Saving pkl as well as validated image

import keras

import pickle

class HistoryCheckpoint(keras.callbacks.Callback):

def \_\_init\_\_(self, filepath):

self.filepath = filepath

def on\_epoch\_end(self, epoch, logs=None):

with open(self.filepath.format(epoch=epoch+1), 'wb') as f:

pickle.dump(self.model.history.history, f)

# Define the model architecture

model = keras.Sequential([keras.layers.Dense(10, input\_shape=(4,), activation='softmax')])

# Compile the model

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

# Define the image saving callback

image\_save\_callback = keras.callbacks.LambdaCallback(

on\_epoch\_end=lambda epoch, logs: keras.preprocessing.image.save\_img(

f"valid\_img\_{epoch}.png", validation\_data[0][0]

)

)

# Define the model saving callback

model\_save\_callback = keras.callbacks.ModelCheckpoint(

'path/to/best\_model.h5', monitor='val\_accuracy', save\_best\_only=True, mode='max'

)

# Train the model and save history after each epoch

filepath = 'path/to/history\_epoch\_{epoch:02d}.pkl'

checkpoint = HistoryCheckpoint(filepath)

model.fit\_generator(

train\_generator,

steps\_per\_epoch=train\_steps\_per\_epoch,

epochs=3,

validation\_data=validation\_generator,

validation\_steps=validation\_steps\_per\_epoch,

callbacks=[checkpoint, image\_save\_callback, model\_save\_callback]

)

import keras

import pickle

class HistoryCheckpoint(keras.callbacks.Callback):

def \_\_init\_\_(self, filepath):

self.filepath = filepath

def on\_epoch\_end(self, epoch, logs=None):

# Save model history

with open(self.filepath.format(epoch=epoch+1, mode='history'), 'wb') as f:

pickle.dump(self.model.history.history, f)

# Save validation images

if hasattr(self.model, 'validation\_data'):

x\_val, y\_val = self.model.validation\_data[0], self.model.validation\_data[1]

y\_pred = self.model.predict(x\_val)

num\_images = min(len(x\_val), 10) # Save up to 10 images

for i in range(num\_images):

img = x\_val[i]

true\_label = y\_val[i]

pred\_label = y\_pred[i]

with open(self.filepath.format(epoch=epoch+1, mode=f'val\_img\_{i+1}'), 'wb') as f:

pickle.dump({'image': img, 'true\_label': true\_label, 'pred\_label': pred\_label}, f)

# Define the model architecture

model = keras.Sequential([keras.layers.Dense(10, input\_shape=(4,), activation='softmax')])

# Compile the model

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model and save history and validation images after each epoch

filepath = 'path/to/epoch\_{epoch:02d}\_{mode}.pkl'

checkpoint = HistoryCheckpoint(filepath)

model.fit(x\_train, y\_train, epochs=3, callbacks=[checkpoint], validation\_data=(x\_val, y\_val))