Leaf Classification Using CNN- Comparison Models

In [ ]:
```
import tensorflow as tf
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [ ]:
```
BATCH_SIZE = 32
IMAGE_SIZE = 256
CHANNELS=3
EPOCHS=30
```

Load dataset from drive

In [ ]:
```
valid_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    "/content/drive/MyDrive/Data/course_project/val",
    seed=123,
    shuffle=True,
    image_size=(IMAGE_SIZE,IMAGE_SIZE),
    batch_size=BATCH_SIZE
)
```

```
Found 310 files belonging to 10 classes.
```

In [ ]:
```
train_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    "/content/drive/MyDrive/Data/course_project/train",
    seed=123,
    shuffle=True,
    image_size=(IMAGE_SIZE,IMAGE_SIZE),
    batch_size=BATCH_SIZE
)
```

```
Found 2523 files belonging to 10 classes.
```

In [ ]:
```
test_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    "/content/drive/MyDrive/Data/course_project/test",
    seed=123,
    shuffle=True,
    image_size=(IMAGE_SIZE,IMAGE_SIZE),
    batch_size=BATCH_SIZE
)
```

```
Found 325 files belonging to 10 classes.
```

In [ ]:
```
class_names = train_dataset.class_names
class_names
```

Out[7]:
```
['Anacardiaceae',
 'Dipterocarpaceae',
 'Euphorbiaceae',
 'Fabaceae',
 'Fagaceae',
 'Magnoliaceae',
 'Moraceae',
 'Proteaceae',
 'Rosaceae',
 'Ulmaceae']
```

In [ ]:
```
for image_batch, labels_batch in train_dataset.take(1):
    print(image_batch.shape)
    print(labels_batch.numpy())
```

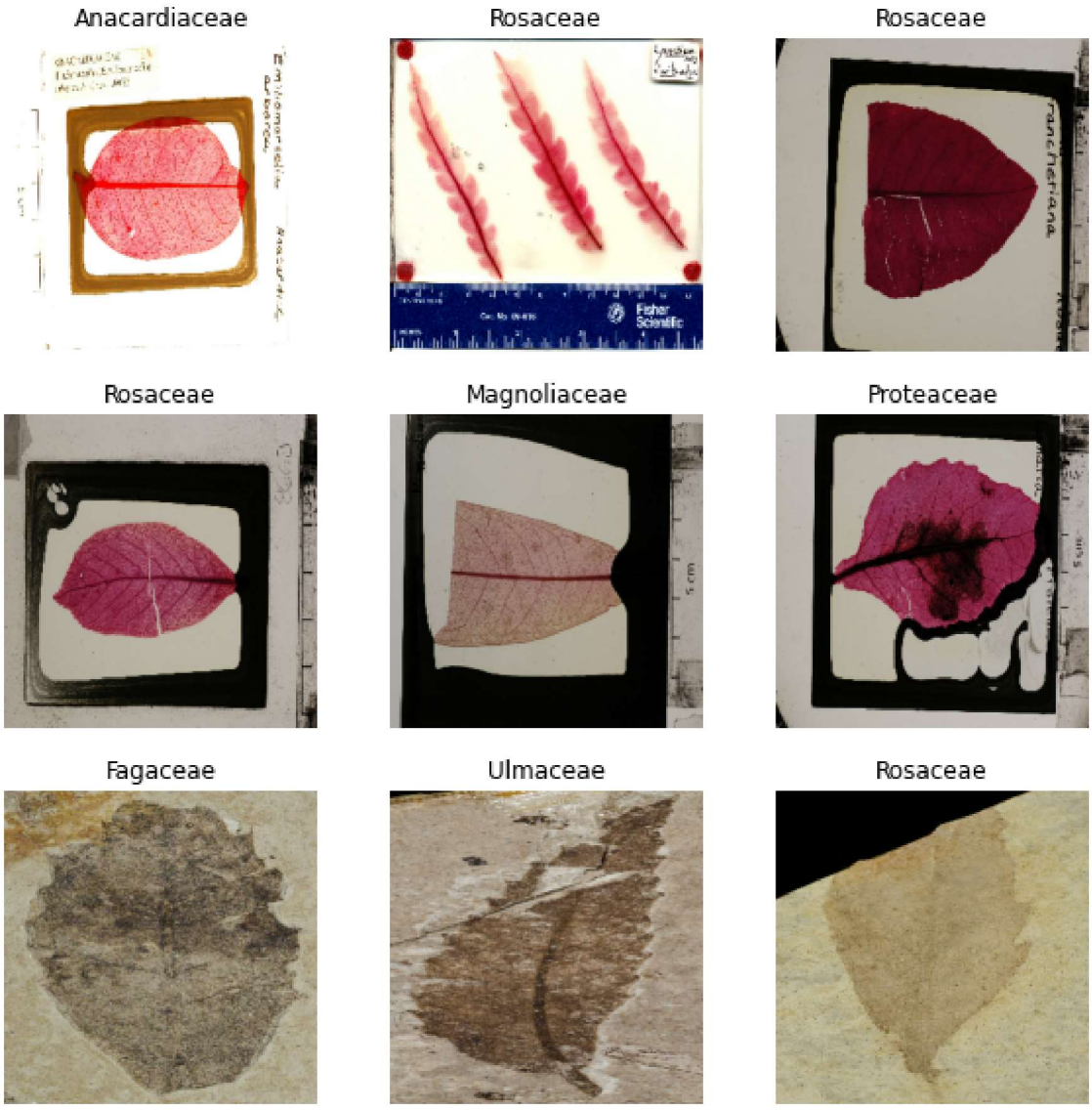
```
(32, 256, 256, 3)
[0 8 0 9 9 2 8 7 9 2 8 1 4 8 8 6 9 3 9 1 3 8 4 8 3 2 9 8 3 8 2 9]
```

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import os
import keras
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Input, Conv2DTransp
from tensorflow.keras.layers import  Dropout, Activation
from tensorflow.keras.optimizers import Adam, SGD
from keras.layers.advanced_activations import LeakyReLU
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, Ea
from tensorflow.keras.utils import plot_model
import tensorflow as tf
```

In [ ]:

```
plt.figure(figsize=(10, 10))
for image_batch, labels_batch in train_dataset.take(2):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(image_batch[i].numpy().astype("uint8"))
        plt.title(class_names[labels_batch[i]])
        plt.axis("off")
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: MatplotlibD
eprecationWarning: Adding an axes using the same arguments as a previous ax
es currently reuses the earlier instance.  In a future version, a new insta
nce will always be created and returned.  Meanwhile, this warning can be su
ppressed, and the future behavior ensured, by passing a unique label to eac
h axes instance.
  after removing the cwd from sys.path.
```



In [ ]:
```
#prefetch elements from the input dataset ahead of the time they are requeste
train_dataset = train_dataset.cache().shuffle(10).prefetch(buffer_size=tf.dat
valid_dataset = valid_dataset.cache().shuffle(10).prefetch(buffer_size=tf.dat
```

In [ ]:
```
resize_and_rescale = tf.keras.Sequential([
  layers.experimental.preprocessing.Rescaling(1./255),
  layers.experimental.preprocessing.Resizing(IMAGE_SIZE, IMAGE_SIZE)

])
```

In [ ]:
```
#test the effect of data crop
data_crop = tf.keras.Sequential([
    layers.experimental.preprocessing.CenterCrop(130, 130),
    layers.experimental.preprocessing.Resizing(227, 227)
])
```

In [ ]:
```
#test the effect of flipping and rotation
data_augmentation = tf.keras.Sequential([
  layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
  layers.experimental.preprocessing.RandomRotation(0.2),
])
```

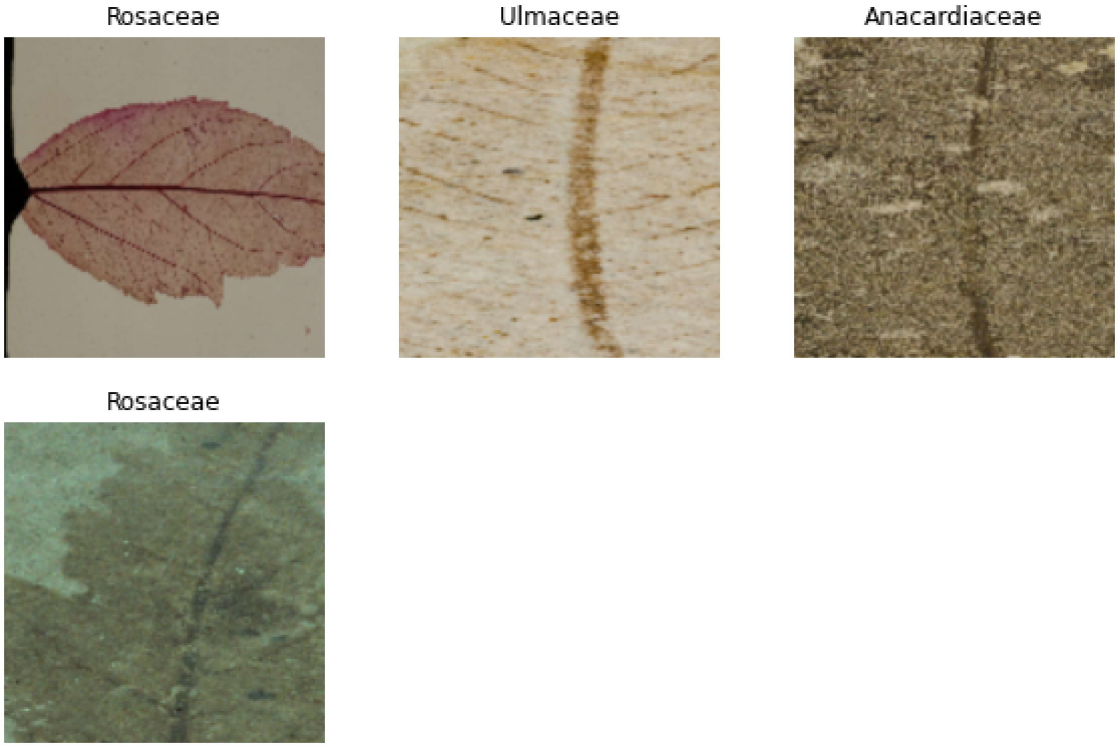
In [ ]:
```

```

In [ ]:
```
train_ds = train_dataset.map(
    lambda x, y: (data_crop(x), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)
```

In [ ]:
```
val_ds = valid_dataset.map(
    lambda x, y: (data_crop(x), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)
```

In [ ]:
```
train_ds = train_ds.map(
    lambda x, y: (data_augmentation(x, training=True), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)
```

In [ ]:
```
#After applying random crop
plt.figure(figsize=(10, 10))
for image_batch, labels_batch in train_ds.take(1):
    for i in range(4):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(image_batch[i].numpy().astype("uint8"))
        plt.title(class_names[labels_batch[i]])
        plt.axis("off")
```



In [ ]:
```
#convert to grayscale for LeNet
train_ds_new = train_dataset.map(
    lambda x, y: (tf.image.rgb_to_grayscale(x), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)
```

In [ ]:
```
#resize for LeNet
train_ds_new = train_ds_new.map(
    lambda x, y: (tf.image.resize(x,[256,256]), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)
```

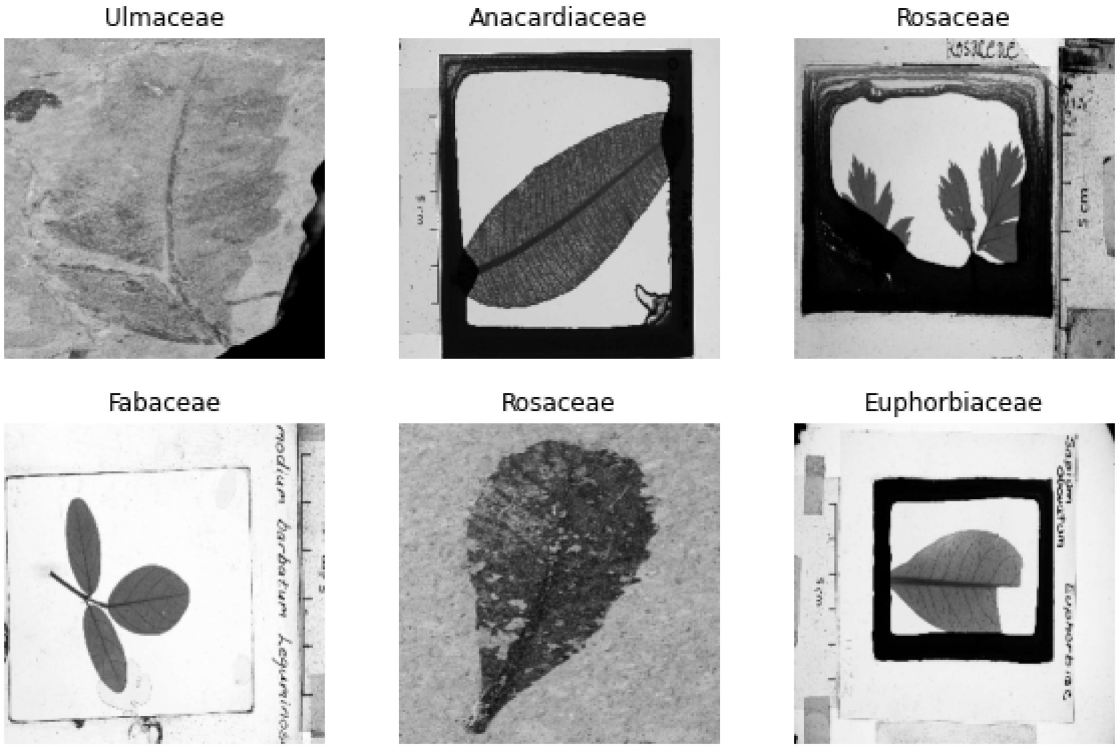
In [ ]:
```
valid_ds_new = valid_dataset.map(
    lambda x, y: (tf.image.rgb_to_grayscale(x), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)
```

In [ ]:
```
valid_ds_new = valid_ds_new.map(
    lambda x, y: (tf.image.resize(x,[256,256]), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)
```

In [ ]:
```
train_ds_new = train_ds_new.cache().shuffle(10).prefetch(buffer_size=tf.data.
valid_ds_new = valid_ds_new.cache().shuffle(10).prefetch(buffer_size=tf.data.
```

In [ ]:
```
plt.figure(figsize=(10, 10))
for image_batch, labels_batch in train_ds_new.take(2):
    for i in range(6):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(tf.squeeze(image_batch[i].numpy().astype("uint8")), cmap =
        plt.title(class_names[labels_batch[i]])
        plt.axis("off")
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: MatplotlibD
eprecationWarning: Adding an axes using the same arguments as a previous ax
es currently reuses the earlier instance.  In a future version, a new insta
nce will always be created and returned.  Meanwhile, this warning can be su
ppressed, and the future behavior ensured, by passing a unique label to eac
h axes instance.
  after removing the cwd from sys.path.
```



Test a deep convolutional neural network

In [ ]:

```
input_shape = (32, 227, 227, 1)
n_classes = 3

model = models.Sequential([
    resize_and_rescale,
    layers.Conv2D(32, kernel_size = (3,3), activation='relu', input_shape=inp
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax'),
])

model.build(input_shape=input_shape)
```

In [ ]:
```
model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_2 (Sequential)   (None, 256, 256, None)    0

 conv2d_32 (Conv2D)          (32, 254, 254, 32)        320

 max_pooling2d_28 (MaxPoolin  (32, 127, 127, 32)       0
 g2D)

 conv2d_33 (Conv2D)          (32, 125, 125, 64)        18496

 max_pooling2d_29 (MaxPoolin  (32, 62, 62, 64)         0
 g2D)

 conv2d_34 (Conv2D)          (32, 60, 60, 64)          36928

 max_pooling2d_30 (MaxPoolin  (32, 30, 30, 64)         0
 g2D)

 conv2d_35 (Conv2D)          (32, 28, 28, 64)          36928

 max_pooling2d_31 (MaxPoolin  (32, 14, 14, 64)         0
 g2D)

 conv2d_36 (Conv2D)          (32, 12, 12, 64)          36928

 max_pooling2d_32 (MaxPoolin  (32, 6, 6, 64)           0
 g2D)

 conv2d_37 (Conv2D)          (32, 4, 4, 64)            36928

 max_pooling2d_33 (MaxPoolin  (32, 2, 2, 64)           0
 g2D)

 flatten_7 (Flatten)         (32, 256)                 0

 dense_20 (Dense)            (32, 64)                  16448

 dense_21 (Dense)            (32, 10)                  650

=================================================================
Total params: 183,626
Trainable params: 183,626
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:

```
model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)
```

In [ ]:

```
history = model.fit(
    train_ds_new,
    batch_size= 32,
    validation_data=valid_ds_new,
    verbose=1,
    epochs=30,
)
```

```
Epoch 1/30
79/79 [==============================] - 220s 3s/step - loss: 2.0644 - ac
curacy: 0.2564 - val_loss: 1.9918 - val_accuracy: 0.2806
Epoch 2/30
79/79 [==============================] - 215s 3s/step - loss: 1.8933 - ac
curacy: 0.3040 - val_loss: 1.7187 - val_accuracy: 0.3548
Epoch 3/30
79/79 [==============================] - 213s 3s/step - loss: 1.7440 - ac
curacy: 0.3555 - val_loss: 1.6914 - val_accuracy: 0.3871
Epoch 4/30
79/79 [==============================] - 212s 3s/step - loss: 1.6834 - ac
curacy: 0.3690 - val_loss: 1.6170 - val_accuracy: 0.4226
Epoch 5/30
79/79 [==============================] - 212s 3s/step - loss: 1.6182 - ac
curacy: 0.3916 - val_loss: 1.5596 - val_accuracy: 0.4097
Epoch 6/30
79/79 [==============================] - 213s 3s/step - loss: 1.5392 - ac
curacy: 0.4285 - val_loss: 1.4936 - val_accuracy: 0.4323
Epoch 7/30
79/79 [==============================] - 212s 3s/step - loss: 1.4688 - ac
curacy: 0.4625 - val_loss: 1.5121 - val_accuracy: 0.4290
Epoch 8/30
79/79 [==============================] - 216s 3s/step - loss: 1.4265 - ac
curacy: 0.4709 - val_loss: 1.4497 - val_accuracy: 0.4742
Epoch 9/30
79/79 [==============================] - 213s 3s/step - loss: 1.3431 - ac
curacy: 0.5014 - val_loss: 1.4726 - val_accuracy: 0.4742
Epoch 10/30
79/79 [==============================] - 213s 3s/step - loss: 1.2639 - ac
curacy: 0.5466 - val_loss: 1.4463 - val_accuracy: 0.4581
Epoch 11/30
79/79 [==============================] - 212s 3s/step - loss: 1.1910 - ac
curacy: 0.5541 - val_loss: 1.5318 - val_accuracy: 0.4613
Epoch 12/30
79/79 [==============================] - 213s 3s/step - loss: 1.1027 - ac
curacy: 0.5890 - val_loss: 1.5839 - val_accuracy: 0.4742
Epoch 13/30
79/79 [==============================] - 212s 3s/step - loss: 1.0187 - ac
curacy: 0.6203 - val_loss: 1.3955 - val_accuracy: 0.5258
Epoch 14/30
79/79 [==============================] - 212s 3s/step - loss: 0.9210 - ac
curacy: 0.6544 - val_loss: 1.4905 - val_accuracy: 0.5581
Epoch 15/30
79/79 [==============================] - 211s 3s/step - loss: 0.8151 - ac
curacy: 0.7027 - val_loss: 1.6348 - val_accuracy: 0.5581
Epoch 16/30
79/79 [==============================] - 215s 3s/step - loss: 0.7946 - ac
```

```
curacy: 0.7071 - val_loss: 1.4903 - val_accuracy: 0.5419
Epoch 17/30
79/79 [==============================] - 212s 3s/step - loss: 0.6649 - ac
curacy: 0.7527 - val_loss: 1.7898 - val_accuracy: 0.5581
Epoch 18/30
79/79 [==============================] - 213s 3s/step - loss: 0.6329 - ac
curacy: 0.7693 - val_loss: 1.7140 - val_accuracy: 0.5323
Epoch 19/30
79/79 [==============================] - 213s 3s/step - loss: 0.5429 - ac
curacy: 0.7955 - val_loss: 2.1236 - val_accuracy: 0.5355
Epoch 20/30
79/79 [==============================] - 211s 3s/step - loss: 0.5294 - ac
curacy: 0.8050 - val_loss: 2.3097 - val_accuracy: 0.5452
Epoch 21/30
79/79 [==============================] - 211s 3s/step - loss: 0.4385 - ac
curacy: 0.8407 - val_loss: 2.2570 - val_accuracy: 0.5677
Epoch 22/30
79/79 [==============================] - 212s 3s/step - loss: 0.4193 - ac
curacy: 0.8466 - val_loss: 2.5835 - val_accuracy: 0.5194
Epoch 23/30
79/79 [==============================] - 213s 3s/step - loss: 0.3738 - ac
curacy: 0.8736 - val_loss: 2.4660 - val_accuracy: 0.5516
Epoch 24/30
79/79 [==============================] - 211s 3s/step - loss: 0.3204 - ac
curacy: 0.8783 - val_loss: 2.4046 - val_accuracy: 0.5645
Epoch 25/30
79/79 [==============================] - 212s 3s/step - loss: 0.3094 - ac
curacy: 0.8981 - val_loss: 2.6867 - val_accuracy: 0.5355
Epoch 26/30
79/79 [==============================] - 211s 3s/step - loss: 0.2376 - ac
curacy: 0.9156 - val_loss: 2.9552 - val_accuracy: 0.5258
Epoch 27/30
79/79 [==============================] - 210s 3s/step - loss: 0.2150 - ac
curacy: 0.9287 - val_loss: 3.2277 - val_accuracy: 0.5194
Epoch 28/30
79/79 [==============================] - 211s 3s/step - loss: 0.2468 - ac
curacy: 0.9180 - val_loss: 3.2965 - val_accuracy: 0.5226
Epoch 29/30
79/79 [==============================] - 213s 3s/step - loss: 0.2514 - ac
curacy: 0.9263 - val_loss: 2.9613 - val_accuracy: 0.5226
Epoch 30/30
79/79 [==============================] - 220s 3s/step - loss: 0.1597 - ac
curacy: 0.9481 - val_loss: 3.4829 - val_accuracy: 0.5548
```

Optimize the above model

In [ ]:
```
input_shape = (32, 227, 227, 1)
n_classes = 10

model_2 = models.Sequential([
    resize_and_rescale,
    layers.Conv2D(32, kernel_size = (3,3), activation='relu', input_shape=inp
    layers.MaxPooling2D((2, 2)),
    layers.Dropout(0.2),
    layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Dropout(0.2),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax'),
])

model_2.build(input_shape=input_shape)
```

In [ ]:
```
model_2.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)
```

In [ ]:

```
history = model_2.fit(
    train_ds_new,
    batch_size= 128,
    validation_data=valid_ds_new,
    verbose=1,
    epochs=30,
)
```

```
Epoch 1/30
79/79 [==============================] - 246s 3s/step - loss: 2.0745 - ac
curacy: 0.2624 - val_loss: 1.9996 - val_accuracy: 0.2806
Epoch 2/30
79/79 [==============================] - 233s 3s/step - loss: 1.9296 - ac
curacy: 0.2989 - val_loss: 1.7887 - val_accuracy: 0.3387
Epoch 3/30
79/79 [==============================] - 232s 3s/step - loss: 1.7738 - ac
curacy: 0.3460 - val_loss: 1.6920 - val_accuracy: 0.3774
Epoch 4/30
79/79 [==============================] - 233s 3s/step - loss: 1.7008 - ac
curacy: 0.3690 - val_loss: 1.6117 - val_accuracy: 0.3839
Epoch 5/30
79/79 [==============================] - 233s 3s/step - loss: 1.6713 - ac
curacy: 0.3753 - val_loss: 1.5952 - val_accuracy: 0.4032
Epoch 6/30
79/79 [==============================] - 235s 3s/step - loss: 1.6198 - ac
curacy: 0.3912 - val_loss: 1.5607 - val_accuracy: 0.4161
Epoch 7/30
79/79 [==============================] - 235s 3s/step - loss: 1.5724 - ac
curacy: 0.4178 - val_loss: 1.5059 - val_accuracy: 0.4516
Epoch 8/30
79/79 [==============================] - 235s 3s/step - loss: 1.5157 - ac
curacy: 0.4372 - val_loss: 1.4438 - val_accuracy: 0.4548
Epoch 9/30
79/79 [==============================] - 233s 3s/step - loss: 1.4297 - ac
curacy: 0.4574 - val_loss: 1.4164 - val_accuracy: 0.4581
Epoch 10/30
79/79 [==============================] - 230s 3s/step - loss: 1.3980 - ac
curacy: 0.4804 - val_loss: 1.4094 - val_accuracy: 0.4742
Epoch 11/30
79/79 [==============================] - 235s 3s/step - loss: 1.3493 - ac
curacy: 0.4855 - val_loss: 1.3567 - val_accuracy: 0.4935
Epoch 12/30
79/79 [==============================] - 234s 3s/step - loss: 1.3045 - ac
curacy: 0.5180 - val_loss: 1.3148 - val_accuracy: 0.4839
Epoch 13/30
79/79 [==============================] - 235s 3s/step - loss: 1.2376 - ac
curacy: 0.5319 - val_loss: 1.3130 - val_accuracy: 0.5226
Epoch 14/30
79/79 [==============================] - 231s 3s/step - loss: 1.2038 - ac
curacy: 0.5478 - val_loss: 1.3113 - val_accuracy: 0.5194
Epoch 15/30
79/79 [==============================] - 230s 3s/step - loss: 1.1292 - ac
curacy: 0.5822 - val_loss: 1.2749 - val_accuracy: 0.5129
Epoch 16/30
79/79 [==============================] - 228s 3s/step - loss: 1.1023 - ac
```

```
curacy: 0.5846 - val_loss: 1.2585 - val_accuracy: 0.5581
Epoch 17/30
79/79 [==============================] - 229s 3s/step - loss: 1.0468 - ac
curacy: 0.6128 - val_loss: 1.2639 - val_accuracy: 0.5645
Epoch 18/30
79/79 [==============================] - 226s 3s/step - loss: 0.9803 - ac
curacy: 0.6350 - val_loss: 1.2994 - val_accuracy: 0.5452
Epoch 19/30
79/79 [==============================] - 228s 3s/step - loss: 0.9449 - ac
curacy: 0.6441 - val_loss: 1.2854 - val_accuracy: 0.5613
Epoch 20/30
79/79 [==============================] - 227s 3s/step - loss: 0.9178 - ac
curacy: 0.6686 - val_loss: 1.3003 - val_accuracy: 0.5742
Epoch 21/30
79/79 [==============================] - 228s 3s/step - loss: 0.8734 - ac
curacy: 0.6829 - val_loss: 1.3491 - val_accuracy: 0.5710
Epoch 22/30
79/79 [==============================] - 227s 3s/step - loss: 0.8081 - ac
curacy: 0.7051 - val_loss: 1.4008 - val_accuracy: 0.5548
Epoch 23/30
79/79 [==============================] - 227s 3s/step - loss: 0.7799 - ac
curacy: 0.7218 - val_loss: 1.3394 - val_accuracy: 0.5774
Epoch 24/30
79/79 [==============================] - 226s 3s/step - loss: 0.7397 - ac
curacy: 0.7285 - val_loss: 1.4589 - val_accuracy: 0.5355
Epoch 25/30
79/79 [==============================] - 230s 3s/step - loss: 0.7166 - ac
curacy: 0.7483 - val_loss: 1.4676 - val_accuracy: 0.5355
Epoch 26/30
79/79 [==============================] - 232s 3s/step - loss: 0.6862 - ac
curacy: 0.7566 - val_loss: 1.5041 - val_accuracy: 0.5581
Epoch 27/30
79/79 [==============================] - 231s 3s/step - loss: 0.6596 - ac
curacy: 0.7527 - val_loss: 1.4521 - val_accuracy: 0.5581
Epoch 28/30
79/79 [==============================] - 230s 3s/step - loss: 0.6343 - ac
curacy: 0.7646 - val_loss: 1.5032 - val_accuracy: 0.5581
Epoch 29/30
79/79 [==============================] - 235s 3s/step - loss: 0.6315 - ac
curacy: 0.7638 - val_loss: 1.5909 - val_accuracy: 0.5548
Epoch 30/30
79/79 [==============================] - 234s 3s/step - loss: 0.5958 - ac
curacy: 0.7808 - val_loss: 1.6020 - val_accuracy: 0.5806
```

In [ ]:

```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt

from tensorflow.keras import Model
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.losses import categorical_crossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dro
```

Try AlexNet without grayscale images

In [ ]:
```
#AlexNet
class AlexNet(Sequential):
    def __init__(self, input_shape, num_classes):
        super().__init__()

        self.add(Conv2D(96, kernel_size=(11,11), strides= 4,
                        padding= 'valid', activation= 'relu',
                        input_shape= input_shape,
                        kernel_initializer= 'he_normal'))
        self.add(MaxPooling2D(pool_size=(3,3), strides= (2,2),
                              padding= 'valid', data_format= None))

        self.add(Conv2D(256, kernel_size=(5,5), strides= 1,
                        padding= 'same', activation= 'relu',
                        kernel_initializer= 'he_normal'))
        self.add(MaxPooling2D(pool_size=(3,3), strides= (2,2),
                              padding= 'valid', data_format= None))
        self.add(Conv2D(384, kernel_size=(3,3), strides= 1,
                        padding= 'same', activation= 'relu',
                        kernel_initializer= 'he_normal'))

        self.add(Conv2D(384, kernel_size=(3,3), strides= 1,
                        padding= 'same', activation= 'relu',
                        kernel_initializer= 'he_normal'))

        self.add(Conv2D(256, kernel_size=(3,3), strides= 1,
                        padding= 'same', activation= 'relu',
                        kernel_initializer= 'he_normal'))

        self.add(MaxPooling2D(pool_size=(3,3), strides= (2,2),
                              padding= 'valid', data_format= None))
        self.add(Flatten())
        self.add(Dense(4096, activation= 'relu'))
        self.add(Dense(4096, activation= 'relu'))
        self.add(Dense(1000, activation= 'relu'))
        self.add(Dense(10, activation= 'softmax'))

        self.compile(optimizer= tf.keras.optimizers.Adam(0.001),
                     loss='sparse_categorical_crossentropy',
                     metrics=['accuracy'])
```

In [ ]:
```
num_classes = 10
model_alex = AlexNet((227, 227, 3), num_classes)
```

In [ ]:
```
model_alex.fit(train_ds,
                    epochs= 30,
                    validation_data=val_ds,
                    batch_size = 32,
                    verbose=1)
```

```
Epoch 1/30
79/79 [==============================] - 596s 7s/step - loss: 692.5383 - ac
curacy: 0.2556 - val_loss: 1.8141 - val_accuracy: 0.3806
Epoch 2/30
79/79 [==============================] - 589s 7s/step - loss: 1.8867 - accu
racy: 0.3432 - val_loss: 1.7390 - val_accuracy: 0.4000
Epoch 3/30
79/79 [==============================] - 609s 7s/step - loss: 1.7746 - accu
racy: 0.3670 - val_loss: 1.6592 - val_accuracy: 0.4161
Epoch 4/30
79/79 [==============================] - 674s 8s/step - loss: 1.7260 - accu
racy: 0.3908 - val_loss: 1.6114 - val_accuracy: 0.4387
Epoch 5/30
79/79 [==============================] - 661s 8s/step - loss: 1.7087 - accu
racy: 0.3884 - val_loss: 1.6918 - val_accuracy: 0.4258
Epoch 6/30
79/79 [==============================] - 680s 8s/step - loss: 1.7732 - accu
racy: 0.3642 - val_loss: 1.6161 - val_accuracy: 0.4419
Epoch 7/30
79/79 [==============================] - 628s 8s/step - loss: 1.6943 - accu
racy: 0.3817 - val_loss: 1.7066 - val_accuracy: 0.3839
Epoch 8/30
79/79 [==============================] - 586s 7s/step - loss: 1.6831 - accu
racy: 0.3864 - val_loss: 1.6879 - val_accuracy: 0.3613
Epoch 9/30
79/79 [==============================] - 581s 7s/step - loss: 1.6838 - accu
racy: 0.3908 - val_loss: 1.6371 - val_accuracy: 0.4129
Epoch 10/30
79/79 [==============================] - 583s 7s/step - loss: 1.6846 - accu
racy: 0.3757 - val_loss: 1.6129 - val_accuracy: 0.4516
Epoch 11/30
79/79 [==============================] - 602s 7s/step - loss: 1.6329 - accu
racy: 0.4114 - val_loss: 1.6067 - val_accuracy: 0.4032
Epoch 12/30
79/79 [==============================] - 589s 7s/step - loss: 1.6347 - accu
racy: 0.4082 - val_loss: 1.6479 - val_accuracy: 0.3968
Epoch 13/30
79/79 [==============================] - 583s 7s/step - loss: 1.5903 - accu
racy: 0.4253 - val_loss: 1.5907 - val_accuracy: 0.4161
Epoch 14/30
79/79 [==============================] - 592s 7s/step - loss: 1.6140 - accu
racy: 0.4043 - val_loss: 1.5454 - val_accuracy: 0.4484
Epoch 15/30
79/79 [==============================] - 586s 7s/step - loss: 1.5500 - accu
racy: 0.4459 - val_loss: 1.5656 - val_accuracy: 0.4581
Epoch 16/30
79/79 [==============================] - 587s 7s/step - loss: 1.5379 - accu
racy: 0.4372 - val_loss: 1.5224 - val_accuracy: 0.4548
Epoch 17/30
79/79 [==============================] - 584s 7s/step - loss: 1.4810 - accu
```

```
racy: 0.4594 - val_loss: 1.5589 - val_accuracy: 0.4581
Epoch 18/30
79/79 [==============================] - 593s 7s/step - loss: 1.5168 - accu
racy: 0.4487 - val_loss: 1.5986 - val_accuracy: 0.4677
Epoch 19/30
79/79 [==============================] - 650s 8s/step - loss: 1.4799 - accu
racy: 0.4558 - val_loss: 1.5543 - val_accuracy: 0.4258
Epoch 20/30
79/79 [==============================] - 593s 7s/step - loss: 1.4715 - accu
racy: 0.4594 - val_loss: 1.5444 - val_accuracy: 0.4484
Epoch 21/30
79/79 [==============================] - 598s 7s/step - loss: 1.4688 - accu
racy: 0.4546 - val_loss: 1.5527 - val_accuracy: 0.4677
Epoch 22/30
79/79 [==============================] - 602s 7s/step - loss: 1.4124 - accu
racy: 0.4836 - val_loss: 1.5917 - val_accuracy: 0.4355
Epoch 23/30
79/79 [==============================] - 607s 7s/step - loss: 1.3824 - accu
racy: 0.4879 - val_loss: 1.5522 - val_accuracy: 0.5065
Epoch 24/30
79/79 [==============================] - 615s 7s/step - loss: 1.3320 - accu
racy: 0.5010 - val_loss: 1.6279 - val_accuracy: 0.4355
Epoch 25/30
79/79 [==============================] - 617s 8s/step - loss: 1.3916 - accu
racy: 0.4899 - val_loss: 1.5989 - val_accuracy: 0.4129
Epoch 26/30
79/79 [==============================] - 608s 7s/step - loss: 1.3931 - accu
racy: 0.4788 - val_loss: 1.5805 - val_accuracy: 0.4000
Epoch 27/30
79/79 [==============================] - 617s 8s/step - loss: 1.2978 - accu
racy: 0.5200 - val_loss: 1.6037 - val_accuracy: 0.4323
Epoch 28/30
79/79 [==============================] - 633s 8s/step - loss: 1.2348 - accu
racy: 0.5371 - val_loss: 1.5266 - val_accuracy: 0.4935
Epoch 29/30
79/79 [==============================] - 622s 8s/step - loss: 1.1765 - accu
racy: 0.5561 - val_loss: 1.7103 - val_accuracy: 0.4645
Epoch 30/30
79/79 [==============================] - 614s 7s/step - loss: 1.1734 - accu
racy: 0.5620 - val_loss: 1.8840 - val_accuracy: 0.4419
```

Out[15]: <keras.callbacks.History at 0x7f0e1829f990>

In [ ]:
```
model_alex.save("/content/drive/MyDrive/alex_net.h5")
```

Try LeNet

In [ ]:
```
#learning_rate
#define learning rate scheduler
from keras.callbacks import ReduceLROnPlateau
reduce_lr = ReduceLROnPlateau(monitor='loss', factor=0.2, patience=2, min_lr=
```

In [ ]:
```
#LeNet
model_LeNet = Sequential()
model_LeNet.add(Conv2D(6, kernel_size=(5, 5), activation='relu', input_shape=
model_LeNet.add(MaxPooling2D(pool_size=(2, 2)))
model_LeNet.add(Conv2D(16, kernel_size=(5, 5), activation='relu'))
model_LeNet.add(MaxPooling2D(pool_size=(2, 2)))
model_LeNet.add(Flatten())
model_LeNet.add(Dense(120, activation='relu'))
model_LeNet.add(Dense(84, activation='relu'))
model_LeNet.add(Dense(10, activation='softmax'))
```

In [ ]:
```
import keras
model_LeNet.compile(loss="sparse_categorical_crossentropy", optimizer= "adam"
```

In [ ]:
```
model_LeNet.fit(train_ds_new,
                    epochs= 30,
                    validation_data=valid_ds_new,
                    batch_size = 32,
                    callbacks = [reduce_lr],
                    verbose=1)
```

```
Epoch 1/30
79/79 [==============================] - 72s 892ms/step - loss: 45.2327 -
accuracy: 0.2481 - val_loss: 1.9272 - val_accuracy: 0.3613 - lr: 0.0010
Epoch 2/30
79/79 [==============================] - 71s 896ms/step - loss: 1.5899 -
accuracy: 0.4388 - val_loss: 1.7841 - val_accuracy: 0.3677 - lr: 0.0010
Epoch 3/30
79/79 [==============================] - 70s 892ms/step - loss: 1.2064 -
accuracy: 0.5351 - val_loss: 1.9528 - val_accuracy: 0.3968 - lr: 0.0010
Epoch 4/30
79/79 [==============================] - 70s 884ms/step - loss: 1.0072 -
accuracy: 0.6017 - val_loss: 2.3955 - val_accuracy: 0.3581 - lr: 0.0010
Epoch 5/30
79/79 [==============================] - 70s 888ms/step - loss: 0.9541 -
accuracy: 0.6568 - val_loss: 2.1969 - val_accuracy: 0.3710 - lr: 0.0010
Epoch 6/30
79/79 [==============================] - 70s 884ms/step - loss: 0.8188 -
accuracy: 0.7146 - val_loss: 2.2400 - val_accuracy: 0.4097 - lr: 0.0010
Epoch 7/30
79/79 [==============================] - 90s 1s/step - loss: 0.6097 - acc
uracy: 0.8006 - val_loss: 2.9645 - val_accuracy: 0.4032 - lr: 0.0010
Epoch 8/30
79/79 [==============================] - 68s 867ms/step - loss: 0.4457 -
accuracy: 0.8660 - val_loss: 3.0449 - val_accuracy: 0.4194 - lr: 0.0010
Epoch 9/30
79/79 [==============================] - 68s 866ms/step - loss: 0.3311 -
accuracy: 0.8938 - val_loss: 3.2840 - val_accuracy: 0.4516 - lr: 0.0010
Epoch 10/30
79/79 [==============================] - 69s 868ms/step - loss: 0.3279 -
accuracy: 0.9005 - val_loss: 3.5046 - val_accuracy: 0.4516 - lr: 0.0010
Epoch 11/30
79/79 [==============================] - 69s 870ms/step - loss: 0.2648 -
accuracy: 0.9180 - val_loss: 4.0762 - val_accuracy: 0.4129 - lr: 0.0010
Epoch 12/30
79/79 [==============================] - 68s 864ms/step - loss: 0.2259 -
accuracy: 0.9287 - val_loss: 3.6371 - val_accuracy: 0.4065 - lr: 0.0010
Epoch 13/30
79/79 [==============================] - 69s 873ms/step - loss: 0.2048 -
accuracy: 0.9449 - val_loss: 3.5770 - val_accuracy: 0.4129 - lr: 0.0010
Epoch 14/30
79/79 [==============================] - 69s 878ms/step - loss: 0.2474 -
accuracy: 0.9405 - val_loss: 4.2214 - val_accuracy: 0.4258 - lr: 0.0010
Epoch 15/30
79/79 [==============================] - 69s 875ms/step - loss: 0.2263 -
accuracy: 0.9350 - val_loss: 3.8327 - val_accuracy: 0.4516 - lr: 0.0010
Epoch 16/30
79/79 [==============================] - 69s 868ms/step - loss: 0.0914 -
accuracy: 0.9707 - val_loss: 4.2843 - val_accuracy: 0.4484 - lr: 2.0000e-
04
```

```
Epoch 17/30
79/79 [==============================] - 68s 866ms/step - loss: 0.0574 -
accuracy: 0.9810 - val_loss: 4.4038 - val_accuracy: 0.4484 - lr: 2.0000e-
04
Epoch 18/30
79/79 [==============================] - 68s 866ms/step - loss: 0.0463 -
accuracy: 0.9857 - val_loss: 4.5409 - val_accuracy: 0.4452 - lr: 2.0000e-
04
Epoch 19/30
79/79 [==============================] - 68s 865ms/step - loss: 0.0394 -
accuracy: 0.9877 - val_loss: 4.7914 - val_accuracy: 0.4419 - lr: 2.0000e-
04
Epoch 20/30
79/79 [==============================] - 68s 867ms/step - loss: 0.0345 -
accuracy: 0.9889 - val_loss: 4.9527 - val_accuracy: 0.4419 - lr: 2.0000e-
04
Epoch 21/30
79/79 [==============================] - 68s 867ms/step - loss: 0.0306 -
accuracy: 0.9905 - val_loss: 5.1312 - val_accuracy: 0.4387 - lr: 2.0000e-
04
Epoch 22/30
79/79 [==============================] - 68s 866ms/step - loss: 0.0269 -
accuracy: 0.9929 - val_loss: 5.3050 - val_accuracy: 0.4387 - lr: 2.0000e-
04
Epoch 23/30
79/79 [==============================] - 69s 869ms/step - loss: 0.0239 -
accuracy: 0.9937 - val_loss: 5.5715 - val_accuracy: 0.4355 - lr: 2.0000e-
04
Epoch 24/30
79/79 [==============================] - 69s 880ms/step - loss: 0.0210 -
accuracy: 0.9952 - val_loss: 5.8835 - val_accuracy: 0.4452 - lr: 2.0000e-
04
Epoch 25/30
79/79 [==============================] - 70s 881ms/step - loss: 0.0187 -
accuracy: 0.9956 - val_loss: 5.8375 - val_accuracy: 0.4387 - lr: 2.0000e-
04
Epoch 26/30
79/79 [==============================] - 69s 873ms/step - loss: 0.0174 -
accuracy: 0.9960 - val_loss: 6.2446 - val_accuracy: 0.4387 - lr: 2.0000e-
04
Epoch 27/30
79/79 [==============================] - 70s 881ms/step - loss: 0.0150 -
accuracy: 0.9960 - val_loss: 6.3748 - val_accuracy: 0.4419 - lr: 2.0000e-
04
Epoch 28/30
79/79 [==============================] - 69s 878ms/step - loss: 0.0131 -
accuracy: 0.9960 - val_loss: 6.7055 - val_accuracy: 0.4452 - lr: 2.0000e-
04
Epoch 29/30
79/79 [==============================] - 69s 877ms/step - loss: 0.0166 -
accuracy: 0.9960 - val_loss: 6.3423 - val_accuracy: 0.4323 - lr: 2.0000e-
04
Epoch 30/30
79/79 [==============================] - 71s 904ms/step - loss: 0.0110 -
accuracy: 0.9964 - val_loss: 6.9840 - val_accuracy: 0.4452 - lr: 2.0000e-
04
```

Out[42]: <keras.callbacks.History at 0x7f59cc103510>

Optimize LeNet

In [ ]:
```
#LeNet
model_LeNet_2 = Sequential()
model_LeNet_2.add(Conv2D(6, kernel_size=(5, 5), activation='relu', input_shap
model_LeNet_2.add(MaxPooling2D(pool_size=(2, 2)))
model_LeNet_2.add(Dropout(0.2))
model_LeNet_2.add(Conv2D(16, kernel_size=(5, 5), activation='relu'))
model_LeNet_2.add(MaxPooling2D(pool_size=(2, 2)))
model_LeNet_2.add(Dropout(0.2))
model_LeNet_2.add(Flatten())
model_LeNet_2.add(Dense(120, activation='relu'))
model_LeNet_2.add(Dense(84, activation='relu'))
model_LeNet_2.add(Dense(10, activation='softmax'))
```

In [ ]:
```
model_LeNet_2.compile(loss="sparse_categorical_crossentropy", optimizer= "ada
```

In [ ]:
```
model_LeNet_2.fit(train_ds_new,
                  epochs= 30,
                  validation_data=valid_ds_new,
                  batch_size = 32,
                  callbacks = [reduce_lr],
                  verbose=1)
```

```
Epoch 1/30
79/79 [==============================] - 305s 3s/step - loss: 159.4847 -
accuracy: 0.2081 - val_loss: 2.2353 - val_accuracy: 0.2065 - lr: 0.0010
Epoch 2/30
79/79 [==============================] - 112s 1s/step - loss: 1.9889 - ac
curacy: 0.3286 - val_loss: 1.9782 - val_accuracy: 0.3484 - lr: 0.0010
Epoch 3/30
79/79 [==============================] - 129s 2s/step - loss: 1.5896 - ac
curacy: 0.4265 - val_loss: 1.8617 - val_accuracy: 0.3839 - lr: 0.0010
Epoch 4/30
79/79 [==============================] - 155s 2s/step - loss: 1.3388 - ac
curacy: 0.5121 - val_loss: 1.7588 - val_accuracy: 0.4161 - lr: 0.0010
Epoch 5/30
79/79 [==============================] - 129s 2s/step - loss: 1.1466 - ac
curacy: 0.5898 - val_loss: 1.7482 - val_accuracy: 0.4226 - lr: 0.0010
Epoch 6/30
79/79 [==============================] - 134s 2s/step - loss: 0.9620 - ac
curacy: 0.6397 - val_loss: 1.8036 - val_accuracy: 0.4097 - lr: 0.0010
Epoch 7/30
79/79 [==============================] - 122s 2s/step - loss: 0.7924 - ac
curacy: 0.7087 - val_loss: 1.7833 - val_accuracy: 0.4581 - lr: 0.0010
Epoch 8/30
79/79 [==============================] - 126s 2s/step - loss: 0.6693 - ac
curacy: 0.7578 - val_loss: 1.7938 - val_accuracy: 0.4581 - lr: 0.0010
Epoch 9/30
79/79 [==============================] - 113s 1s/step - loss: 0.4862 - ac
curacy: 0.8216 - val_loss: 1.7353 - val_accuracy: 0.5000 - lr: 0.0010
Epoch 10/30
79/79 [==============================] - 116s 1s/step - loss: 0.3952 - ac
curacy: 0.8541 - val_loss: 2.0779 - val_accuracy: 0.4839 - lr: 0.0010
Epoch 11/30
79/79 [==============================] - 112s 1s/step - loss: 0.3387 - ac
curacy: 0.8831 - val_loss: 1.9628 - val_accuracy: 0.5000 - lr: 0.0010
Epoch 12/30
79/79 [==============================] - 114s 1s/step - loss: 0.2760 - ac
curacy: 0.9037 - val_loss: 1.9912 - val_accuracy: 0.5000 - lr: 0.0010
Epoch 13/30
79/79 [==============================] - 132s 2s/step - loss: 0.2221 - ac
curacy: 0.9239 - val_loss: 2.0185 - val_accuracy: 0.5000 - lr: 0.0010
Epoch 14/30
79/79 [==============================] - 150s 2s/step - loss: 0.2109 - ac
curacy: 0.9298 - val_loss: 1.9425 - val_accuracy: 0.5516 - lr: 0.0010
Epoch 15/30
79/79 [==============================] - 122s 2s/step - loss: 0.2386 - ac
curacy: 0.9342 - val_loss: 2.2623 - val_accuracy: 0.5290 - lr: 0.0010
Epoch 16/30
79/79 [==============================] - 122s 2s/step - loss: 0.1820 - ac
curacy: 0.9461 - val_loss: 2.2098 - val_accuracy: 0.5323 - lr: 0.0010
Epoch 17/30
```

```
79/79 [==============================] - 119s 2s/step - loss: 0.1604 - ac
curacy: 0.9512 - val_loss: 2.4418 - val_accuracy: 0.5419 - lr: 0.0010
Epoch 18/30
79/79 [==============================] - 119s 1s/step - loss: 0.1303 - ac
curacy: 0.9639 - val_loss: 2.4299 - val_accuracy: 0.5355 - lr: 0.0010
Epoch 19/30
79/79 [==============================] - 123s 2s/step - loss: 0.1183 - ac
curacy: 0.9663 - val_loss: 2.4601 - val_accuracy: 0.5290 - lr: 0.0010
Epoch 20/30
79/79 [==============================] - 117s 1s/step - loss: 0.1375 - ac
curacy: 0.9604 - val_loss: 2.3170 - val_accuracy: 0.5516 - lr: 0.0010
Epoch 21/30
79/79 [==============================] - 116s 1s/step - loss: 0.0917 - ac
curacy: 0.9754 - val_loss: 2.5820 - val_accuracy: 0.5258 - lr: 0.0010
Epoch 22/30
79/79 [==============================] - 118s 1s/step - loss: 0.0636 - ac
curacy: 0.9806 - val_loss: 2.8302 - val_accuracy: 0.5161 - lr: 0.0010
Epoch 23/30
79/79 [==============================] - 119s 2s/step - loss: 0.0881 - ac
curacy: 0.9719 - val_loss: 3.0427 - val_accuracy: 0.5387 - lr: 0.0010
Epoch 24/30
79/79 [==============================] - 116s 1s/step - loss: 0.0925 - ac
curacy: 0.9766 - val_loss: 2.4604 - val_accuracy: 0.5484 - lr: 0.0010
Epoch 25/30
79/79 [==============================] - 118s 1s/step - loss: 0.0510 - ac
curacy: 0.9845 - val_loss: 2.4991 - val_accuracy: 0.5484 - lr: 2.0000e-04
Epoch 26/30
79/79 [==============================] - 115s 1s/step - loss: 0.0295 - ac
curacy: 0.9925 - val_loss: 2.5194 - val_accuracy: 0.5581 - lr: 2.0000e-04
Epoch 27/30
79/79 [==============================] - 114s 1s/step - loss: 0.0229 - ac
curacy: 0.9933 - val_loss: 2.5821 - val_accuracy: 0.5484 - lr: 2.0000e-04
Epoch 28/30
79/79 [==============================] - 119s 2s/step - loss: 0.0241 - ac
curacy: 0.9937 - val_loss: 2.6118 - val_accuracy: 0.5742 - lr: 2.0000e-04
Epoch 29/30
79/79 [==============================] - 113s 1s/step - loss: 0.0255 - ac
curacy: 0.9925 - val_loss: 2.5997 - val_accuracy: 0.5613 - lr: 2.0000e-04
Epoch 30/30
79/79 [==============================] - 113s 1s/step - loss: 0.0198 - ac
curacy: 0.9941 - val_loss: 2.6345 - val_accuracy: 0.5645 - lr: 4.0000e-05
```

Out[36]: <keras.callbacks.History at 0x7f0e170caed0>

Compare with resnet 50

In [ ]:
```
train_ds_new_2= train_dataset.map(
    lambda x, y: (tf.image.resize(x, [224, 224]), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)
```

In [ ]:
```
val_ds_new_2 = valid_dataset.map(
    lambda x, y: (tf.image.resize(x, [224, 224]), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)
```

In [ ]:
```
#transfer learning using image net
from tensorflow import keras
model_2 = keras.applications.resnet50.ResNet50(weights="imagenet")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-appli
cations/resnet/resnet50_weights_tf_dim_ordering_tf_kernels.h5 (https://stor
age.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf
_dim_ordering_tf_kernels.h5)
102973440/102967424 [==============================] - 1s 0us/step
102981632/102967424 [==============================] - 1s 0us/step
```

In [ ]:
```
base_model = keras.applications.resnet50.ResNet50(weights="imagenet", include
avg = keras.layers.GlobalAveragePooling2D()(base_model.output)
output = keras.layers.Dense(10, activation="softmax")(avg)
model_3 = keras.Model(inputs=base_model.input, outputs=output)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-appli
cations/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5 (http
s://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_we
ights_tf_dim_ordering_tf_kernels_notop.h5)
94773248/94765736 [==============================] - 1s 0us/step
94781440/94765736 [==============================] - 1s 0us/step
```

In [ ]:
```
for layer in base_model.layers:
 layer.trainable = False

optimizer = keras.optimizers.SGD(lr=0.2, momentum=0.9, decay=0.01)
model_3.compile(loss="sparse_categorical_crossentropy", optimizer= "adam",
 metrics=["accuracy"])
history = model_3.fit(train_ds_new_2, epochs=20, validation_data= val_ds_new_
```

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descent.
py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate` i
nstead.
  super(SGD, self).__init__(name, **kwargs)

Epoch 1/20
79/79 [==============================] - 773s 8s/step - loss: 1.5079 - accu
racy: 0.4970 - val_loss: 0.9870 - val_accuracy: 0.6742
Epoch 2/20
79/79 [==============================] - 385s 5s/step - loss: 0.9766 - accu
racy: 0.6635 - val_loss: 0.8715 - val_accuracy: 0.7226
Epoch 3/20
79/79 [==============================] - 385s 5s/step - loss: 0.7989 - accu
racy: 0.7321 - val_loss: 0.7797 - val_accuracy: 0.7290
Epoch 4/20
79/79 [==============================] - 384s 5s/step - loss: 0.6898 - accu
racy: 0.7697 - val_loss: 0.7418 - val_accuracy: 0.7452
Epoch 5/20
79/79 [==============================] - 383s 5s/step - loss: 0.6133 - accu
racy: 0.8006 - val_loss: 0.7030 - val_accuracy: 0.7387
Epoch 6/20
79/79 [==============================] - 383s 5s/step - loss: 0.5576 - accu
racy: 0.8189 - val_loss: 0.6874 - val_accuracy: 0.7548
Epoch 7/20
79/79 [==============================] - 383s 5s/step - loss: 0.4858 - accu
racy: 0.8537 - val_loss: 0.6888 - val_accuracy: 0.7613
Epoch 8/20
79/79 [==============================] - 384s 5s/step - loss: 0.4434 - accu
racy: 0.8676 - val_loss: 0.6289 - val_accuracy: 0.7871
Epoch 9/20
79/79 [==============================] - 384s 5s/step - loss: 0.4116 - accu
racy: 0.8791 - val_loss: 0.6365 - val_accuracy: 0.7742
Epoch 10/20
79/79 [==============================] - 383s 5s/step - loss: 0.3776 - accu
racy: 0.8950 - val_loss: 0.6353 - val_accuracy: 0.7774
Epoch 11/20
79/79 [==============================] - 384s 5s/step - loss: 0.3498 - accu
racy: 0.9112 - val_loss: 0.6420 - val_accuracy: 0.7677
Epoch 12/20
79/79 [==============================] - 384s 5s/step - loss: 0.3262 - accu
racy: 0.9164 - val_loss: 0.6118 - val_accuracy: 0.7806
Epoch 13/20
79/79 [==============================] - 384s 5s/step - loss: 0.2994 - accu
racy: 0.9283 - val_loss: 0.6491 - val_accuracy: 0.7677
Epoch 14/20
79/79 [==============================] - 384s 5s/step - loss: 0.2862 - accu
racy: 0.9310 - val_loss: 0.5916 - val_accuracy: 0.8032
Epoch 15/20
```

```
79/79 [==============================] - 385s 5s/step - loss: 0.2626 - accu
racy: 0.9437 - val_loss: 0.6135 - val_accuracy: 0.7871
Epoch 16/20
79/79 [==============================] - 385s 5s/step - loss: 0.2553 - accu
racy: 0.9429 - val_loss: 0.6294 - val_accuracy: 0.7742
Epoch 17/20
79/79 [==============================] - 385s 5s/step - loss: 0.2342 - accu
racy: 0.9501 - val_loss: 0.5802 - val_accuracy: 0.8032
Epoch 18/20
79/79 [==============================] - 385s 5s/step - loss: 0.2227 - accu
racy: 0.9556 - val_loss: 0.5821 - val_accuracy: 0.8290
Epoch 19/20
79/79 [==============================] - 393s 5s/step - loss: 0.2074 - accu
racy: 0.9608 - val_loss: 0.6173 - val_accuracy: 0.7903
Epoch 20/20
79/79 [==============================] - 397s 5s/step - loss: 0.1980 - accu
racy: 0.9620 - val_loss: 0.5945 - val_accuracy: 0.7839
```

In [ ]:
```
model_3.save("/content/drive/MyDrive/res50_net.h5")
```

In [ ]:
```
model_3_call = tf.keras.models.load_model('/content/drive/MyDrive/res50_net.h
```

In [ ]:
```
checkpoint_cb = keras.callbacks.ModelCheckpoint("/content/drive/MyDrive/model
```

In [ ]:
```
for layer in base_model.layers:
 layer.trainable = True

optimizer = keras.optimizers.SGD(lr=0.2, momentum=0.9, decay=0.01)
model_3_call.compile(loss="sparse_categorical_crossentropy", optimizer="adam"
 metrics=["accuracy"])
history = model_3_call.fit(train_ds_new_2, epochs=30, validation_data= val_ds
```

```
Epoch 1/30

/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descent.
py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate` i
nstead.
  super(SGD, self).__init__(name, **kwargs)

79/79 [==============================] - 766s 8s/step - loss: 0.2104 - accu
racy: 0.9501 - val_loss: 0.6575 - val_accuracy: 0.7677
Epoch 2/30
79/79 [==============================] - 431s 5s/step - loss: 0.1871 - accu
racy: 0.9639 - val_loss: 0.6715 - val_accuracy: 0.7839
Epoch 3/30
79/79 [==============================] - 437s 6s/step - loss: 0.1698 - accu
racy: 0.9707 - val_loss: 0.6177 - val_accuracy: 0.7806
Epoch 4/30
79/79 [==============================] - 437s 6s/step - loss: 0.1555 - accu
racy: 0.9738 - val_loss: 0.6259 - val_accuracy: 0.7871
Epoch 5/30
79/79 [==============================] - 439s 6s/step - loss: 0.1431 - accu
racy: 0.9774 - val_loss: 0.6164 - val_accuracy: 0.7871
Epoch 6/30
79/79 [==============================] - 434s 5s/step - loss: 0.1341 - accu
racy: 0.9806 - val_loss: 0.6283 - val_accuracy: 0.7871
Epoch 7/30
79/79 [==============================] - 439s 6s/step - loss: 0.1244 - accu
racy: 0.9853 - val_loss: 0.6685 - val_accuracy: 0.7774
Epoch 8/30
79/79 [==============================] - 436s 6s/step - loss: 0.1193 - accu
racy: 0.9857 - val_loss: 0.6417 - val_accuracy: 0.7806
Epoch 9/30
79/79 [==============================] - 435s 6s/step - loss: 0.1111 - accu
racy: 0.9865 - val_loss: 0.6061 - val_accuracy: 0.7903
Epoch 10/30
79/79 [==============================] - 436s 6s/step - loss: 0.1053 - accu
racy: 0.9877 - val_loss: 0.5903 - val_accuracy: 0.8032
Epoch 11/30
79/79 [==============================] - 434s 5s/step - loss: 0.0954 - accu
racy: 0.9909 - val_loss: 0.6345 - val_accuracy: 0.7903
Epoch 12/30
79/79 [==============================] - 434s 5s/step - loss: 0.0934 - accu
racy: 0.9913 - val_loss: 0.6161 - val_accuracy: 0.7935
Epoch 13/30
79/79 [==============================] - 433s 5s/step - loss: 0.0874 - accu
racy: 0.9909 - val_loss: 0.6445 - val_accuracy: 0.7968
Epoch 14/30
79/79 [==============================] - 434s 6s/step - loss: 0.0814 - accu
racy: 0.9945 - val_loss: 0.6081 - val_accuracy: 0.7903
Epoch 15/30
```

```
79/79 [==============================] - 433s 5s/step - loss: 0.0777 - accu
racy: 0.9941 - val_loss: 0.6122 - val_accuracy: 0.8000
Epoch 16/30
79/79 [==============================] - 433s 5s/step - loss: 0.0734 - accu
racy: 0.9948 - val_loss: 0.6262 - val_accuracy: 0.7935
Epoch 17/30
79/79 [==============================] - 433s 5s/step - loss: 0.0719 - accu
racy: 0.9952 - val_loss: 0.6574 - val_accuracy: 0.7935
Epoch 18/30
79/79 [==============================] - 433s 5s/step - loss: 0.0673 - accu
racy: 0.9956 - val_loss: 0.6381 - val_accuracy: 0.8000
Epoch 19/30
79/79 [==============================] - 433s 5s/step - loss: 0.0640 - accu
racy: 0.9948 - val_loss: 0.6353 - val_accuracy: 0.7968
Epoch 20/30
79/79 [==============================] - 433s 5s/step - loss: 0.0589 - accu
racy: 0.9972 - val_loss: 0.6486 - val_accuracy: 0.7935
Epoch 21/30
79/79 [==============================] - 432s 5s/step - loss: 0.0578 - accu
racy: 0.9968 - val_loss: 0.6299 - val_accuracy: 0.7903
Epoch 22/30
79/79 [==============================] - 434s 6s/step - loss: 0.0544 - accu
racy: 0.9968 - val_loss: 0.6658 - val_accuracy: 0.8065
Epoch 23/30
79/79 [==============================] - 433s 5s/step - loss: 0.0543 - accu
racy: 0.9968 - val_loss: 0.6358 - val_accuracy: 0.7935
Epoch 24/30
79/79 [==============================] - 432s 5s/step - loss: 0.0500 - accu
racy: 0.9976 - val_loss: 0.6418 - val_accuracy: 0.7968
Epoch 25/30
79/79 [==============================] - 432s 5s/step - loss: 0.0497 - accu
racy: 0.9968 - val_loss: 0.6597 - val_accuracy: 0.7968
Epoch 26/30
79/79 [==============================] - 431s 5s/step - loss: 0.0456 - accu
racy: 0.9980 - val_loss: 0.6682 - val_accuracy: 0.7968
Epoch 27/30
79/79 [==============================] - 433s 5s/step - loss: 0.0436 - accu
racy: 0.9980 - val_loss: 0.6632 - val_accuracy: 0.7903
Epoch 28/30
79/79 [==============================] - 432s 5s/step - loss: 0.0408 - accu
racy: 0.9984 - val_loss: 0.6432 - val_accuracy: 0.7968
Epoch 29/30
79/79 [==============================] - 433s 5s/step - loss: 0.0397 - accu
racy: 0.9988 - val_loss: 0.6532 - val_accuracy: 0.8000
Epoch 30/30
79/79 [==============================] - 432s 5s/step - loss: 0.0381 - accu
racy: 0.9984 - val_loss: 0.6597 - val_accuracy: 0.8000
```

In [ ]:

In [ ]:
```
#predict on test set
```

In [ ]:
```
from tensorflow.keras.models import load_model
model_final_res50 = tf.keras.models.load_model('/content/drive/MyDrive/model_
```

In [ ]:
```
val_score = model_final_res50.evaluate(val_ds_new_2)
val_score
```

```
10/10 [==============================] - 161s 5s/step - loss: 0.5903 - accu
racy: 0.8032
```

Out[15]:
```
[0.590263307094574, 0.8032258152961731]
```

In [ ]:
```
test_score = model_final_res50.evaluate(test_dataset)
test_score
```

```
11/11 [==============================] - 120s 9s/step - loss: 1.0489 - accu
racy: 0.6800
```

Out[17]:
```
[1.048945426940918, 0.6800000071525574]
```