Plotting pkl file

import matplotlib.pyplot as plt

import pickle

# Load the model from its .h5 file

model = keras.models.load\_model('path/to/model.h5')

# Load the history from a separate file

with open('path/to/history.pkl', 'rb') as file:

history = pickle.load(file)

# Plot the loss

plt.plot(history['loss'])

plt.plot(history['val\_loss'])

plt.title('Model Loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend(['Training', 'Validation'], loc='upper right')

plt.show()

# Plot the accuracy

plt.plot(history['acc'])

plt.plot(history['val\_acc'])

plt.title('Model Accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.legend(['Training', 'Validation'], loc='lower right')

plt.show()

Training example code and saving pkl file

import keras

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Dense, Dropout

from keras.optimizers import RMSprop

import pickle

# Load the MNIST dataset

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

# Preprocess the data

x\_train = x\_train.reshape(60000, 784).astype('float32') / 255

x\_test = x\_test.reshape(10000, 784).astype('float32') / 255

y\_train = keras.utils.to\_categorical(y\_train, 10)

y\_test = keras.utils.to\_categorical(y\_test, 10)

# Define the model architecture

model = Sequential()

model.add(Dense(512, activation='relu', input\_shape=(784,)))

model.add(Dropout(0.2))

model.add(Dense(10, activation='softmax'))

# Compile the model

model.compile(loss='categorical\_crossentropy',

optimizer=RMSprop(),

metrics=['accuracy'])

# Train the model and store the history

history = model.fit(x\_train, y\_train,

batch\_size=128,

epochs=10,

verbose=1,

validation\_data=(x\_test, y\_test))

# Save the model

model.save('path/to/model.h5')

# Save the history object separately

with open('path/to/history.pkl', 'wb') as file:

pickle.dump(history.history, file)

Pkl passing as 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):

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'])

# Train the model and save history after each epoch

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

checkpoint = HistoryCheckpoint(filepath)

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