Application to Our Script

Saving our model

Step 1.

```
from keras.callbacks import ModelCheckpoint, EarlyStopping
```

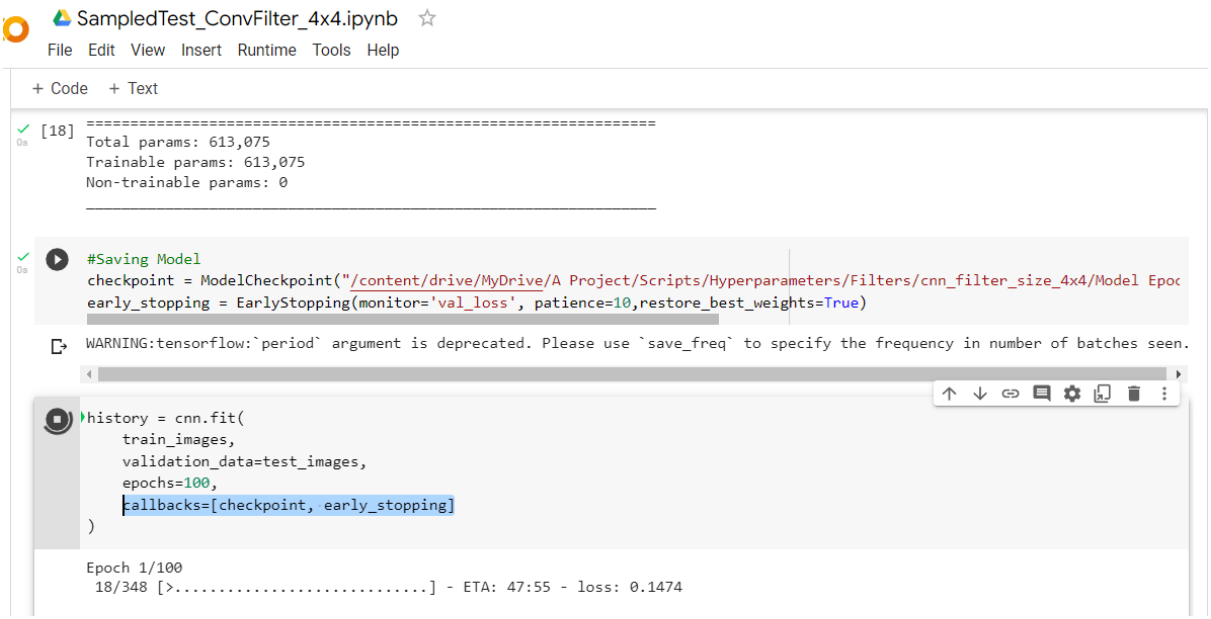
```
import numpy as np
import pandas as pd
import matplotlib as mp
import tensorflow as tf
import math, os, glob, re
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from keras.callbacks import ModelCheckpoint, EarlyStopping
ss = StandardScaler()
from tensorflow.keras.preprocessing.image import ImageDataGenerator

import matplotlib.pyplot as plt
#from numpy.core.fromnumeric import argmax
from sklearn.model_selection import train_test_split
```

Step 2. (Add in the saving model code and change **saving location** and the line highlighted)

```
#Saving Model
checkpoint = ModelCheckpoint("/content/drive/MyDrive/A
Project/Scripts/Hyperparameters/Filters/cnn_filter_size_4x4/Model
Epochs/model-{epoch:02d}.h5", save_best_only=False,
save_weights_only=False, period=1)
early_stopping = EarlyStopping(monitor='val_loss',
patience=10,restore_best_weights=True)
```

```
cnnhistory = cnn.fit(
    train_images,
    validation_data=test_images,
    epochs=100,
    callbacks=[checkpoint, early_stopping]
)
```



Loading model

Add in this block under CNN Architecture before `def make_model(dim1,dim2):`

```
model = tf.keras.models.load_model("/content/drive/MyDrive/A Project/Scripts/Hyperparameters/Filters/cnn_filter_size_4x4/Model Epochs/model-24.h5") #Change the location
model.compile(optimizer='adam', loss='mse')
#Saving Model
checkpoint = ModelCheckpoint("/content/drive/MyDrive/A Project/Scripts/Hyperparameters/Filters/cnn_filter_size_4x4/Model Epochs/model-{epoch:02d}.h5", save_best_only=False, save_weights_only=False, period=1)#Change the location
early_stopping = EarlyStopping(monitor='val_loss', patience=10,restore_best_weights=True)
```

Taking into consideration the previous epochs

Loading history file

```
# Load the training history from each epoch
history_list = []
for i in range(50):
  history = tf.keras.callbacks.History()
  history.load("/content/gdrive/My Drive/training_history_{}.h5".format(i))
  history_list.append(history.history)

# Combine the training history of all epochs into a single DataFrame
history_df = pd.concat(history_list, axis=1)
```

```
# Transpose the DataFrame to have epochs as rows and metrics as columns
history_df = history_df.T

# Save the training history to a CSV file
history_df.to_csv("/content/gdrive/My Drive/training_history.csv", index=False)
```

Taking previous epochs

```
# Load the latest checkpoint and the training history from Google Drive
checkpoint_path = "/content/gdrive/My Drive/checkpoint.h5"
history_path = "/content/gdrive/My Drive/history.csv"

# Load the latest checkpoint and the training history
checkpoint = checkpoint_path
history = pd.read_csv(history_path)

# Load the model and continue training from the epoch in the checkpoint
model = tf.keras.models.load_model(checkpoint)

# Get the epoch number from the checkpoint
initial_epoch = tf.train.epoch_from_checkpoint(checkpoint)

# Get the index of the latest improved epoch
latest_improved_epoch = np.argmin(history.validation_loss.values[:initial_epoch])

# Calculate the number of epochs since the latest improved epoch
epochs_since_improvement = initial_epoch - latest_improved_epoch

# Check if the performance has stopped improving based on the patience value
if epochs_since_improvement >= patience:
    print("Performance has stopped improving.")
else:
    model.fit(train_data, train_labels, epochs=50, initial_epoch=initial_epoch)

# Save the model checkpoint and the training history to Google Drive
model.save(checkpoint_path)
history.to_csv(history_path, index=False)
```

Training the model with this block

```
history = model.fit(
    train_images,
    validation_data=test_images,
    epochs=100,
    initial_epoch=24,
    callbacks=[checkpoint, early_stopping]
)
```

Plotting the whole model (haven't tried this yet)

```
import numpy as np
import matplotlib.pyplot as plt
import os
import h5py # remember to import this

num_history_files = 3 # change this

history = {}
history['loss'] = []
history['val_loss'] = []

for i in range(num_history_files):
    history_file = h5py.File(f'/content/drive/My Drive/model-{i}.h5', 'r')
    history['loss'].extend(history_file['loss'][:])
    history['val_loss'].extend(history_file['val_loss'][:])

plt.plot(history['loss'], label='loss')
plt.plot(history['val_loss'], label='val_loss')
plt.legend()
plt.show()
```

Load the history

```
import h5py

history = []

for i in range(num_history_files):
  filename = '/content/gdrive/My Drive/history_{}.h5'.format(i)
  history.append(h5py.File(filename, 'r'))

compiled_history = {
  'loss': [],
  'acc': []
}

for h in history:
```

```
  compiled_history['loss'].extend(h['loss'][:])
  compiled_history['val_loss'].extend(h['val_loss'][:])

with h5py.File('/content/gdrive/My Drive/compiled_history.h5', 'w') as f:
  f.create_dataset('loss', data=compiled_history['loss'])
  f.create_dataset('val_loss', data=compiled_history['val_loss'])
```

Reference

Saving Model at Each Epoch

If you want to save all the versions of your model after each epoch, you can modify the filename to include the epoch number, like this:

```
from keras.callbacks import ModelCheckpoint
from google.colab import drive

# Mount Google Drive
drive.mount('/content/gdrive')

# Define the ModelCheckpoint callback
checkpoint = ModelCheckpoint("/content/gdrive/My Drive/model-{epoch:02d}.h5", save_best_only=False, save_weights_only=False, period=1)

# Fit the model and specify the callbacks
model.fit(x_train, y_train, epochs=100, callbacks=[checkpoint])
```

In this example, the model will be saved after each epoch to a file named "model-XX.h5", where XX is the epoch number with leading zeros. The `save_best_only` argument is set to False, which means that all the models after each epoch will be saved. The `period` argument specifies the number of epochs after which the model should be saved.

Training Model from Specific Epoch (eg. 48)

To train a model starting from epoch 48, you need to load the weights from a previously saved model and then continue training from there. Here's an example in Keras:

```
from keras.callbacks import ModelCheckpoint
from google.colab import drive
```

```
# Mount Google Drive
drive.mount('/content/gdrive')

# Load the weights from a previously saved model
model.load_weights("/content/gdrive/My Drive/model-48.h5")

# Compile the model (if needed)
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Define the ModelCheckpoint callback
checkpoint = ModelCheckpoint("/content/gdrive/My Drive/model-{epoch:02d}.h5",
save_best_only=False, save_weights_only=False, period=1)

# Fit the model and specify the callbacks
model.fit(x_train, y_train, epochs=100, initial_epoch=48, callbacks=[checkpoint])
```

In this example, the model is first loaded with the weights from a previously saved model named "model-48.h5". The model is then recompiled (if needed), and finally fit to the training data, starting from epoch 48 with the `initial_epoch` argument. The rest of the code is the same as in the previous example.

Prevent Timeout Due to Inactivity

[Reference]

Press `Ctrl+ Shift + i`

Paste this code in the console

```
function ClickConnect(){
console.log("Working");
document.querySelector("#top-toolbar >
colab-connect-button").shadowRoot.querySelector("#connect").click();
}
var clicker = setInterval(ClickConnect,60000);
```

To stop this

```
clearInterval(clicker);
```

From my experience, google colab still asks if you are there after the first 30 mins you run the code but that only happened to me once and only at the beginning 30 mins. It would be still be a good idea to just click on the page every so often to make sure it is working.

Update: it ran for 2 hours and it suddenly popped up asking if im there so ig it might not work :C