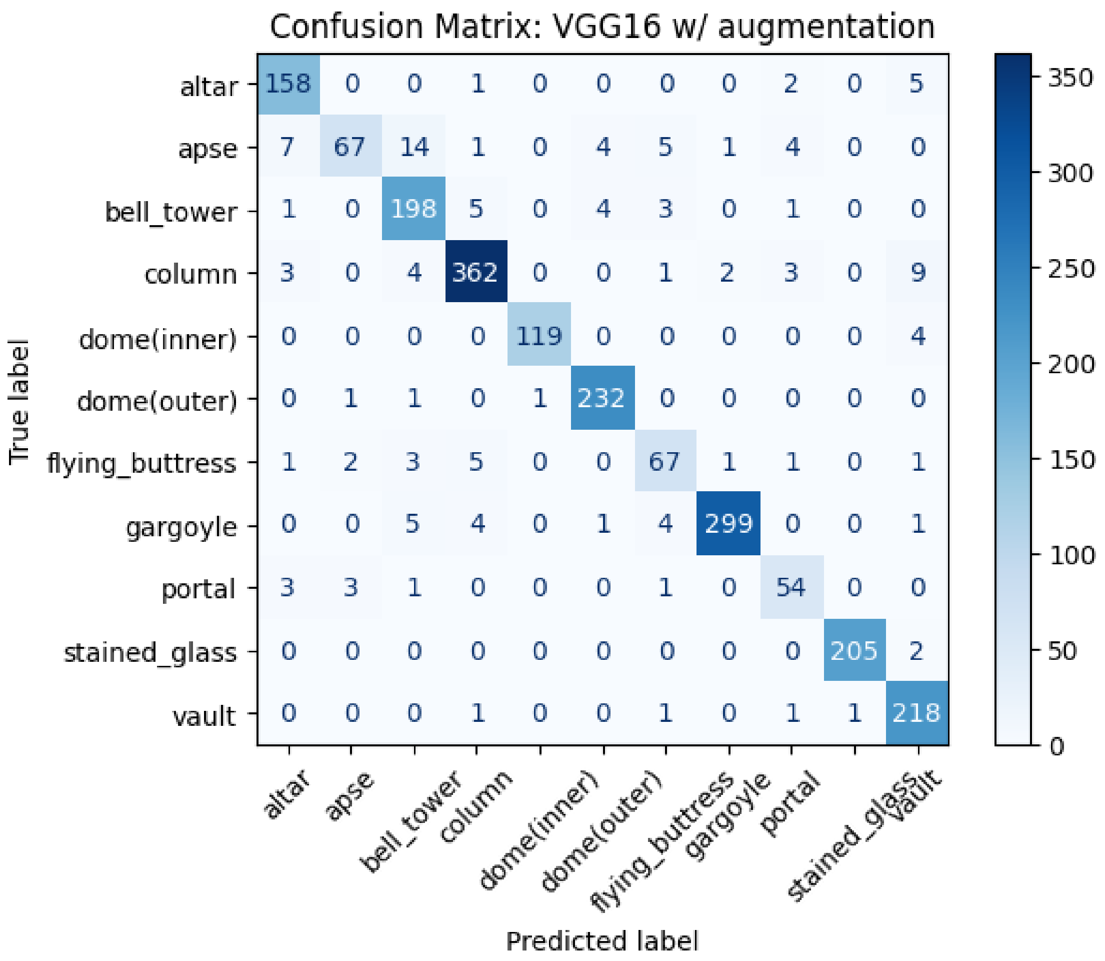
```
[ ]: predictions_vgg16_aug, predicted_labels_vgg16_aug =␣
      ↪get_predictions(model_vgg16_aug, test_generator)
```

```
17/17 [==============================] - 38s 2s/step
```

```
[ ]: training_plot(history_vgg16_aug, predicted_labels_vgg16_aug,␣
      ↪predictions_vgg16_aug, test_generator, "Training History (VGG16 w/␣
      ↪augmentation)")
```


Training History (VGG16 w/ augmentation)
Loss
Training
Validation
Test (0.22)
Accuracy
Test (93.8%)
Precision
Test (92.2%)
Recall
Test (91.4%)
F1 score
Test (91.8%)
Epoch

```
[ ]: confusion_matrix_and_classification_report(predicted_labels_vgg16_aug,␣
     ↪test_generator, 'VGG16 w/ augmentation')
```



| | altar | apse | bell_tower | column | dome(inner) | dome(outer) | flying_buttress | gargoyle | portal | stained_glass | vault |
|---|---|---|---|---|---|---|---|---|---|---|---|
| altar | 158 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 2 | 0 | 5 |
| apse | 7 | 67 | 14 | 1 | 0 | 4 | 5 | 1 | 4 | 0 | 0 |
| bell_tower | 1 | 0 | 198 | 5 | 0 | 4 | 3 | 0 | 1 | 0 | 0 |
| column | 3 | 0 | 4 | 362 | 0 | 0 | 1 | 2 | 3 | 0 | 9 |
| dome(inner) | 0 | 0 | 0 | 0 | 119 | 0 | 0 | 0 | 0 | 0 | 4 |
| dome(outer) | 0 | 1 | 1 | 0 | 1 | 232 | 0 | 0 | 0 | 0 | 0 |
| flying_buttress | 1 | 2 | 3 | 5 | 0 | 0 | 67 | 1 | 1 | 0 | 1 |
| gargoyle | 0 | 0 | 5 | 4 | 0 | 1 | 4 | 299 | 0 | 0 | 1 |
| portal | 3 | 3 | 1 | 0 | 0 | 0 | 1 | 0 | 54 | 0 | 0 |
| stained_glass | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 205 | 2 |
| vault | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 218 |

```
                 precision    recall  f1-score   support

          altar       0.91      0.95      0.93       166
           apse       0.92      0.65      0.76       103
     bell_tower       0.88      0.93      0.90       212
         column       0.96      0.94      0.95       384
    dome(inner)       0.99      0.97      0.98       123
    dome(outer)       0.96      0.99      0.97       235
flying_buttress       0.82      0.83      0.82        81
       gargoyle       0.99      0.95      0.97       314
         portal       0.82      0.87      0.84        62
  stained_glass       1.00      0.99      0.99       207
```

```
       vault       0.91      0.98      0.94       222

    accuracy                           0.94      2109
   macro avg       0.92      0.91      0.92      2109
weighted avg       0.94      0.94      0.94      2109
```