```
#!/usr/bin/env python
# coding: utf-8

# In[114]:


from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
from keras.models import model_from_json
import matplotlib.pyplot as plt
import warnings


# In[115]:


warnings.filterwarnings('ignore')
batch_size = 32


# In[116]:


from tensorflow.keras.preprocessing.image import ImageDataGenerator

# All images will be rescaled by 1./255
train_datagen = ImageDataGenerator(rescale=1/255)


# In[117]:


# Flow training images in batches of 128 using train_datagen generator
train_generator = train_datagen.flow_from_directory(
        'G:/CAR_DATASET/Car damage/level/training',  # This is the source directory for training
images
        target_size=(200, 200),  # All images will be resized to 200 x 200
        batch_size=batch_size,
        # Specify the classes explicitly
        classes = ['01-minor','02-moderate','03-severe'],
        # Since we use categorical_crossentropy loss, we need categorical labels
        class_mode='categorical')


# In[118]:


import tensorflow as tf


# In[119]:


#cnn Model
```

```
model = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 200x 200 with 3 bytes color
    # The first convolution
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(200, 200, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The third convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fourth convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fifth convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # Flatten the results to feed into a dense layer
    tf.keras.layers.Flatten(),
    # 128 neuron in the fully-connected layer
    tf.keras.layers.Dense(128, activation='relu'),
    # 5 output neurons for 5 classes with the softmax activation
    tf.keras.layers.Dense(3, activation='softmax')
])


# In[120]:


model.summary()


# In[121]:


model.compile(loss="categorical_crossentropy",metrics=["accuracy"],optimizer='adam')


# In[122]:


from tensorflow.keras.optimizers import RMSprop


# In[123]:


early = tf.keras.callbacks.EarlyStopping(monitor='val_loss',patience=5)
model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(lr=0.001),
              metrics=['accuracy'])


# In[124]:


total_sample=train_generator.n
```

```
# In[125]:


n_epochs = 10


# In[126]:


history = model.fit_generator(
            train_generator,
            steps_per_epoch=int(total_sample/batch_size),
            epochs=n_epochs,
            verbose=1)


# In[127]:


model.save('level.h5')


# In[128]:


model


# In[129]:


acc = history.history['accuracy']


# In[130]:


loss = history.history['loss']


# In[131]:


epochs = range(1, len(acc) + 1)


# In[132]:


# Train and validation accuracy
plt.plot(epochs, acc, 'b', label='accurarcy')


# In[133]:
```

```
plt.title('    accurarcy')
plt.legend()


# In[134]:


plt.figure()


# In[135]:


# Train and validation loss
plt.plot(epochs, loss, 'b', label=' loss')
plt.title('    loss')
plt.legend()
plt.show()


# In[ ]:




# In[ ]:




# In[ ]:




# In[ ]:




# In[ ]:
```