In [ ]:
```
from google.colab import drive, files
import pandas as pd
import numpy as np
import os
from PIL import Image
import tensorflow as tf
from keras import datasets, layers, models
import matplotlib.pyplot as plt
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential, load_model
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense, Activation, BatchNormalization
drive.mount('/content/drive')
```

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

In [ ]:
```
device_name = tf.test.gpu_device_name()
```

In [ ]:
```
#!unzip '/content/drive/MyDrive/Deep Learning/CarDetection.zip' -d '/content/drive/MyDrive/Deep Learning'
```

Data Analysis

In [ ]:
```
#Create Dataframes
namesDF = pd.read_csv('/content/drive/MyDrive/Deep Learning/names.csv', header=None)
namesDF.columns = ['Names']
trainLabels = pd.read_csv('/content/drive/MyDrive/Deep Learning/anno_train.csv', header=None)
testLabels = pd.read_csv('/content/drive/MyDrive/Deep Learning/anno_test.csv', header=None)
```

In [ ]:
```
namesDF.head(10)
```

Out[ ]:

	Names
0	AM General Hummer SUV 2000
1	Acura RL Sedan 2012
2	Acura TL Sedan 2012
3	Acura TL Type-S 2008

	Names
4	Acura TSX Sedan 2012
5	Acura Integra Type R 2001
6	Acura ZDX Hatchback 2012
7	Aston Martin V8 Vantage Convertible 2012
8	Aston Martin V8 Vantage Coupe 2012
9	Aston Martin Virage Convertible 2012

In [ ]:
```
trainLabels.head(10)
```

Out[ ]:

	0	1	2	3	4	5
0	00001.jpg	39	116	569	375	14
1	00002.jpg	36	116	868	587	3
2	00003.jpg	85	109	601	381	91
3	00004.jpg	621	393	1484	1096	134
4	00005.jpg	14	36	133	99	106
5	00006.jpg	259	289	515	416	123
6	00007.jpg	88	80	541	397	89
7	00008.jpg	73	79	591	410	96
8	00009.jpg	20	126	1269	771	167
9	00010.jpg	21	110	623	367	58

In [ ]:
```
testLabels.head(10)
```

Out[ ]:

	0	1	2	3	4	5
0	00001.jpg	30	52	246	147	181

	0	1	2	3	4	5
1	00002.jpg	100	19	576	203	103
2	00003.jpg	51	105	968	659	145
3	00004.jpg	67	84	581	407	187
4	00005.jpg	140	151	593	339	185
5	00006.jpg	20	77	420	301	78
6	00007.jpg	249	166	2324	1459	118
7	00008.jpg	119	215	1153	719	165
8	00009.jpg	1	7	275	183	32
9	00010.jpg	28	55	241	177	60

In [ ]:

```
#Not sure if the coordinate columns are necessary, so I will remove them for now.
trainLabels.drop(columns=trainLabels.columns[1:-1], axis=1, inplace=True)
trainLabels.columns = ['Image Label', 'Name List Index']
trainLabels.head(10)
```

Out[ ]:

	Image Label	Name List Index
0	00001.jpg	14
1	00002.jpg	3
2	00003.jpg	91
3	00004.jpg	134
4	00005.jpg	106
5	00006.jpg	123
6	00007.jpg	89
7	00008.jpg	96
8	00009.jpg	167
9	00010.jpg	58

In [ ]:

```
testLabels.drop(columns=testLabels.columns[1:-1], axis=1, inplace=True)
testLabels.columns = ['Image Label', 'Name List Index']
testLabels.head(10)
```

Out[ ]:

	Image Label	Name List Index
0	00001.jpg	181
1	00002.jpg	103
2	00003.jpg	145
3	00004.jpg	187
4	00005.jpg	185
5	00006.jpg	78
6	00007.jpg	118
7	00008.jpg	165
8	00009.jpg	32
9	00010.jpg	60

In [ ]:
```
#Make a dataframe including the names corresponding to the indexes.
#Here is the training set.
trainNames = []
for index in trainLabels['Name List Index']:
  name = namesDF['Names'][index-1]
  trainNames.append(name)

trainLabels['Names'] = trainNames
trainLabels.head(10)
```

Out[ ]:

	Image Label	Name List Index	Names
0	00001.jpg	14	Audi TTS Coupe 2012
1	00002.jpg	3	Acura TL Sedan 2012
2	00003.jpg	91	Dodge Dakota Club Cab 2007
3	00004.jpg	134	Hyundai Sonata Hybrid Sedan 2012
4	00005.jpg	106	Ford F-450 Super Duty Crew Cab 2012
5	00006.jpg	123	Geo Metro Convertible 1993
6	00007.jpg	89	Dodge Journey SUV 2012
7	00008.jpg	96	Dodge Charger Sedan 2012
8	00009.jpg	167	Mitsubishi Lancer Sedan 2012
9	00010.jpg	58	Chevrolet Traverse SUV 2012

In [ ]:
```
#Do same thing for test set.
testNames = []
for index in testLabels['Name List Index']:
  name = namesDF['Names'][index-1]
  testNames.append(name)

testLabels['Names'] = testNames
testLabels.head(10)
```

Out[ ]:

	Image Label	Name List Index	Names
0	00001.jpg	181	Suzuki Aerio Sedan 2007
1	00002.jpg	103	Ferrari 458 Italia Convertible 2012
2	00003.jpg	145	Jeep Patriot SUV 2012
3	00004.jpg	187	Toyota Camry Sedan 2012
4	00005.jpg	185	Tesla Model S Sedan 2012
5	00006.jpg	78	Chrysler Town and Country Minivan 2012
6	00007.jpg	118	GMC Terrain SUV 2012
7	00008.jpg	165	Mercedes-Benz S-Class Sedan 2012
8	00009.jpg	32	BMW X5 SUV 2007
9	00010.jpg	60	Chevrolet HHR SS 2010

In [ ]:
```
#Now the 'Names' column for each the training and test dataframes will be our y-values/ground truths.
```

In [ ]:
```
#Split the labels into lists of Makes, Models, Types, and Years
trainMakeAndModels =[]; trainCarTypes =[]; years =[]
for label in trainNames:
  makeAndModel, carType, year = label.rsplit(' ', 1)
  trainMakeAndModels.append(makeAndModel)
  trainCarTypes.append(carType)
  years.append(year)

makes =[]; models =[]
for i in trainMakeAndModels:
  make, model = i.split(' ', 1)
  makes.append(make)
  models.append(model)

print(makes)
print(models)
print(trainCarTypes)
print(years)
```

In [ ]:
```
#Check the types are accurately labeled.
checkTypes = np.array(trainCarTypes)
uniqTypes = np.unique(checkTypes)
print(uniqTypes)
```

```
['Abarth' 'Cab' 'Convertible' 'Coupe' 'GS' 'Hatchback' 'IPL' 'Minivan' 'R'
 'SRT-8' 'SRT8' 'SS' 'SUV' 'Sedan' 'SuperCab' 'Superleggera' 'Type-S'
 'Van' 'Wagon' 'XKR' 'Z06' 'ZR1']
```

In [ ]:
```
#Some of these do not include the type of the car (Sedan, coup, van, etc....) so we may need to get rid of those instance
```

In [ ]:
```
totalUnlabeledTypes = 0
for inst in uniqTypes:
  if inst=='Cab' or inst=='Convertible' or inst=='Coupe' or inst=='Hatchback' or inst=='Minivan' or inst=='SUV'\
  or inst=='Sedan' or inst=='Van' or inst=='Wagon':
    pass
  else:
    print('Instances of %s is %d' %(inst, trainCarTypes.count(inst)))
    totalUnlabeledTypes += trainCarTypes.count(inst)

print(' ')
print('Total samples with no given car type is %d.' % totalUnlabeledTypes)
print(len(trainCarTypes))
print('Percentage of unusable labels for type of car: %d percent.' % ((totalUnlabeledTypes/len(trainCarTypes))*100))
```

```
Instances of Abarth is 28
Instances of GS is 35
Instances of IPL is 34
Instances of R is 45
Instances of SRT-8 is 91
Instances of SRT8 is 39
Instances of SS is 119
Instances of SuperCab is 42
Instances of Superleggera is 36
Instances of Type-S is 42
Instances of XKR is 47
Instances of Z06 is 38
Instances of ZR1 is 47

Total samples with no given car type is 643.
8144
Percentage of unusable labels for type of car: 7 percent.
```

In [ ]:
```
#So unfortunately there are 896 instances that don't include the type of the car. This will be an issue if we split the l
#into 3 separate labels. We can either delete all of those samples or just not include the label for the type of car in
#general and leave the model to train only one make, model, and year.
#brayan: I would split into 2 separate labels, one for maker and the second for the remaining text of the class
#Robert: if we add Wagon we now have 7 percent.  But I agree that splitting the manufacturer(Make) and the rest will help
  # inconsistences in the names.
```

In [ ]:
```
#Robert: This is splitting the 'Names' column at the first space, to give us the Make/manufacturer.

df1=testLabels

df1['makes'] = df1['Names'].str.split(n=1).str[0]  # create 'First Name' column with first word
df1['model_type_year'] = df1['Names'].str.split(n=1).str[1:]
df1['model_type_year'] = df1['model_type_year'].str.join(' ')
df1.head()
```

Out[ ]:

	Image Label	Name List Index	Names	makes	model_type_year
0	00001.jpg	181	Suzuki Aerio Sedan 2007	Suzuki	Aerio Sedan 2007
1	00002.jpg	103	Ferrari 458 Italia Convertible 2012	Ferrari	458 Italia Convertible 2012
2	00003.jpg	145	Jeep Patriot SUV 2012	Jeep	Patriot SUV 2012
3	00004.jpg	187	Toyota Camry Sedan 2012	Toyota	Camry Sedan 2012
4	00005.jpg	185	Tesla Model S Sedan 2012	Tesla	Model S Sedan 2012

In [ ]:
```
#Let's look at an example of an image in the dataset.
import cv2
from google.colab.patches import cv2_imshow
randomImagePath = '/content/drive/MyDrive/Deep Learning/car_data/car_data/train/Acura TL Type-S 2008/00392.jpg'
randomImg = cv2.imread(randomImagePath)
cv2_imshow(randomImg)
```

In [ ]:
```
print(randomImg.shape)
```

```
(768, 1024, 3)
```

Data Augmentation

In [ ]:
```
def plotAugmentations(augImages):
    fig, axes = plt.subplots(1, 10, figsize=(20,20))
    axes = axes.flatten()
    for img, ax in zip(augImages, axes):
        ax.imshow(img)
    plt.tight_layout()
    plt.show()
```

Test on a single image first.

In [ ]:
```
#Test
image = np.expand_dims(randomImg,0)
plt.imshow(image[0])
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7fe5cc1d8550>
```

In [ ]:
```
dataAug = ImageDataGenerator(rotation_range=10, width_shift_range=0.1, height_shift_range=0.1, shear_range=0.15, zoom_rar
aug_iter = dataAug.flow(image)
aug_images = [next(aug_iter)[0].astype(np.uint8) for i in range(10)]
plotAugmentations(aug_images)
```

Now implement.

In [ ]:
```
#This section was moved to the DL_DataPreprocessing notebook.
import os
dataAug = ImageDataGenerator(rotation_range=10, width_shift_range=0.1, height_shift_range=0.1, shear_range=0.15, zoom_rar
trainFolder = '/content/drive/MyDrive/Deep Learning/car_data/car_data/train'
```

```
for folder in os.listdir(trainFolder):
  save_in = os.path.join(trainFolder, folder)
  for image in os.listdir(os.path.join(trainFolder, folder)):
    currentImg = os.path.join(save_in, image)
    readImg = cv2.imread(currentImg)
    im2aug = np.expand_dims(readImg,0)
    aug_iter = dataAug.flow(im2aug, save_to_dir=save_in, save_prefix='aug', save_format='jpg')
    #aug_images = [next(aug_iter)[0].astype(np.uint8) for i in range(5)]
```

Define Test, Train, and Validation Sets

Original Dataset

In [ ]:

```
#This is for the original dataset given as is with no preprocessing.
train = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/drive/MyDrive/Deep Learning/car_data/car_data/train', labels='inferred',
    image_size=(180, 180),
    batch_size=32,
)
validation = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/drive/MyDrive/Deep Learning/car_data/car_data/test', labels='inferred', validation_split=0.5,
    subset='training', shuffle=True,
    seed=100,
    image_size=(180, 180),
    batch_size=32,
)
test = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/drive/MyDrive/Deep Learning/car_data/car_data/test', labels='inferred', validation_split=0.5,
    subset='validation', shuffle=True,
    seed=100,
    image_size=(180, 180),
    batch_size=32,
)
```

```
Found 8088 files belonging to 196 classes.
Found 7985 files belonging to 196 classes.
Using 3993 files for training.
Found 7985 files belonging to 196 classes.
Using 3992 files for validation.
```

New Test/Train Split

In [ ]:
```
#Here are new subsets from altered dataset, moving half of the test images to the training set.
train2 = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/drive/MyDrive/Deep Learning/Keras/car_data/car_data/train', labels='inferred',
    image_size=(180, 180),
    batch_size=32,
)
validation2 = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/drive/MyDrive/Deep Learning/Keras/car_data/car_data/test', labels='inferred', validation_split=0.4,
    subset='training', shuffle=True,
    seed=100,
    image_size=(180, 180),
    batch_size=32,
)
test2 = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/drive/MyDrive/Deep Learning/Keras/car_data/car_data/test', labels='inferred', validation_split=0.4,
    subset='validation', shuffle=True,
    seed=100,
    image_size=(180, 180),
    batch_size=32,
)
```

```
Found 12090 files belonging to 196 classes.
Found 3983 files belonging to 196 classes.
Using 1594 files for training.
Found 3983 files belonging to 196 classes.
Using 1593 files for validation.
```

In [ ]:
```
#Here are new subsets from altered dataset, moving half of the test images to the training set.
#Try taking validation from training.
train3 = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/drive/MyDrive/Deep Learning/Keras/car_data/car_data/train', color_mode='grayscale', labels='inferred', vali
    subset='training', shuffle=True,
    seed=100,
    image_size=(250, 250),
    batch_size=32,
)
validation3 = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/drive/MyDrive/Deep Learning/Keras/car_data/car_data/train', color_mode='grayscale', labels='inferred', vali
    subset='validation', shuffle=True,
    seed=100,
    image_size=(250, 250),
    batch_size=32,
)
test3 = tf.keras.preprocessing.image_dataset_from_directory(
```

```
    '/content/drive/MyDrive/Deep Learning/Keras/car_data/car_data/test', color_mode='grayscale', labels='inferred',
    image_size=(250, 250), shuffle=False,
    batch_size=32,
)
```

```
Found 12090 files belonging to 196 classes.
Using 8463 files for training.
Found 12090 files belonging to 196 classes.
Using 3627 files for validation.
Found 3983 files belonging to 196 classes.
```

In [ ]:

```
#Make sure labels are correct.
plt.figure(figsize=(10, 20))
fig, ax = plt.subplots(3, 3, sharex=True, sharey=True)

for images, labels in validation3.take(1):
    for i in range(3):
        for j in range(3):
            ax[i][j].imshow(images[i*3+j].numpy().astype("uint8"))
            ax[i][j].set_title(validation3.class_names[labels[i*3+j]])
plt.show()
```

Dataset Separated By Car Make

In [ ]:

```
#For dataset split into folders by Make:
MakeTrain = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/drive/MyDrive/Deep Learning/SeparatedByMake/car_data/car_data/train', color_mode='grayscale', labels='infer
    image_size=(180, 180),
    batch_size=32,
)
MakeValidation = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/drive/MyDrive/Deep Learning/SeparatedByMake/car_data/car_data/test', color_mode='grayscale', labels='inferr
    subset='training', shuffle=True,
    seed=100,
    image_size=(180, 180),
    batch_size=32,
)
MakeTest = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/drive/MyDrive/Deep Learning/SeparatedByMake/car_data/car_data/test', color_mode='grayscale', labels='inferr
    subset='validation', shuffle=True,
    seed=100,
    image_size=(180, 180),
```

```
    batch_size=32,
)
```

```
Found 8144 files belonging to 48 classes.
Found 8027 files belonging to 48 classes.
Using 4014 files for training.
Found 8027 files belonging to 48 classes.
Using 4013 files for validation.
```

Dataset Separated By Vehicle Type

In [ ]:

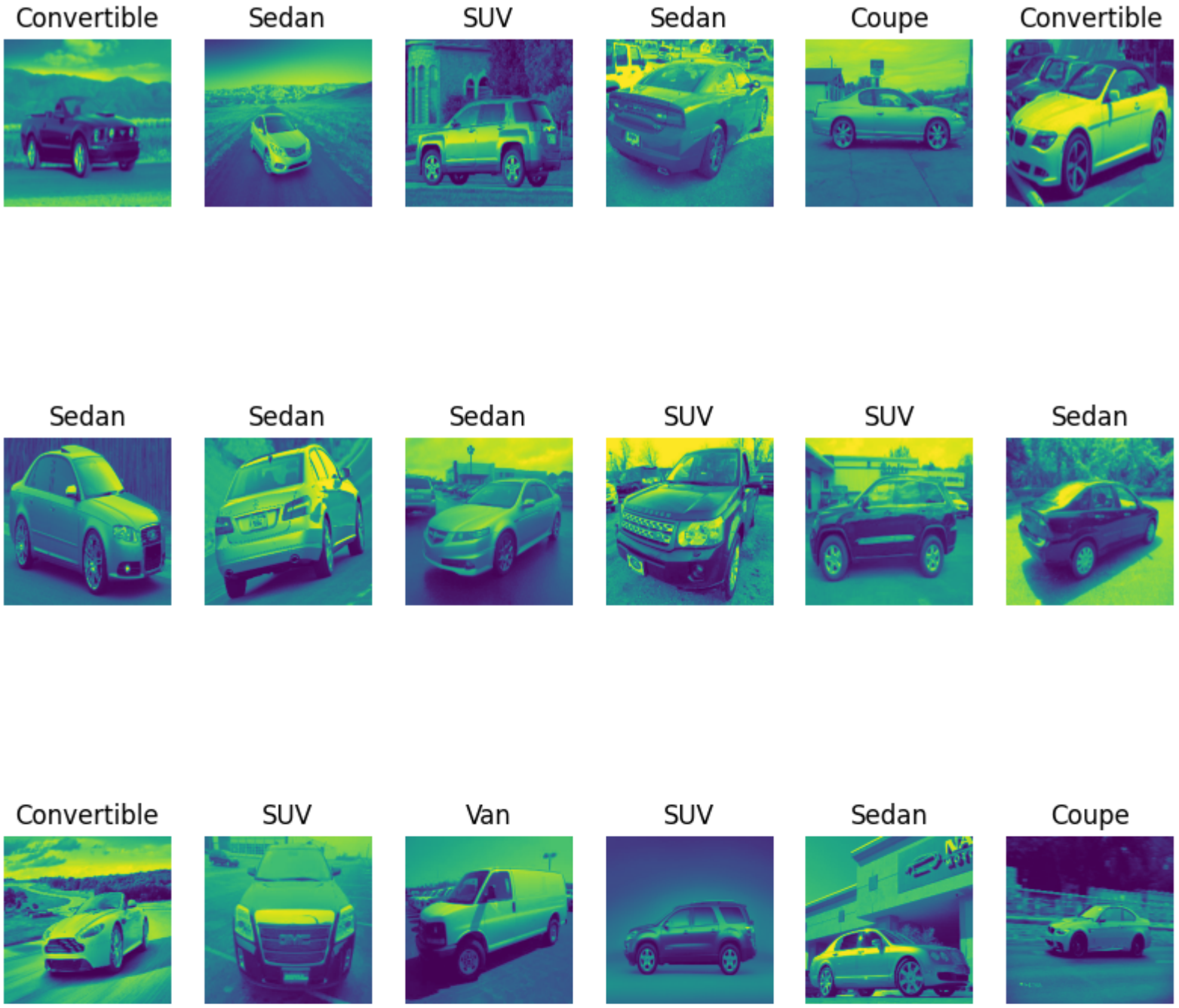
```
#For dataset split by vehicle type (Sedan, Coupe, Van, etc.)
TypeTrain = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/drive/MyDrive/Deep Learning/SeparatedByType/car_data/car_data/train', color_mode='grayscale', labels='infer
    subset='training', shuffle=True,
    seed=100,
    image_size=(224, 224),
    batch_size=32,
)
TypeVal = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/drive/MyDrive/Deep Learning/SeparatedByType/car_data/car_data/train', color_mode='grayscale', labels='infer
    subset='validation', shuffle=True,
    seed=100,
    image_size=(224, 224),
    batch_size=32,
)
TypeTest = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/drive/MyDrive/Deep Learning/SeparatedByType/car_data/car_data/test', color_mode='grayscale', labels='inferr
    image_size=(224, 224), shuffle=False,
    batch_size=32,
)
```

```
Found 11289 files belonging to 9 classes.
Using 9032 files for training.
Found 11289 files belonging to 9 classes.
Using 2257 files for validation.
Found 3747 files belonging to 9 classes.
```

In [ ]:

```
#Make sure our split kept our data labeled correctly.
plt.figure(figsize=(10, 20))
for images, labels in TypeVal.take(1):
    for i in range(32):
        ax = plt.subplot(6, 6, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
```

```
        plt.title(TypeVal.class_names[labels[i]])
        plt.axis("off")
```

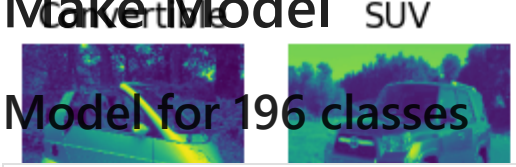

Convertible
Sedan
SUV
Sedan
Coupe
Convertible
Sedan
Sedan
Sedan
SUV
SUV
Sedan
Convertible
SUV
Van
SUV
Sedan
Coupe

Enlarged Dataset Including Augmented Images

In [ ]:
```
#For dataset with data augmentation:
augTrain = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/drive/MyDrive/Deep Learning/DataWithAugmentation/car_data/car_data/train', color_mode='grayscale', labels='
    image_size=(224, 224),
    batch_size=32,
)
augVal = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/drive/MyDrive/Deep Learning/DataWithAugmentation/car_data/car_data/validation', color_mode='grayscale', lab
    image_size=(224, 224), shuffle=False,
    batch_size=32,
)
augTest = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/drive/MyDrive/Deep Learning/DataWithAugmentation/car_data/car_data/test', color_mode='grayscale', labels='i
    shuffle=False,
    image_size=(224, 224),
    batch_size=32,
)
```

```
Found 32228 files belonging to 196 classes.
Found 3970 files belonging to 196 classes.
Found 3973 files belonging to 196 classes.
```

In [ ]:
```
#No longer need this because I am using grayscale.
#def norm(image,label):
#    image = tf.cast(image/255.,tf.float32)
#    return image,label

#train = train.map(norm)
#test = train.map(norm)
```



In [ ]:
```
data_augmentation = tf.keras.Sequential([
  layers.RandomFlip("horizontal"),
  layers.RandomRotation(0.2),
```

```
  layers.RandomTranslation(0.15,0.15)
])
```

In [ ]:

```
with tf.device(device_name):
  model = models.Sequential()
  model.add(data_augmentation)
  model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 1)))
  model.add(layers.BatchNormalization())
  model.add(layers.MaxPooling2D((2, 2)))
  model.add(layers.Dropout(0.25))

  model.add(layers.Conv2D(64, (3, 3), activation='relu'))
  model.add(layers.BatchNormalization())
  model.add(layers.MaxPooling2D((2, 2)))
  model.add(layers.Dropout(0.25))

  model.add(layers.Conv2D(128, (3, 3), activation='relu'))
  model.add(layers.BatchNormalization())
  model.add(layers.MaxPooling2D((2, 2)))
  model.add(layers.Dropout(0.25))

  model.add(layers.Flatten())
  model.add(layers.Dense(512, activation='relu'))
  model.add(layers.BatchNormalization())
  model.add(layers.Dropout(0.5))
  model.add(layers.Dense(196, activation='softmax'))
```

In [ ]:

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential (Sequential)     (None, 224, 224, 1)       0

 conv2d (Conv2D)             (None, 222, 222, 32)      320

 batch_normalization (BatchN  (None, 222, 222, 32)     128
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 111, 111, 32)     0
 )

 dropout (Dropout)           (None, 111, 111, 32)      0
```

```
conv2d_1 (Conv2D)           (None, 109, 109, 64)      18496

batch_normalization_1 (Batc  (None, 109, 109, 64)     256
hNormalization)

max_pooling2d_1 (MaxPooling  (None, 54, 54, 64)       0
2D)

dropout_1 (Dropout)         (None, 54, 54, 64)        0

conv2d_2 (Conv2D)           (None, 52, 52, 128)       73856

batch_normalization_2 (Batc  (None, 52, 52, 128)      512
hNormalization)

max_pooling2d_2 (MaxPooling  (None, 26, 26, 128)      0
2D)

dropout_2 (Dropout)         (None, 26, 26, 128)       0

flatten (Flatten)           (None, 86528)             0

dense (Dense)               (None, 512)               44302848

batch_normalization_3 (Batc  (None, 512)              2048
hNormalization)

dropout_3 (Dropout)         (None, 512)               0

dense_1 (Dense)             (None, 196)               100548

=================================================================
Total params: 44,499,012
Trainable params: 44,497,540
Non-trainable params: 1,472
_________________________________________________________________
```

In [ ]:

```
from keras.callbacks import ModelCheckpoint
callbacks = [
    ModelCheckpoint(filepath="/content/drive/MyDrive/Deep Learning/SavedModels",
                            monitor='val_accuracy', verbose=1,
                            save_best_only=True, mode='max')]
```

In [ ]:

```
with tf.device(device_name):
  model.compile(optimizer=tf.keras.optimizers.Adam(1e-2),loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

```
model.fit(train3, epochs=300, callbacks=callbacks, validation_data=validation3)
```

Model For Classifying Vehicle Type

In [ ]:

```
with tf.device(device_name):
  Type_model = models.Sequential()
  Type_model.add(data_augmentation)
  Type_model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 1)))
  Type_model.add(layers.BatchNormalization())
  Type_model.add(layers.MaxPooling2D((2, 2)))
  Type_model.add(layers.Dropout(0.25))

  Type_model.add(layers.Conv2D(64, (3, 3), activation='relu'))
  Type_model.add(layers.BatchNormalization())
  Type_model.add(layers.MaxPooling2D((2, 2)))
  Type_model.add(layers.Dropout(0.25))

  Type_model.add(layers.Conv2D(128, (3, 3), activation='relu'))
  Type_model.add(layers.BatchNormalization())
  Type_model.add(layers.MaxPooling2D((2, 2)))
  Type_model.add(layers.Dropout(0.25))

  Type_model.add(layers.Flatten())
  Type_model.add(layers.Dense(512, activation='relu'))
  Type_model.add(layers.BatchNormalization())
  Type_model.add(layers.Dropout(0.5))
  Type_model.add(layers.Dense(9, activation='softmax'))
```

In [ ]:

```
Type_model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential (Sequential)     (None, 224, 224, 1)       0

 conv2d (Conv2D)             (None, 222, 222, 32)      320

 batch_normalization (BatchN  (None, 222, 222, 32)     128
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 111, 111, 32)     0
```

```
)

dropout (Dropout)           (None, 111, 111, 32)      0

conv2d_1 (Conv2D)           (None, 109, 109, 64)      18496

batch_normalization_1 (Batc  (None, 109, 109, 64)     256
hNormalization)

max_pooling2d_1 (MaxPooling  (None, 54, 54, 64)       0
2D)

dropout_1 (Dropout)         (None, 54, 54, 64)        0

conv2d_2 (Conv2D)           (None, 52, 52, 128)       73856

batch_normalization_2 (Batc  (None, 52, 52, 128)      512
hNormalization)

max_pooling2d_2 (MaxPooling  (None, 26, 26, 128)      0
2D)

dropout_2 (Dropout)         (None, 26, 26, 128)       0

flatten (Flatten)           (None, 86528)             0

dense (Dense)               (None, 512)               44302848

batch_normalization_3 (Batc  (None, 512)              2048
hNormalization)

dropout_3 (Dropout)         (None, 512)               0

dense_1 (Dense)             (None, 9)                 4617

=================================================================
Total params: 44,403,081
Trainable params: 44,401,609
Non-trainable params: 1,472
_________________________________________________________________
```

In [ ]:

```
from keras.callbacks import ModelCheckpoint
Type_callbacks = [
    ModelCheckpoint(filepath="/content/drive/MyDrive/Deep Learning/SavedModels/BestModelForType",
                            monitor='val_accuracy', verbose=1,
                            save_best_only=True, mode='max')]
```

In [ ]:

```
#Train Model
with tf.device(device_name):
  Type_model.compile(optimizer=tf.keras.optimizers.Adam(3e-3),loss='sparse_categorical_crossentropy', metrics=['accuracy'

Type_hist = Type_model.fit(TypeTrain, epochs=500, callbacks=Type_callbacks, validation_data=TypeVal)
```

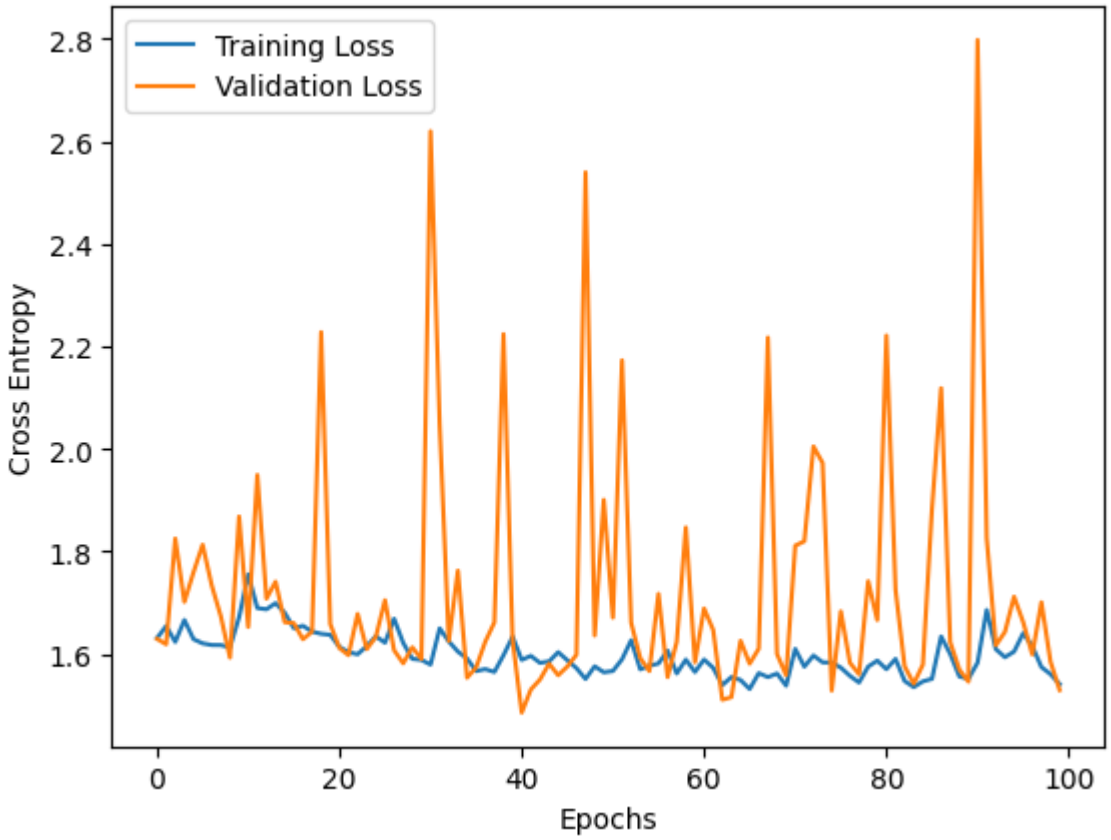
```
Epoch 1/500
283/283 [==============================] - ETA: 0s - loss: 2.6054 - accuracy: 0.1905
Epoch 1: val_accuracy improved from -inf to 0.19716, saving model to /content/drive/MyDrive/Deep Learning/SavedModels/Bes
tModelForType
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_c
onvolution_op, _update_step_xla while saving (showing 4 of 4). These functions will not be directly callable after loadin
g.
283/283 [==============================] - 61s 184ms/step - loss: 2.6054 - accuracy: 0.1905 - val_loss: 2.2848 - val_accu
racy: 0.1972
Epoch 2/500
282/283 [============================>.] - ETA: 0s - loss: 2.1947 - accuracy: 0.2118
Epoch 2: val_accuracy improved from 0.19716 to 0.20160, saving model to /content/drive/MyDrive/Deep Learning/SavedModels/
BestModelForType
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_c
onvolution_op, _update_step_xla while saving (showing 4 of 4). These functions will not be directly callable after loadin
g.
283/283 [==============================] - 54s 187ms/step - loss: 2.1945 - accuracy: 0.2117 - val_loss: 2.9743 - val_accu
racy: 0.2016
Epoch 3/500
282/283 [============================>.] - ETA: 0s - loss: 2.1272 - accuracy: 0.2215
Epoch 3: val_accuracy improved from 0.20160 to 0.22641, saving model to /content/drive/MyDrive/Deep Learning/SavedModels/
BestModelForType
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_c
onvolution_op, _update_step_xla while saving (showing 4 of 4). These functions will not be directly callable after loadin
g.
283/283 [==============================] - 57s 197ms/step - loss: 2.1273 - accuracy: 0.2217 - val_loss: 2.2767 - val_accu
racy: 0.2264
Epoch 4/500
282/283 [============================>.] - ETA: 0s - loss: 2.1008 - accuracy: 0.2246
Epoch 4: val_accuracy did not improve from 0.22641
283/283 [==============================] - 46s 161ms/step - loss: 2.1004 - accuracy: 0.2246 - val_loss: 3.0873 - val_accu
racy: 0.1684
Epoch 5/500
282/283 [============================>.] - ETA: 0s - loss: 2.0551 - accuracy: 0.2307
Epoch 5: val_accuracy did not improve from 0.22641
283/283 [==============================] - 48s 167ms/step - loss: 2.0554 - accuracy: 0.2306 - val_loss: 3.1377 - val_accu
racy: 0.1808
Epoch 6/500
282/283 [============================>.] - ETA: 0s - loss: 2.0322 - accuracy: 0.2376
Epoch 6: val_accuracy did not improve from 0.22641
```

```
racy: 0.4922
Epoch 487/500
282/283 [============================>.] - ETA: 0s - loss: 1.3322 - accuracy: 0.5040
Epoch 487: val_accuracy did not improve from 0.50465
283/283 [==============================] - 50s 175ms/step - loss: 1.3323 - accuracy: 0.5041 - val_loss: 1.6141 - val_accu
racy: 0.4599
Epoch 488/500
283/283 [==============================] - ETA: 0s - loss: 1.3214 - accuracy: 0.5093
Epoch 488: val_accuracy did not improve from 0.50465
283/283 [==============================] - 47s 162ms/step - loss: 1.3214 - accuracy: 0.5093 - val_loss: 1.5407 - val_accu
racy: 0.4790
Epoch 489/500
282/283 [============================>.] - ETA: 0s - loss: 1.4416 - accuracy: 0.4746
Epoch 489: val_accuracy did not improve from 0.50465
283/283 [==============================] - 47s 165ms/step - loss: 1.4412 - accuracy: 0.4749 - val_loss: 1.3800 - val_accu
racy: 0.4931
Epoch 490/500
282/283 [============================>.] - ETA: 0s - loss: 1.3473 - accuracy: 0.4969
Epoch 490: val_accuracy improved from 0.50465 to 0.51440, saving model to /content/drive/MyDrive/Deep Learning/SavedModel
s/BestModelForType
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_c
onvolution_op, _update_step_xla while saving (showing 4 of 4). These functions will not be directly callable after loadin
g.
283/283 [==============================] - 55s 191ms/step - loss: 1.3477 - accuracy: 0.4968 - val_loss: 1.5100 - val_accu
racy: 0.5144
Epoch 491/500
282/283 [============================>.] - ETA: 0s - loss: 1.3362 - accuracy: 0.5020
Epoch 491: val_accuracy did not improve from 0.51440
283/283 [==============================] - 51s 178ms/step - loss: 1.3360 - accuracy: 0.5020 - val_loss: 1.7380 - val_accu
racy: 0.4922
Epoch 492/500
282/283 [============================>.] - ETA: 0s - loss: 1.3371 - accuracy: 0.5076
Epoch 492: val_accuracy did not improve from 0.51440
283/283 [==============================] - 47s 163ms/step - loss: 1.3365 - accuracy: 0.5079 - val_loss: 1.7576 - val_accu
racy: 0.4971
Epoch 493/500
282/283 [============================>.] - ETA: 0s - loss: 1.3105 - accuracy: 0.5126
Epoch 493: val_accuracy did not improve from 0.51440
283/283 [==============================] - 48s 168ms/step - loss: 1.3109 - accuracy: 0.5125 - val_loss: 1.3867 - val_accu
racy: 0.5091
Epoch 494/500
282/283 [============================>.] - ETA: 0s - loss: 1.3039 - accuracy: 0.5176
Epoch 494: val_accuracy did not improve from 0.51440
283/283 [==============================] - 49s 171ms/step - loss: 1.3041 - accuracy: 0.5176 - val_loss: 1.5460 - val_accu
racy: 0.4922
Epoch 495/500
282/283 [============================>.] - ETA: 0s - loss: 1.3271 - accuracy: 0.5092
Epoch 495: val_accuracy did not improve from 0.51440
283/283 [==============================] - 47s 164ms/step - loss: 1.3271 - accuracy: 0.5092 - val_loss: 1.6378 - val_accu
```

```
racy: 0.5117
Epoch 496/500
282/283 [============================>.] - ETA: 0s - loss: 1.3128 - accuracy: 0.5110
Epoch 496: val_accuracy did not improve from 0.51440
283/283 [==============================] - 51s 176ms/step - loss: 1.3125 - accuracy: 0.5111 - val_loss: 1.5520 - val_accu
racy: 0.4962
Epoch 497/500
282/283 [============================>.] - ETA: 0s - loss: 1.3338 - accuracy: 0.5068
Epoch 497: val_accuracy did not improve from 0.51440
283/283 [==============================] - 48s 168ms/step - loss: 1.3338 - accuracy: 0.5068 - val_loss: 2.7064 - val_accu
racy: 0.5069
Epoch 498/500
282/283 [============================>.] - ETA: 0s - loss: 1.3551 - accuracy: 0.5030
Epoch 498: val_accuracy did not improve from 0.51440
283/283 [==============================] - 49s 168ms/step - loss: 1.3558 - accuracy: 0.5028 - val_loss: 1.9271 - val_accu
racy: 0.3026
Epoch 499/500
282/283 [============================>.] - ETA: 0s - loss: 1.4446 - accuracy: 0.4651
Epoch 499: val_accuracy did not improve from 0.51440
283/283 [==============================] - 48s 164ms/step - loss: 1.4446 - accuracy: 0.4649 - val_loss: 1.4460 - val_accu
racy: 0.4887
Epoch 500/500
282/283 [============================>.] - ETA: 0s - loss: 1.3578 - accuracy: 0.4965
Epoch 500: val_accuracy did not improve from 0.51440
283/283 [==============================] - 48s 168ms/step - loss: 1.3581 - accuracy: 0.4963 - val_loss: 1.4228 - val_accu
racy: 0.4945
```

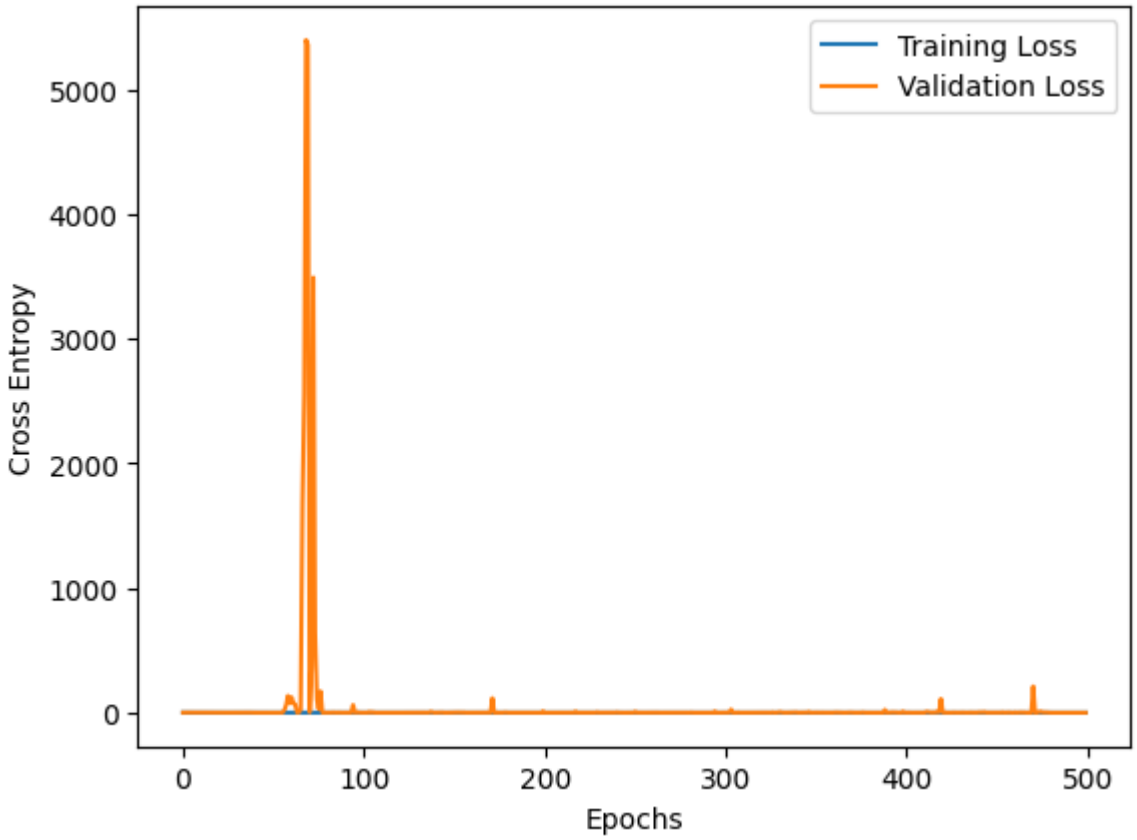
In [ ]:

```
#For the first 100 epochs.
plt.plot(Type_hist.history['loss'], label='Training Loss')
plt.plot(Type_hist.history['val_loss'], label='Validation Loss')
plt.legend()
plt.xlabel('Epochs')
plt.ylabel('Cross Entropy')
plt.savefig('model_training_history')
plt.show()
```



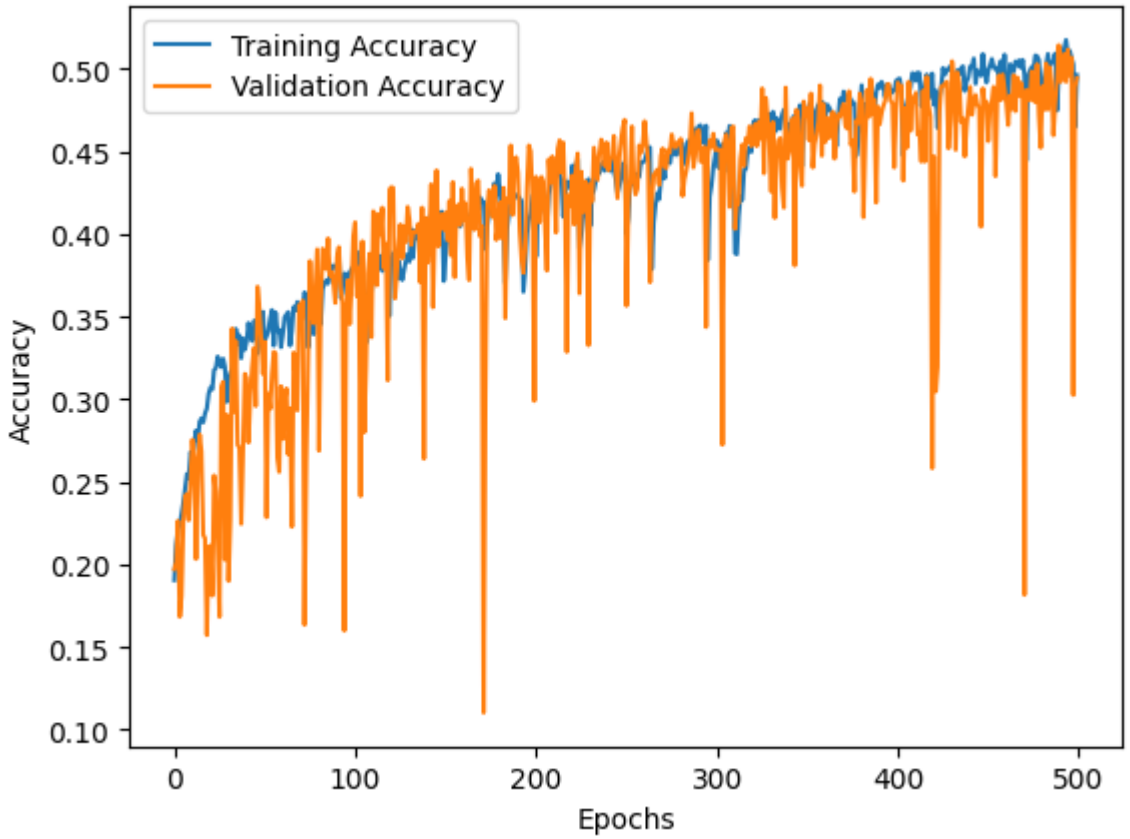
In [ ]:

```
#For 500 more epochs.
plt.plot(Type_hist.history['loss'], label='Training Loss')
plt.plot(Type_hist.history['val_loss'], label='Validation Loss')
plt.legend()
plt.xlabel('Epochs')
plt.ylabel('Cross Entropy')
plt.savefig('model_training_history')
plt.show()
```



In [ ]:

```
#Plot for accuracy over 500 epochs.
plt.plot(Type_hist.history['accuracy'], label='Training Accuracy')
plt.plot(Type_hist.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.savefig('model_training_history')
plt.show()
```



In [ ]:
```
Type_model.load_weights('/content/drive/MyDrive/Deep Learning/SavedModels/BestModelForType')
```

Out[ ]:
```
<tensorflow.python.checkpoint.checkpoint.CheckpointLoadStatus at 0x7f14b96c2410>
```

In [ ]:
```
#Evaluate the model on the test set.
result = Type_model.evaluate(TypeTest)
```

```
118/118 [==============================] - 14s 116ms/step - loss: 1.4179 - accuracy: 0.5191
```

In [ ]:
```
#Make predictions from test set.
Type_predictions = Type_model.predict(TypeTest)
```

```
118/118 [==============================] - 14s 117ms/step
```

In [ ]:
```
Type_predictions.shape
```

Out[ ]:
```
(3747, 9)
```

In [ ]:
```
print(Type_predictions[0])
```

```
[0.07558715 0.02483506 0.01093115 0.03120333 0.11110371 0.34519753
 0.32071775 0.00171897 0.07870525]
```

In [ ]:
```
#Run this if you would like to see the ground truths of the test set.
file_paths = TypeTest.file_paths
print(file_paths)
groundTruths = []
testImages = []
for path in file_paths:
  imgFile, groundTruth, img = path.rsplit('/', 2)
  groundTruths.append(groundTruth)
  testImages.append(img)

print(groundTruths)
```

In [ ]:
```
#Index the ground truths to compare with the highest index from the predictions.
testTruths = []
for i in range(len(groundTruths)):
  testTruths.append(TypeTest.class_names.index(groundTruths[i]))
print(testTruths)
```

In [ ]:
```
preds = np.argmax (Type_predictions, axis = 1)
tests = np.array(testTruths)
print(preds)
print(tests)
```

```
[5 5 4 ... 6 6 6]
[0 0 0 ... 8 8 8]
```

In [ ]:
```
predicted_labels = []
for i in range(len(preds)):
  predicted_labels.append(TypeTest.class_names[preds[i]])
#Uncomment if you'd like to see predictions vs. ground truths.
```

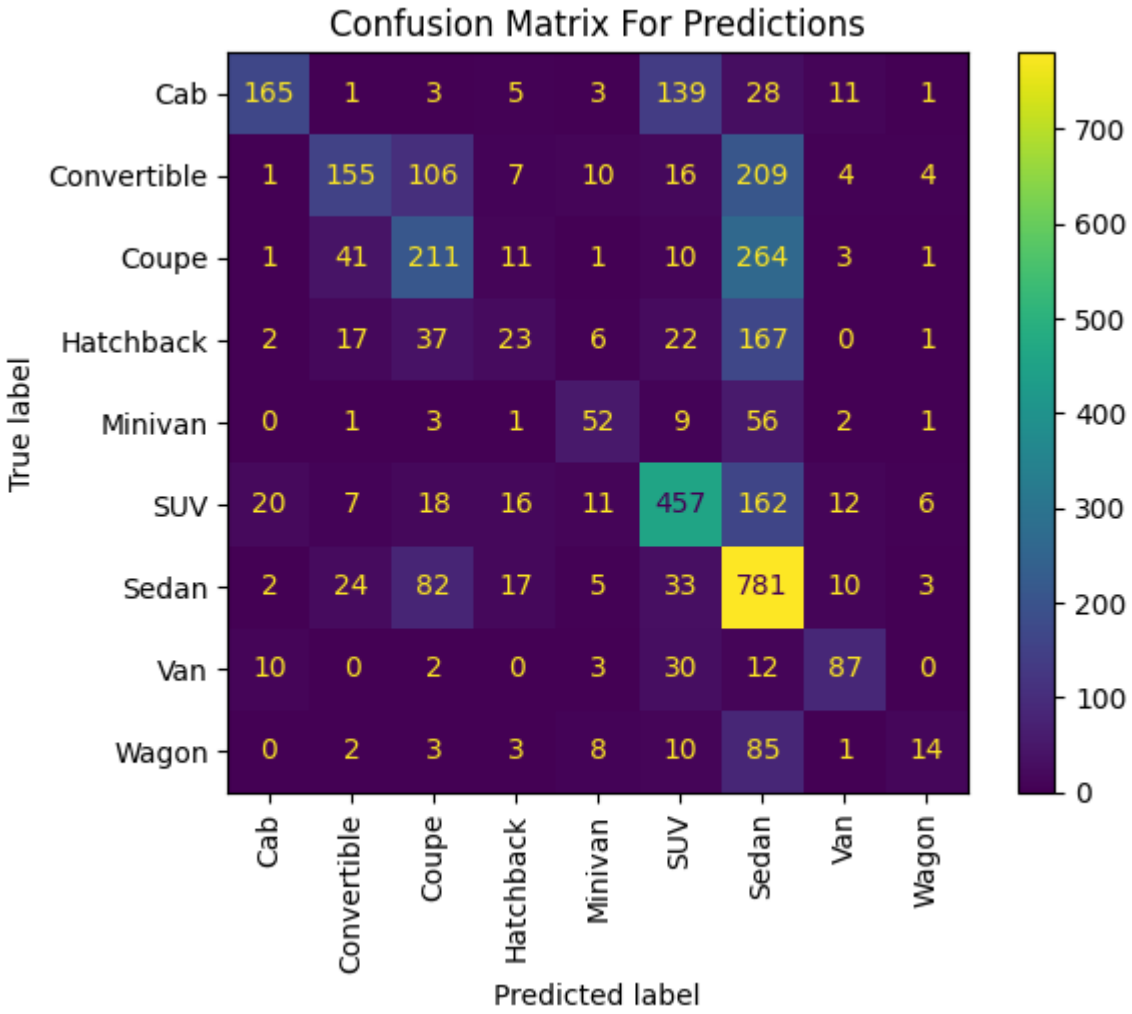
```
#print(predicted_labels)
#print(groundTruths)
```

In [ ]:

```
from sklearn.metrics import confusion_matrix
from sklearn import metrics
confusionMat = confusion_matrix(tests, preds)
dispConfMat = metrics.ConfusionMatrixDisplay(confusion_matrix = confusionMat)
dispConfMat.plot()
plt.title('Confusion Matrix For Predictions')
plt.xticks(np.arange(9), TypeTest.class_names, rotation=90)
plt.yticks(np.arange(9), TypeTest.class_names)
```

Out[ ]:

```
([<matplotlib.axis.YTick at 0x7f13402ee320>,
  <matplotlib.axis.YTick at 0x7f13402eea70>,
  <matplotlib.axis.YTick at 0x7f13402ecee0>,
  <matplotlib.axis.YTick at 0x7f1340711de0>,
  <matplotlib.axis.YTick at 0x7f1340710c70>,
  <matplotlib.axis.YTick at 0x7f13407126e0>,
  <matplotlib.axis.YTick at 0x7f1340713190>,
  <matplotlib.axis.YTick at 0x7f1340713c40>,
  <matplotlib.axis.YTick at 0x7f1340712dd0>],
 [Text(0, 0, 'Cab'),
  Text(0, 1, 'Convertible'),
  Text(0, 2, 'Coupe'),
  Text(0, 3, 'Hatchback'),
  Text(0, 4, 'Minivan'),
  Text(0, 5, 'SUV'),
  Text(0, 6, 'Sedan'),
  Text(0, 7, 'Van'),
  Text(0, 8, 'Wagon')])
```



In [ ]:

```
from sklearn.metrics import classification_report
print(classification_report(tests, preds))
```

```
              precision    recall  f1-score   support

           0       0.82      0.46      0.59       356
           1       0.62      0.30      0.41       512
           2       0.45      0.39      0.42       543
           3       0.28      0.08      0.13       275
           4       0.53      0.42      0.46       125
           5       0.63      0.64      0.64       709
           6       0.44      0.82      0.57       957
```

```
           7       0.67      0.60      0.64       144
           8       0.45      0.11      0.18       126

    accuracy                           0.52      3747
   macro avg       0.54      0.43      0.45      3747
weighted avg       0.54      0.52      0.50      3747
```

In [ ]:
```
Type_model.save('/content/drive/MyDrive/Deep Learning/SavedModels/VehicleTypeModel51.hdf5')
Type_model.save('/content/drive/MyDrive/Deep Learning/SavedModels/VehicleTypeModel51.h5')
```

We can see the amount of oscillations during training. So to fix this we can try to decrease out learning rate.

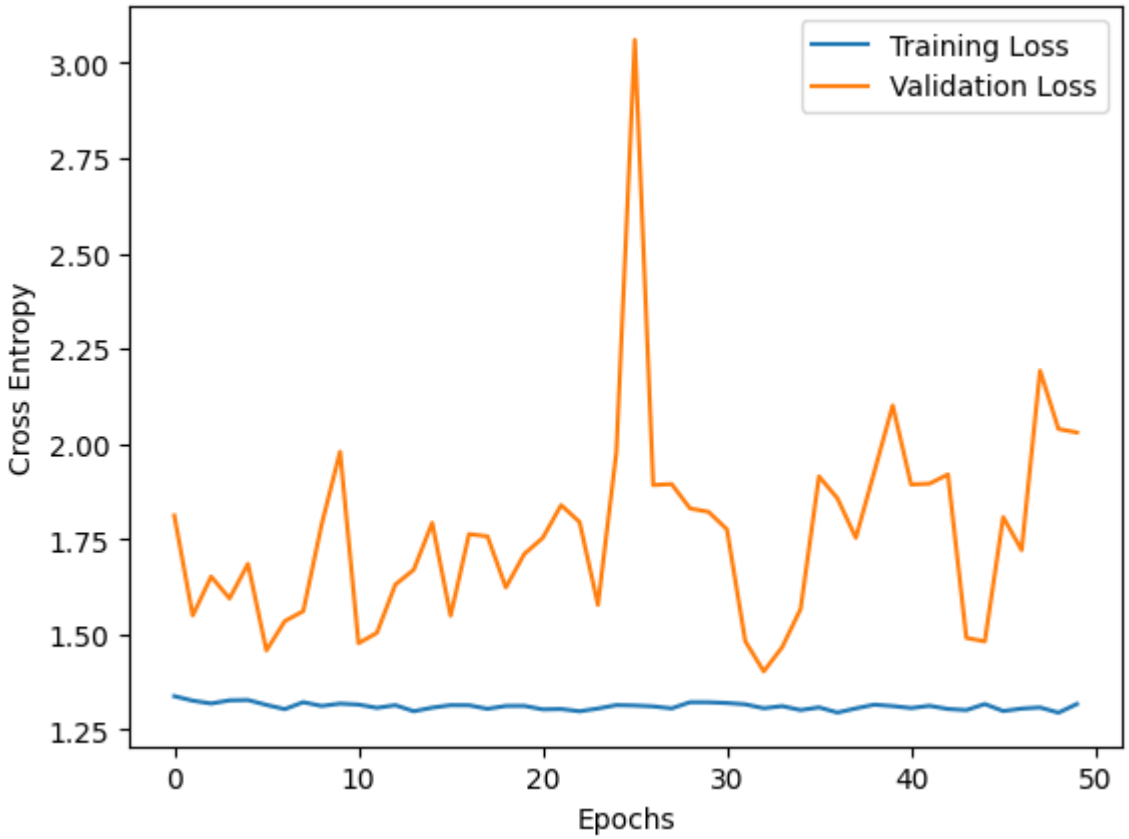
In [ ]:
```
with tf.device(device_name):
  Type_model.compile(optimizer=tf.keras.optimizers.Adam(1e-5),loss='sparse_categorical_crossentropy', metrics=['accuracy'

Type_hist = Type_model.fit(TypeTrain, epochs=50, callbacks=Type_callbacks, validation_data=TypeVal)
```

```
Epoch 1/50
282/283 [============================>.] - ETA: 0s - loss: 1.3375 - accuracy: 0.4977
Epoch 1: val_accuracy did not improve from 0.51440
283/283 [==============================] - 56s 184ms/step - loss: 1.3373 - accuracy: 0.4976 - val_loss: 1.8116 - val_accu
racy: 0.5051
Epoch 2/50
282/283 [============================>.] - ETA: 0s - loss: 1.3255 - accuracy: 0.5099
Epoch 2: val_accuracy did not improve from 0.51440
283/283 [==============================] - 47s 163ms/step - loss: 1.3255 - accuracy: 0.5097 - val_loss: 1.5491 - val_accu
racy: 0.5033
Epoch 3/50
282/283 [============================>.] - ETA: 0s - loss: 1.3184 - accuracy: 0.5102
Epoch 3: val_accuracy did not improve from 0.51440
283/283 [==============================] - 53s 185ms/step - loss: 1.3181 - accuracy: 0.5103 - val_loss: 1.6513 - val_accu
racy: 0.5033
Epoch 4/50
282/283 [============================>.] - ETA: 0s - loss: 1.3260 - accuracy: 0.5076
Epoch 4: val_accuracy did not improve from 0.51440
283/283 [==============================] - 46s 160ms/step - loss: 1.3263 - accuracy: 0.5074 - val_loss: 1.5938 - val_accu
racy: 0.5011
Epoch 5/50
282/283 [============================>.] - ETA: 0s - loss: 1.3275 - accuracy: 0.5096
Epoch 5: val_accuracy did not improve from 0.51440
283/283 [==============================] - 50s 175ms/step - loss: 1.3272 - accuracy: 0.5095 - val_loss: 1.6848 - val_accu
racy: 0.5007
Epoch 6/50
282/283 [============================>.] - ETA: 0s - loss: 1.3143 - accuracy: 0.5091
Epoch 6: val_accuracy did not improve from 0.51440
```

```
racy: 0.4984
Epoch 46/50
282/283 [============================>.] - ETA: 0s - loss: 1.2980 - accuracy: 0.5193
Epoch 46: val_accuracy did not improve from 0.51440
283/283 [==============================] - 52s 181ms/step - loss: 1.2983 - accuracy: 0.5193 - val_loss: 1.8074 - val_accu
racy: 0.4993
Epoch 47/50
282/283 [============================>.] - ETA: 0s - loss: 1.3048 - accuracy: 0.5201
Epoch 47: val_accuracy did not improve from 0.51440
283/283 [==============================] - 49s 171ms/step - loss: 1.3047 - accuracy: 0.5202 - val_loss: 1.7207 - val_accu
racy: 0.4989
Epoch 48/50
282/283 [============================>.] - ETA: 0s - loss: 1.3077 - accuracy: 0.5116
Epoch 48: val_accuracy did not improve from 0.51440
283/283 [==============================] - 50s 172ms/step - loss: 1.3077 - accuracy: 0.5116 - val_loss: 2.1918 - val_accu
racy: 0.4989
Epoch 49/50
282/283 [============================>.] - ETA: 0s - loss: 1.2944 - accuracy: 0.5233
Epoch 49: val_accuracy did not improve from 0.51440
283/283 [==============================] - 46s 161ms/step - loss: 1.2940 - accuracy: 0.5234 - val_loss: 2.0388 - val_accu
racy: 0.4998
Epoch 50/50
282/283 [============================>.] - ETA: 0s - loss: 1.3164 - accuracy: 0.5156
Epoch 50: val_accuracy did not improve from 0.51440
283/283 [==============================] - 48s 165ms/step - loss: 1.3167 - accuracy: 0.5154 - val_loss: 2.0296 - val_accu
racy: 0.4998
```

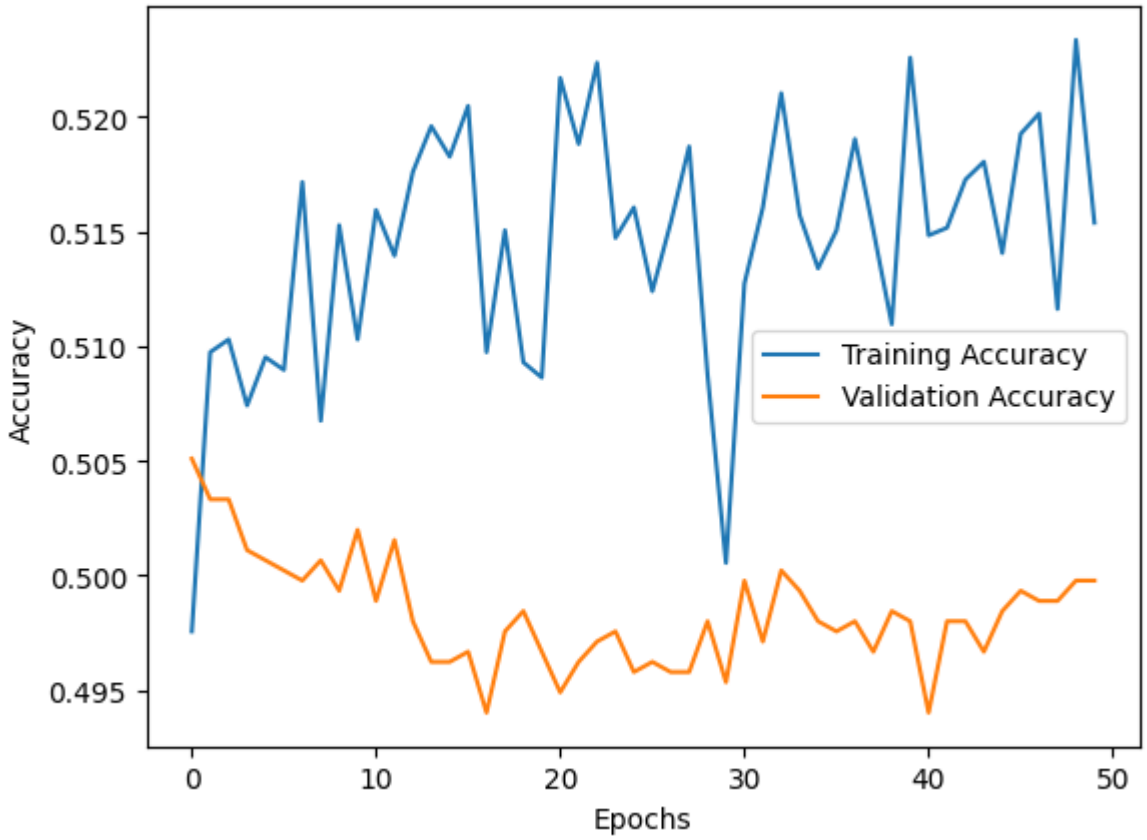
In [ ]:

```
#Loss for 50 epochs with learning rate = 1e-5.
plt.plot(Type_hist.history['loss'], label='Training Loss')
plt.plot(Type_hist.history['val_loss'], label='Validation Loss')
plt.legend()
plt.xlabel('Epochs')
plt.ylabel('Cross Entropy')
plt.savefig('model_training_history')
plt.show()
```



In [ ]:

```
##Accuracy for 50 epochs with learning rate = 1e-5.
plt.plot(Type_hist.history['accuracy'], label='Training Accuracy')
plt.plot(Type_hist.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.savefig('model_training_history')
plt.show()
```



In [ ]:
```
#No longer huge oscillates, but is now stuck at 49/50%. Try learning rate = 1e-4
```

In [ ]:
```
with tf.device(device_name):
  Type_model.compile(optimizer=tf.keras.optimizers.Adam(1e-4),loss='sparse_categorical_crossentropy', metrics=['accuracy'

Type_hist = Type_model.fit(TypeTrain, epochs=20, callbacks=Type_callbacks, validation_data=TypeVal)
```

```
Epoch 1/20
282/283 [============================>.] - ETA: 0s - loss: 1.2917 - accuracy: 0.5199
Epoch 1: val_accuracy did not improve from 0.51440
283/283 [==============================] - 53s 167ms/step - loss: 1.2912 - accuracy: 0.5202 - val_loss: 1.9875 - val_accu
racy: 0.4953
Epoch 2/20
282/283 [============================>.] - ETA: 0s - loss: 1.2972 - accuracy: 0.5129
Epoch 2: val_accuracy did not improve from 0.51440
283/283 [==============================] - 51s 178ms/step - loss: 1.2969 - accuracy: 0.5130 - val_loss: 1.5484 - val_accu
```

```
racy: 0.5042
Epoch 3/20
282/283 [============================>.] - ETA: 0s - loss: 1.2878 - accuracy: 0.5276
Epoch 3: val_accuracy did not improve from 0.51440
283/283 [==============================] - 48s 166ms/step - loss: 1.2877 - accuracy: 0.5276 - val_loss: 1.7758 - val_accu
racy: 0.5002
Epoch 4/20
282/283 [============================>.] - ETA: 0s - loss: 1.2912 - accuracy: 0.5219
Epoch 4: val_accuracy did not improve from 0.51440
283/283 [==============================] - 48s 166ms/step - loss: 1.2910 - accuracy: 0.5219 - val_loss: 1.6431 - val_accu
racy: 0.5016
Epoch 5/20
282/283 [============================>.] - ETA: 0s - loss: 1.2895 - accuracy: 0.5252
Epoch 5: val_accuracy did not improve from 0.51440
283/283 [==============================] - 48s 167ms/step - loss: 1.2900 - accuracy: 0.5250 - val_loss: 1.7823 - val_accu
racy: 0.5033
Epoch 6/20
282/283 [============================>.] - ETA: 0s - loss: 1.2989 - accuracy: 0.5207
Epoch 6: val_accuracy did not improve from 0.51440
283/283 [==============================] - 48s 164ms/step - loss: 1.2990 - accuracy: 0.5206 - val_loss: 1.7402 - val_accu
racy: 0.5020
Epoch 7/20
282/283 [============================>.] - ETA: 0s - loss: 1.3077 - accuracy: 0.5147
Epoch 7: val_accuracy did not improve from 0.51440
283/283 [==============================] - 47s 162ms/step - loss: 1.3075 - accuracy: 0.5147 - val_loss: 1.5168 - val_accu
racy: 0.5011
Epoch 8/20
282/283 [============================>.] - ETA: 0s - loss: 1.3058 - accuracy: 0.5134
Epoch 8: val_accuracy did not improve from 0.51440
283/283 [==============================] - 48s 168ms/step - loss: 1.3059 - accuracy: 0.5135 - val_loss: 1.6221 - val_accu
racy: 0.5042
Epoch 9/20
282/283 [============================>.] - ETA: 0s - loss: 1.2792 - accuracy: 0.5250
Epoch 9: val_accuracy did not improve from 0.51440
283/283 [==============================] - 48s 166ms/step - loss: 1.2789 - accuracy: 0.5251 - val_loss: 1.6673 - val_accu
racy: 0.5055
Epoch 10/20
282/283 [============================>.] - ETA: 0s - loss: 1.2919 - accuracy: 0.5217
Epoch 10: val_accuracy did not improve from 0.51440
283/283 [==============================] - 46s 160ms/step - loss: 1.2919 - accuracy: 0.5217 - val_loss: 2.1324 - val_accu
racy: 0.5051
Epoch 11/20
282/283 [============================>.] - ETA: 0s - loss: 1.2986 - accuracy: 0.5181
Epoch 11: val_accuracy did not improve from 0.51440
283/283 [==============================] - 47s 162ms/step - loss: 1.2987 - accuracy: 0.5179 - val_loss: 1.4685 - val_accu
racy: 0.5047
Epoch 12/20
282/283 [============================>.] - ETA: 0s - loss: 1.2791 - accuracy: 0.5318
Epoch 12: val_accuracy did not improve from 0.51440
```

```
283/283 [==============================] - 48s 168ms/step - loss: 1.2795 - accuracy: 0.5317 - val_loss: 1.6121 - val_accu
racy: 0.5100
Epoch 13/20
282/283 [============================>.] - ETA: 0s - loss: 1.2767 - accuracy: 0.5284
Epoch 13: val_accuracy did not improve from 0.51440
283/283 [==============================] - 46s 161ms/step - loss: 1.2773 - accuracy: 0.5282 - val_loss: 2.2916 - val_accu
racy: 0.5060
Epoch 14/20
282/283 [============================>.] - ETA: 0s - loss: 1.2868 - accuracy: 0.5291
Epoch 14: val_accuracy did not improve from 0.51440
283/283 [==============================] - 48s 168ms/step - loss: 1.2873 - accuracy: 0.5291 - val_loss: 1.8771 - val_accu
racy: 0.5033
Epoch 15/20
282/283 [============================>.] - ETA: 0s - loss: 1.2907 - accuracy: 0.5203
Epoch 15: val_accuracy did not improve from 0.51440
283/283 [==============================] - 50s 176ms/step - loss: 1.2906 - accuracy: 0.5203 - val_loss: 1.7528 - val_accu
racy: 0.5055
Epoch 16/20
282/283 [============================>.] - ETA: 0s - loss: 1.2951 - accuracy: 0.5232
Epoch 16: val_accuracy did not improve from 0.51440
283/283 [==============================] - 46s 161ms/step - loss: 1.2951 - accuracy: 0.5229 - val_loss: 1.4771 - val_accu
racy: 0.5078
Epoch 17/20
282/283 [============================>.] - ETA: 0s - loss: 1.3020 - accuracy: 0.5276
Epoch 17: val_accuracy did not improve from 0.51440
283/283 [==============================] - 50s 175ms/step - loss: 1.3021 - accuracy: 0.5276 - val_loss: 1.6444 - val_accu
racy: 0.5055
Epoch 18/20
282/283 [============================>.] - ETA: 0s - loss: 1.2875 - accuracy: 0.5202
Epoch 18: val_accuracy did not improve from 0.51440
283/283 [==============================] - 48s 167ms/step - loss: 1.2874 - accuracy: 0.5204 - val_loss: 1.4486 - val_accu
racy: 0.5078
Epoch 19/20
282/283 [============================>.] - ETA: 0s - loss: 1.2862 - accuracy: 0.5266
Epoch 19: val_accuracy did not improve from 0.51440
283/283 [==============================] - 49s 169ms/step - loss: 1.2860 - accuracy: 0.5266 - val_loss: 1.4684 - val_accu
racy: 0.5082
Epoch 20/20
282/283 [============================>.] - ETA: 0s - loss: 1.2724 - accuracy: 0.5286
Epoch 20: val_accuracy did not improve from 0.51440
283/283 [==============================] - 47s 162ms/step - loss: 1.2725 - accuracy: 0.5286 - val_loss: 1.5130 - val_accu
racy: 0.5104
```

In [ ]:
```
#Still no increase in accuracy.
```

Simple Model

In [ ]:
```
with tf.device(device_name):
  simpleModel = models.Sequential()
  simpleModel.add(data_augmentation)
  simpleModel.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(224,224,1)))
  simpleModel.add(layers.MaxPooling2D((2,2)))
  simpleModel.add(layers.Conv2D(64, (3, 3), activation='relu'))
  simpleModel.add(layers.MaxPooling2D((2,2)))
  simpleModel.add(layers.Conv2D(64, (3, 3), activation='relu'))
  simpleModel.add(layers.Flatten())
  simpleModel.add(layers.Dense(64, activation='relu'))
  simpleModel.add(layers.Dropout(0.5))
  simpleModel.add(layers.Dense(9, activation='softmax'))
```

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

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential (Sequential)     (None, 224, 224, 1)       0

 conv2d (Conv2D)             (None, 222, 222, 32)      320

 max_pooling2d (MaxPooling2D  (None, 111, 111, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 109, 109, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 54, 54, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 52, 52, 64)        36928

 flatten (Flatten)           (None, 173056)            0

 dense (Dense)               (None, 64)                11075648

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 9)                 585

=================================================================
Total params: 11,131,977
Trainable params: 11,131,977
```

```
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:
```
from keras.callbacks import ModelCheckpoint
simple_callbacks = [
    ModelCheckpoint(filepath="/content/drive/MyDrive/Deep Learning/SavedModels/SimpleModel",
                            monitor='val_accuracy', verbose=1,
                            save_best_only=True, mode='max')]
```

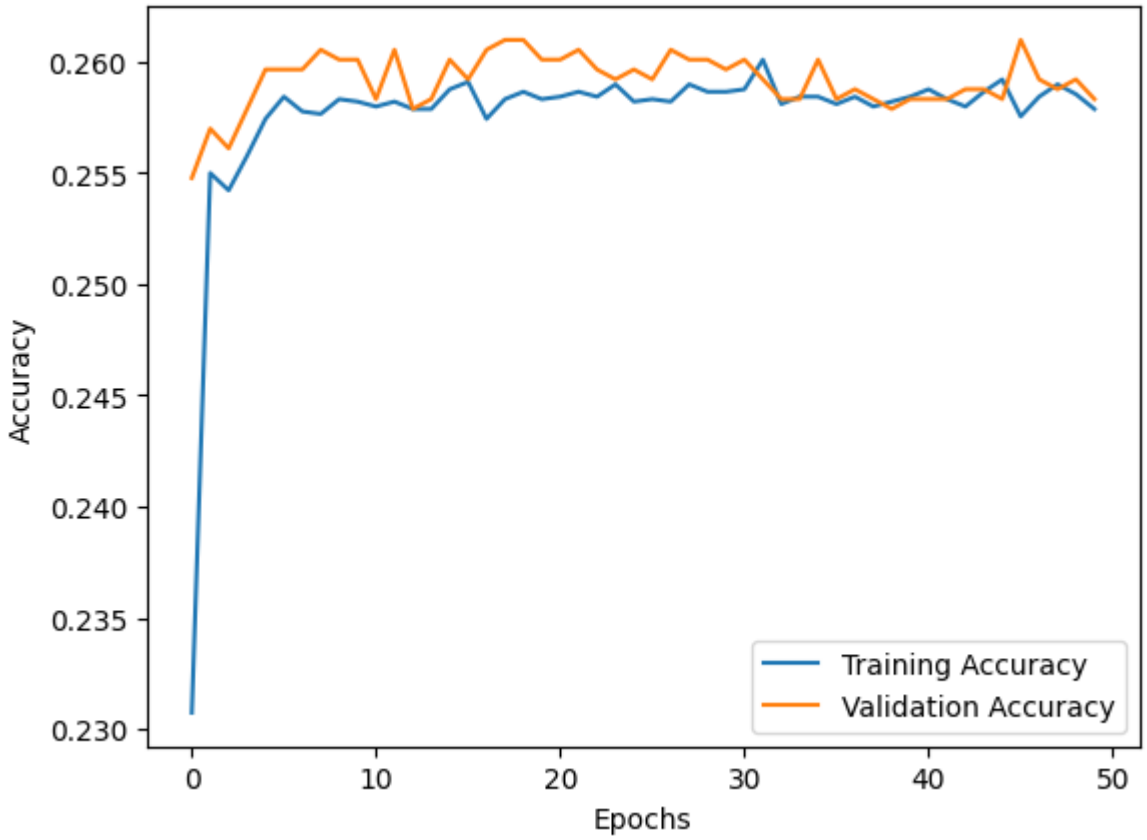
In [ ]:
```
with tf.device(device_name):
  simpleModel.compile(optimizer=tf.keras.optimizers.Adam(1e-3),loss='sparse_categorical_crossentropy', metrics=['accuracy

simple_hist = simpleModel.fit(TypeTrain, epochs=50, callbacks=simple_callbacks, validation_data=TypeVal)
```

```
Epoch 1/50
283/283 [==============================] - ETA: 0s - loss: 7.3917 - accuracy: 0.2307
Epoch 1: val_accuracy improved from -inf to 0.25476, saving model to /content/drive/MyDrive/Deep Learning/SavedModels/Sim
pleModel
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_c
onvolution_op, _update_step_xla while saving (showing 4 of 4). These functions will not be directly callable after loadin
g.
283/283 [==============================] - 1598s 5s/step - loss: 7.3917 - accuracy: 0.2307 - val_loss: 2.0194 - val_accur
acy: 0.2548
Epoch 2/50
283/283 [==============================] - ETA: 0s - loss: 2.0191 - accuracy: 0.2550
Epoch 2: val_accuracy improved from 0.25476 to 0.25698, saving model to /content/drive/MyDrive/Deep Learning/SavedModels/
SimpleModel
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_c
onvolution_op, _update_step_xla while saving (showing 4 of 4). These functions will not be directly callable after loadin
g.
283/283 [==============================] - 57s 196ms/step - loss: 2.0191 - accuracy: 0.2550 - val_loss: 1.9818 - val_accu
racy: 0.2570
Epoch 3/50
283/283 [==============================] - ETA: 0s - loss: 1.9967 - accuracy: 0.2542
Epoch 3: val_accuracy did not improve from 0.25698
283/283 [==============================] - 50s 174ms/step - loss: 1.9967 - accuracy: 0.2542 - val_loss: 1.9727 - val_accu
racy: 0.2561
Epoch 4/50
282/283 [============================>.] - ETA: 0s - loss: 1.9920 - accuracy: 0.2560
Epoch 4: val_accuracy improved from 0.25698 to 0.25786, saving model to /content/drive/MyDrive/Deep Learning/SavedModels/
SimpleModel
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_c
onvolution_op, _update_step_xla while saving (showing 4 of 4). These functions will not be directly callable after loadin
g.
```

```
282/283 [============================>.] - ETA: 0s - loss: 1.9849 - accuracy: 0.2586
Epoch 41: val_accuracy did not improve from 0.26097
283/283 [==============================] - 49s 172ms/step - loss: 1.9849 - accuracy: 0.2587 - val_loss: 1.9721 - val_accu
racy: 0.2583
Epoch 42/50
283/283 [==============================] - ETA: 0s - loss: 1.9853 - accuracy: 0.2583
Epoch 42: val_accuracy did not improve from 0.26097
283/283 [==============================] - 48s 166ms/step - loss: 1.9853 - accuracy: 0.2583 - val_loss: 1.9723 - val_accu
racy: 0.2583
Epoch 43/50
282/283 [============================>.] - ETA: 0s - loss: 1.9862 - accuracy: 0.2581
Epoch 43: val_accuracy did not improve from 0.26097
283/283 [==============================] - 51s 175ms/step - loss: 1.9861 - accuracy: 0.2580 - val_loss: 1.9712 - val_accu
racy: 0.2588
Epoch 44/50
283/283 [==============================] - ETA: 0s - loss: 1.9851 - accuracy: 0.2586
Epoch 44: val_accuracy did not improve from 0.26097
283/283 [==============================] - 49s 171ms/step - loss: 1.9851 - accuracy: 0.2586 - val_loss: 1.9715 - val_accu
racy: 0.2588
Epoch 45/50
283/283 [==============================] - ETA: 0s - loss: 1.9839 - accuracy: 0.2592
Epoch 45: val_accuracy did not improve from 0.26097
283/283 [==============================] - 50s 173ms/step - loss: 1.9839 - accuracy: 0.2592 - val_loss: 1.9716 - val_accu
racy: 0.2583
Epoch 46/50
282/283 [============================>.] - ETA: 0s - loss: 1.9852 - accuracy: 0.2575
Epoch 46: val_accuracy did not improve from 0.26097
283/283 [==============================] - 48s 166ms/step - loss: 1.9854 - accuracy: 0.2575 - val_loss: 1.9704 - val_accu
racy: 0.2610
Epoch 47/50
282/283 [============================>.] - ETA: 0s - loss: 1.9855 - accuracy: 0.2582
Epoch 47: val_accuracy did not improve from 0.26097
283/283 [==============================] - 50s 175ms/step - loss: 1.9853 - accuracy: 0.2584 - val_loss: 1.9711 - val_accu
racy: 0.2592
Epoch 48/50
283/283 [==============================] - ETA: 0s - loss: 1.9856 - accuracy: 0.2590
Epoch 48: val_accuracy did not improve from 0.26097
283/283 [==============================] - 51s 177ms/step - loss: 1.9856 - accuracy: 0.2590 - val_loss: 1.9712 - val_accu
racy: 0.2588
Epoch 49/50
282/283 [============================>.] - ETA: 0s - loss: 1.9848 - accuracy: 0.2585
Epoch 49: val_accuracy did not improve from 0.26097
283/283 [==============================] - 51s 177ms/step - loss: 1.9850 - accuracy: 0.2585 - val_loss: 1.9715 - val_accu
racy: 0.2592
Epoch 50/50
282/283 [============================>.] - ETA: 0s - loss: 1.9857 - accuracy: 0.2578
Epoch 50: val_accuracy did not improve from 0.26097
283/283 [==============================] - 48s 167ms/step - loss: 1.9855 - accuracy: 0.2579 - val_loss: 1.9719 - val_accu
racy: 0.2583
```

In [ ]:
```
#Not great at all.
```

In [ ]:
```
#Plot for accuracy over 50 epochs.
plt.plot(simple_hist.history['accuracy'], label='Training Accuracy')
plt.plot(simple_hist.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.savefig('model_training_history')
plt.show()
```



In [ ]:
```
simpleModel.load_weights('/content/drive/MyDrive/Deep Learning/SavedModels/SimpleModel')
```

Out[ ]:
```
<tensorflow.python.checkpoint.checkpoint.CheckpointLoadStatus at 0x7fa3e00f7880>
```

In [ ]:
```
simpleModel.save('/content/drive/MyDrive/Deep Learning/SavedModels/SimpleVehicleTypeModel.hdf5')
simpleModel.save('/content/drive/MyDrive/Deep Learning/SavedModels/SimpleVehicleTypeModel.h5')
```

In [ ]:
```
simple = tf.keras.models.load_model('/content/drive/MyDrive/Deep Learning/SavedModels/SimpleVehicleTypeModel.h5')
```

In [ ]:
```
#Evaluate the model on the test set.
result = simple.evaluate(TypeTest)
```

```
118/118 [==============================] - 533s 4s/step - loss: 1.9832 - accuracy: 0.2570
```

In [ ]:
```
#Make predictions from test set.
simple_predictions = simple.predict(TypeTest)
```

```
118/118 [==============================] - 13s 106ms/step
```

In [ ]:
```
#Make ground truths of the test set.
file_paths = TypeTest.file_paths
groundTruths = []
testImages = []
for path in file_paths:
  imgFile, groundTruth, img = path.rsplit('/', 2)
  groundTruths.append(groundTruth)
  testImages.append(img)

#Index the ground truths to compare with the highest index from the predictions.
testTruths = []
for i in range(len(groundTruths)):
  testTruths.append(TypeTest.class_names.index(groundTruths[i]))

preds = np.argmax (simple_predictions, axis = 1)
tests = np.array(testTruths)
print(preds)
print(tests)

predicted_labels = []
for i in range(len(preds)):
  predicted_labels.append(TypeTest.class_names[preds[i]])
#Uncomment if you'd like to see predictions vs. ground truths.
```

```
#print(predicted_labels)
#print(groundTruths)
```

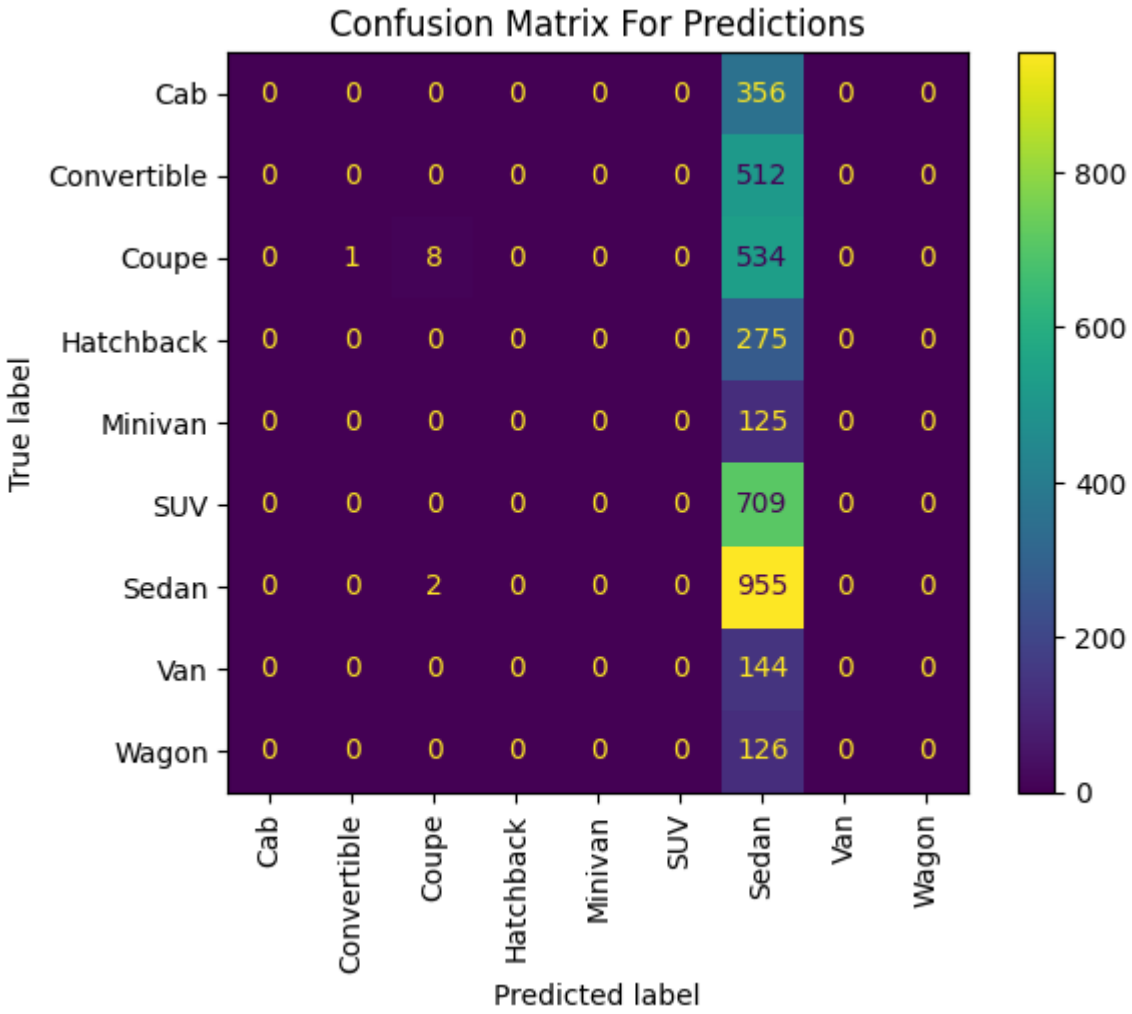
```
[6 6 6 ... 6 6 6]
[0 0 0 ... 8 8 8]
```

In [ ]:

```
from sklearn.metrics import confusion_matrix
from sklearn import metrics
confusionMat = confusion_matrix(tests, preds)
dispConfMat = metrics.ConfusionMatrixDisplay(confusion_matrix = confusionMat)
dispConfMat.plot()
plt.title('Confusion Matrix For Predictions')
plt.xticks(np.arange(9), TypeTest.class_names, rotation=90)
plt.yticks(np.arange(9), TypeTest.class_names)
```

Out[ ]:

```
([<matplotlib.axis.YTick at 0x7fa2e3fe1660>,
  <matplotlib.axis.YTick at 0x7fa2e3fe1120>,
  <matplotlib.axis.YTick at 0x7fa2e3fe0520>,
  <matplotlib.axis.YTick at 0x7fa2e3baf220>,
  <matplotlib.axis.YTick at 0x7fa2e3bdcf40>,
  <matplotlib.axis.YTick at 0x7fa2e3bdd9f0>,
  <matplotlib.axis.YTick at 0x7fa2e3bde4a0>,
  <matplotlib.axis.YTick at 0x7fa2e3bdef50>,
  <matplotlib.axis.YTick at 0x7fa2e3bddde0>],
 [Text(0, 0, 'Cab'),
  Text(0, 1, 'Convertible'),
  Text(0, 2, 'Coupe'),
  Text(0, 3, 'Hatchback'),
  Text(0, 4, 'Minivan'),
  Text(0, 5, 'SUV'),
  Text(0, 6, 'Sedan'),
  Text(0, 7, 'Van'),
  Text(0, 8, 'Wagon')])
```



In [ ]:
```
#This model performs terribly and for some reason really favors the Sedan.
```

In [ ]:
```
from sklearn.metrics import classification_report
print(classification_report(tests, preds))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       356
           1       0.00      0.00      0.00       512
           2       0.80      0.01      0.03       543
```

```
           3       0.00      0.00      0.00       275
           4       0.00      0.00      0.00       125
           5       0.00      0.00      0.00       709
           6       0.26      1.00      0.41       957
           7       0.00      0.00      0.00       144
           8       0.00      0.00      0.00       126

    accuracy                           0.26      3747
   macro avg       0.12      0.11      0.05      3747
weighted avg       0.18      0.26      0.11      3747

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```