1. Download all the data in this folder https://drive.google.com/open?id=1Z4TyI7FcFVEx8qdl4jO9qxvxaqLSqoEu. it contains two file both images and labels. The label file list the images and their categories in the following format:

path/to/the/image.tif,category

where the categories are numbered 0 to 15, in the following order:

0 letter
1 form
2 email
3 handwritten
4 advertisement
5 scientific report
6 scientific publication
7 specification
8 file folder
9 news article
10 budget
11 invoice
12 presentation
13 questionnaire
14 resume
15 memo

2. On this image data, you have to train 3 types of models as given below. You have to split the data into Train and Validation data.

3. Try not to load all the images into memory, use the gernarators that we have given the reference notebooks to load the batch of images only during the train data.
or you can use this method also
https://medium.com/@vijayabhaskar96/tutorial-on-keras-imagedatagenerator-with-flow-from-dataframe-8bd5776e45c1 (https://medium.com/@vijayabhaskar96/tutorial-on-keras-imagedatagenerator-with-flow-from-dataframe-8bd5776e45c1)

https://medium.com/@vijayabhaskar96/tutorial-on-keras-flow-from-dataframe-1fd4493d237c (https://medium.com/@vijayabhaskar96/tutorial-on-keras-flow-from-dataframe-1fd4493d237c)

4. You are free to choose Learning rate, optimizer, loss function, image augmentation, any hyperparameters. but y

ou have to use the same architechture what we are asking below.

5. Use tensorboard for every model and analyse your gradients. (you need to upload the screenshots for each model for evaluation)

Note: fit_genarator() method will have problems with the tensorboard histograms, try to debug it, if you could not do use histgrams=0 i.e don't include histograms, check the documentation of tensorboard for more information.

6. You can check about Transfer Learning in this link - [https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html (https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html)](https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html)

In [2]:
```
%tensorflow_version 2.x
```

In [3]:
```
import numpy as np
import pandas as pd
import os
import tensorflow as tf
from sklearn.model_selection import train_test_split
```

In [4]:
```
tf.__version__
```

Out[4]: '2.4.1'

In [10]:
```
# goto this link: https://drive.google.com/uc?id=1Z4TyI7FcFVEx8qdl4jO9qxvxaqLSqoEu
# click download anyway
# cancel the downloading...
# copy the link from CurlWget extension(if you don't have, install)
# then do => ! copied link from CurlWget extension

! wget --header="Host: doc-04-0k-docs.googleusercontent.com" --header="User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x6
```

```
--2021-03-23 01:25:38--  https://doc-04-0k-docs.googleusercontent.com/docs/securesc/8iph74e76aalmpco83tp1i1i1vloj61q/55f
323hme70a657dflj9sohnf4o8ub0h/1616462700000/00484516897554883881/08709549031353296611/1Z4TyI7FcFVEx8qdl4jO9qxvxaqLSqoEu?
e=download&authuser=0&nonce=2ld3kjsa27i4u&user=08709549031353296611&hash=lrp77usmuks0kc1foptibqeavm5m2cod (https://doc-0
4-0k-docs.googleusercontent.com/docs/securesc/8iph74e76aalmpco83tp1i1i1vloj61q/55f323hme70a657dflj9sohnf4o8ub0h/16164627
00000/00484516897554883881/08709549031353296611/1Z4TyI7FcFVEx8qdl4jO9qxvxaqLSqoEu?e=download&authuser=0&nonce=2ld3kjsa27
i4u&user=08709549031353296611&hash=lrp77usmuks0kc1foptibqeavm5m2cod)
Resolving doc-04-0k-docs.googleusercontent.com (doc-04-0k-docs.googleusercontent.com)... 172.217.15.97, 2607:f8b0:4004:8
11::2001
Connecting to doc-04-0k-docs.googleusercontent.com (doc-04-0k-docs.googleusercontent.com)|172.217.15.97|:443... connecte
d.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/rar]
Saving to: ‘rvl-cdip.rar’

rvl-cdip.rar            [            <=>      ]   4.34G  44.5MB/s    in 87s

2021-03-23 01:27:05 (51.4 MB/s) - ‘rvl-cdip.rar’ saved [4660541790]
```

In [11]:
```
!pip install patool
import patoolib
patoolib.extract_archive('rvl-cdip.rar')
```

```
Requirement already satisfied: patool in /usr/local/lib/python3.7/dist-packages (1.12)
patool: Extracting rvl-cdip.rar ...
patool: running /usr/bin/unrar x -- /content/rvl-cdip.rar
patool:     with cwd='./Unpack_i_c7lxoo'
patool: ... rvl-cdip.rar extracted to `rvl-cdip' (multiple files in root).
```

Out[11]:
```
'rvl-cdip'
```

In [12]:
```
# !gdown --id 1Z4TyI7FcFVEx8qdl4jO9qxvxaqLSqoEu
# get_ipython().system_raw("unrar x rvl-cdip.rar")
```

In [13]:
```
os.listdir('/content')
```
Out[13]:
```
['.config', 'rvl-cdip.rar', 'rvl-cdip', 'sample_data']
```

In [14]:
```
dir_path = '/content/rvl-cdip/data_final'
```

In [15]:
```
# reading label dataframe
imagelabel = pd.read_csv('/content/rvl-cdip/labels_final.csv',dtype=str)
imagelabel.head()
```
Out[15]:

	path	label
0	imagesv/v/o/h/voh71d00/509132755+-2755.tif	3
1	imagesl/l/x/t/lxt19d00/502213303.tif	3
2	imagesx/x/e/d/xed05a00/2075325674.tif	2
3	imageso/o/j/b/ojb60d00/517511301+-1301.tif	3
4	imagesq/q/z/k/qzk17e00/2031320195.tif	7

Let's see the distribution of the dataset

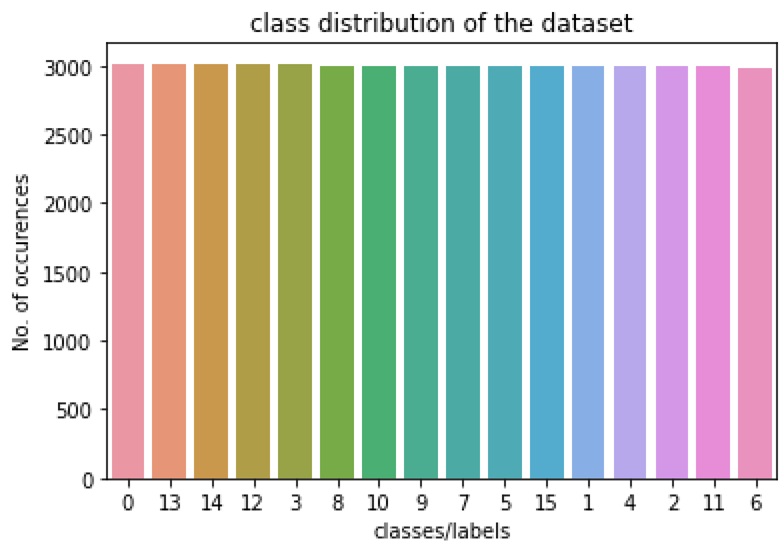
In [16]:
```
class_dist = dict(imagelabel['label'].value_counts())
```

In [17]:
```
class_dist
```
Out[17]:
```
{'0': 3016,
 '1': 2994,
 '10': 3002,
 '11': 2992,
 '12': 3006,
 '13': 3007,
 '14': 3006,
 '15': 2996,
 '2': 2993,
 '3': 3005,
 '4': 2994,
 '5': 2999,
 '6': 2985,
 '7': 3000,
 '8': 3003,
 '9': 3002}
```

In [18]:
```python
import seaborn as sns
import matplotlib.pyplot as plt
```

In [19]:
```python
sns.barplot( list(class_dist.keys()), list(class_dist.values()) )
plt.title('class distribution of the dataset')
plt.xlabel('classes/labels')
plt.ylabel('No. of occurences')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



Observation:

1. Data is balanced. So, there is no need to balance the class.

In [19]:

In [20]:
```python
# imagelabel['label'] = pd.Series(map(lambda x: str(x), imagelabel.label))
```

In [21]:
```python
imagelabel.label.iloc[0]
```

Out[21]:
```
'3'
```

In [22]:
```python
# an example of data point
import matplotlib.pyplot as plt
plt.figure(figsize=(10,10))
img_index = 2340
img = plt.imread(os.path.join(dir_path, imagelabel.path.iloc[img_index]))
plt.imshow(img)
plt.title(imagelabel.label.iloc[img_index])
```

Out[22]: `Text(0.5, 1.0, '4')`

In [23]:
```
#importing tensorflow
from tensorflow.keras.layers import Dense,Input,Conv2D,MaxPool2D,Activation,Dropout,Flatten
from tensorflow.keras.models import Model, Sequential
import random as rn
```

In [23]:

Image Data Generator

Before doing Image Data Generator let's split the dataset using stratified sampling

In [24]:
```
train_df, validation_df = train_test_split(imagelabel,
                                           test_size=0.2,
                                           random_state=42,
                                           shuffle=True,
                                           stratify = imagelabel['label']
                                           )
```

In [25]:
```
train_df.head()
```

Out[25]:

	path	label
10155	imagesj/j/i/w/jiw43c00/89638854_8861.tif	6
45402	imagesb/b/x/k/bxk05e00/2040785858.tif	10
34244	imagesi/i/u/d/iud42e00/2500090606_2500090631.tif	12
46324	imagesp/p/v/z/pvz46c00/2505161384_1385.tif	1
25642	imagesf/f/q/c/fqc66d00/504315522+-5526.tif	3

In [26]:
```
train_df.label.value_counts()
```

Out[26]:
```
0     2413
14    2405
13    2405
12    2405
3     2404
8     2402
9     2402
10    2402
7     2400
5     2399
15    2397
1     2395
4     2395
2     2394
11    2394
6     2388
Name: label, dtype: int64
```

In [27]:
```

validation_df.head()
```

Out[27]:

	path	label
8538	imagesx/x/p/d/xpd30a00/60004171_60004175.tif	6
2381	imagesd/d/z/y/dzy35e00/2040765737_2040765738.tif	10
15246	imagese/e/k/q/ekq43f00/0030049206.tif	4
32549	imagesc/c/m/q/cmq21a00/0071019330.tif	13
41562	imagesq/q/d/i/qdi71a00/2057432757_2057432767.tif	7

In [28]:
```
validation_df.label.value_counts()
```

Out[28]:
```
0     603
13    602
8     601
3     601
14    601
12    601
9     600
5     600
7     600
10    600
15    599
1     599
4     599
2     599
11    598
6     597
Name: label, dtype: int64
```

In [28]:
```

```

In [29]:
```
imagelabel.iloc[33416] # we are cross-checking our dataset
```

Out[29]:
```
path     imagesd/d/u/f/duf23c00/96313362.tif
label                                      4
Name: 33416, dtype: object
```

In [30]:
```
h,w = 224, 224 # image should be resized
```

In [31]:

```
### Image data generator class
datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,
    rotation_range=10,
    shear_range=0.3,
    zoom_range=0.2
    )
test_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255
)

train_dataset = datagen.flow_from_dataframe(
                dataframe=train_df,
                x_col='path',
                y_col='label',
                directory=dir_path,
                target_size=(h,w),
                seed=42,
                batch_size=32, # use small batch size to overcome memory warning
                # subset="training",
                class_mode="sparse"
            )
validation_dataset = test_datagen.flow_from_dataframe(
                dataframe=validation_df,
                x_col='path',
                y_col='label',
                directory=dir_path,
                target_size=(h,w),
                seed=42,
                batch_size=32,
                # subset="validation",
                class_mode="sparse"
            )
```

```
Found 38400 validated image filenames belonging to 16 classes.
Found 9600 validated image filenames belonging to 16 classes.
```

In [32]:

```
n_train_sample = int(imagelabel.shape[0]*(1-0.2))
n_val_sample   = int(imagelabel.shape[0]*0.2)
batch_size=32
n_train_sample, n_val_sample
```

Out[32]:

```
(38400, 9600)
```

Also include callbacks

In [33]:
```python
# save model if accuracy is improved
from tensorflow.keras.callbacks import ModelCheckpoint
filepathw = 'model_save/weights-{epoch:02d}-{val_accuracy:.4f}.hdf5'

checkpoint = ModelCheckpoint(
 filepath = filepathw,
 monitor ='val_accuracy',
 verbose = 1,
 save_best_only=True,
 mode = 'max' # max in case of accuracy # min=> loss # auto=> detect automatic
```

In [34]:
```python
# decay learning rate
class decayLR(tf.keras.callbacks.Callback):
    """
    This is custom callback class to implement decay the Learning Rate.
    Author: Muhammad Iqbal Bajmi @AppliedAICourse.com
    """
    def on_train_begin(self, logs={}):
        self.valid_acc = {'accuracy':[]}

    def on_epoch_begin(self, epoch, logs={}):
        print(epoch)
        print("Learning rate is : {}".format(float(self.model.optimizer.learning_rate)))
        if epoch>1:
            if (epoch+1)%3==0:
                lr = self.model.optimizer.learning_rate
                self.model.optimizer.learning_rate = self.model.optimizer.learning_rate - (lr*0.05)
    def on_epoch_end(self, epoch, logs={}):
        self.valid_acc['accuracy'].append(logs.get('val_accuracy'))
        if epoch>0: # because epoch starts from 0
            if self.valid_acc['accuracy'][epoch] < self.valid_acc['accuracy'][epoch-1]:
                lr = self.model.optimizer.learning_rate
                self.model.optimizer.learning_rate = self.model.optimizer.learning_rate - (lr*0.1)

decaylr = decayLR()
```

In [35]:
```python
# TerminateOnNan from official tensorlfow
from tensorflow.keras.callbacks import TerminateOnNaN
terminate = TerminateOnNaN()
```

In [36]:
```python
# Terminateon Nan
class TerminateNaN(tf.keras.callbacks.Callback):
    def __init__(self):
        self.we = 0
    def on_epoch_end(self, epoch, logs={}):
        import numpy as np
        loss = logs.get('loss')
        w = self.model.get_weights()
        w = np.array(w)
        self.we = w
        if loss is not None:
            if np.isnan(loss) or np.isinf(loss):
                print("Invalid loss and terminated at epoch {}".foramt(epoch))
                self.model.stop_training = True
        no_of_layers = len(self.model.layers)
        for i in range(no_of_layers):
            layer = self.model.layers[i]
            weights = layer.get_weights()[0]
            biases = layer.get_weights()[1]

            if np.isnan(weights).any() or np.isinf(weights).any():
                print("Invalid weights and terminated at epoch ={}".format(epoch))
                self.model.stop_training = True

terminate_nan = TerminateNaN()
```

In [37]:
```python
# Stop training if validation_accuracy not improved
from tensorflow.keras.callbacks import EarlyStopping

earlystop = EarlyStopping(
 monitor = 'val_accuracy',
 mode = 'max',
 patience = 2 # wait for 2 epochs, terminate if no improvement
)
```

In [38]:
```python
import datetime
log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1, write_graph=1)
```

In [39]:
```python
callback_list = [checkpoint, decaylr,terminate,earlystop,tensorboard_callback]
```

Model-1

1. Use [VGG-16 (https://www.tensorflow.org/api_docs/python/tf/keras/applications/VGG16)](https://www.tensorflow.org/api_docs/python/tf/keras/applications/VGG16) pretrained network without Fully Connected layers and initilize all the weights with Imagenet trained weights.
2. After VGG-16 network without FC layers, add a new Conv block (1 Conv layer and 1 Maxpooling), 2 FC layers and a output layer to classify 16 classes. You are free to choose any hyperparameters/parameters of conv block, FC layers, output layer.
3. Final architecture will be **INPUT --> VGG-16 without Top layers(FC) --> Conv Layer --> Maxpool Layer --> 2 FC layers --> Output Layer**
4. Train only new Conv block, FC layers, output layer. Don't train the VGG-16 network.

Adding Layers to the model and then train

In [38]:

```

```

In [47]:
```
# clear all previous sessions
tf.keras.backend.clear_session()
# initializing sequential model
model_1 = tf.keras.models.Sequential()
# load VGG16
model = tf.keras.applications.VGG16(
                          include_top = False,
                          weights = 'imagenet',
                          input_shape=(h,w,3)
                          )
# making all layers.trainable False
for layer in model.layers:
  layer.trainable = False

model_1.add(model) # adding VGG model to our Sequential model
model_1.add(Conv2D(filters=256,
                         kernel_size=(3,3),
                         strides = (1,1),
                         padding='valid',
                         data_format = 'channels_last',
                         activation='relu',
                         kernel_initializer=tf.keras.initializers.he_normal(seed=0),
                         name = 'Conv1'
                         #input_shape = model.output_shape[1:]
                        ))
model_1.add(MaxPool2D(pool_size=(2,2),
                            strides=(2,2),
                            padding='valid',
                            data_format='channels_last',
                            name='Pool1'
                           ))
model_1.add(Flatten(data_format='channels_last', name='Flatten'))
model_1.add(Dense(units=256,
                  activation='relu',
                  kernel_initializer=tf.keras.initializers.glorot_normal(seed=32),name='FC1'
                  ))
model_1.add(Dense(units=128,
                  activation='relu',
                  kernel_initializer=tf.keras.initializers.glorot_normal(seed=32),name='FC2'
                  ))
model_1.add(Dense(units=64,
                  activation='relu',
                  kernel_initializer=tf.keras.initializers.glorot_normal(seed=33),name='FC3'
                  ))
model_1.add(Dense(units=16,
                  activation='softmax',
                  kernel_initializer=tf.keras.initializers.glorot_normal(seed=3), name='Output'
```

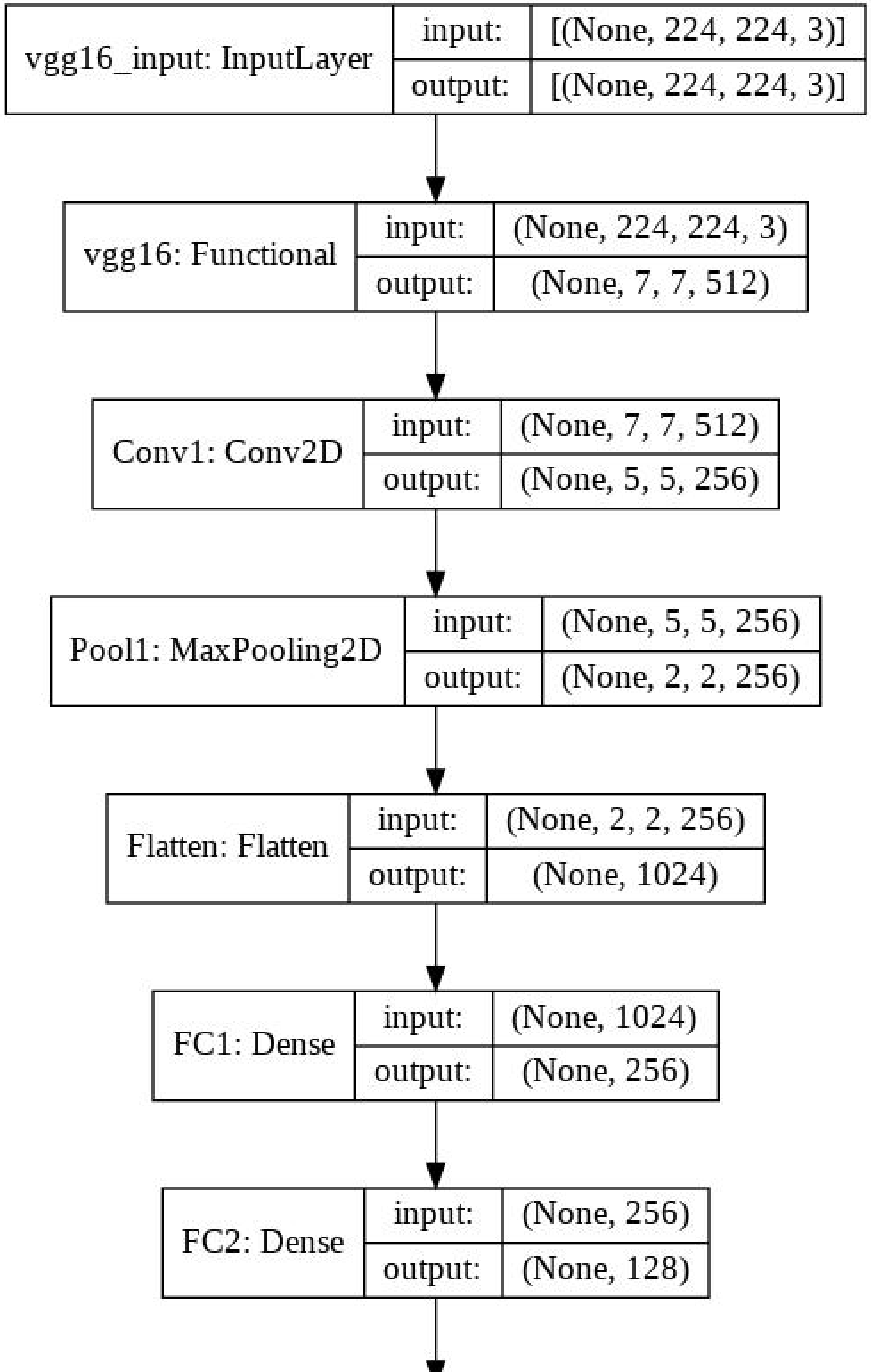
```
                  ))
```

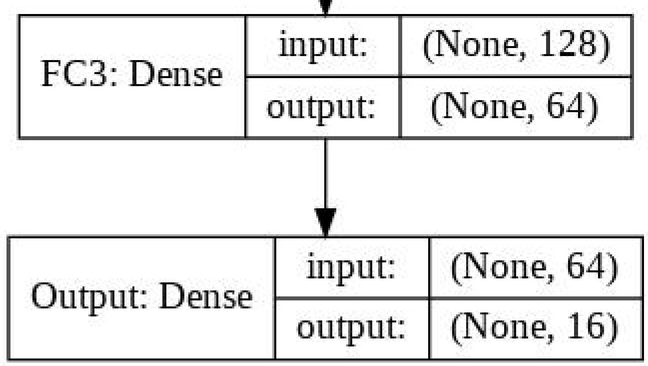
In [48]:
```
model_1.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
vgg16 (Functional)           (None, 7, 7, 512)         14714688  
_________________________________________________________________
Conv1 (Conv2D)               (None, 5, 5, 256)         1179904   
_________________________________________________________________
Pool1 (MaxPooling2D)         (None, 2, 2, 256)         0         
_________________________________________________________________
Flatten (Flatten)            (None, 1024)              0         
_________________________________________________________________
FC1 (Dense)                  (None, 256)               262400    
_________________________________________________________________
FC2 (Dense)                  (None, 128)               32896     
_________________________________________________________________
FC3 (Dense)                  (None, 64)                8256      
_________________________________________________________________
Output (Dense)               (None, 16)                1040      
=================================================================
Total params: 16,199,184
Trainable params: 1,484,496
Non-trainable params: 14,714,688
_________________________________________________________________
```

In [49]:
```
tf.keras.utils.plot_model(model_1, to_file='model1.jpeg',show_shapes=True)
```

Out[49]:




FC3: Dense
input:
(None, 128)
output:
(None, 64)
Output: Dense
input:
(None, 64)
output:
(None, 16)

In [51]:

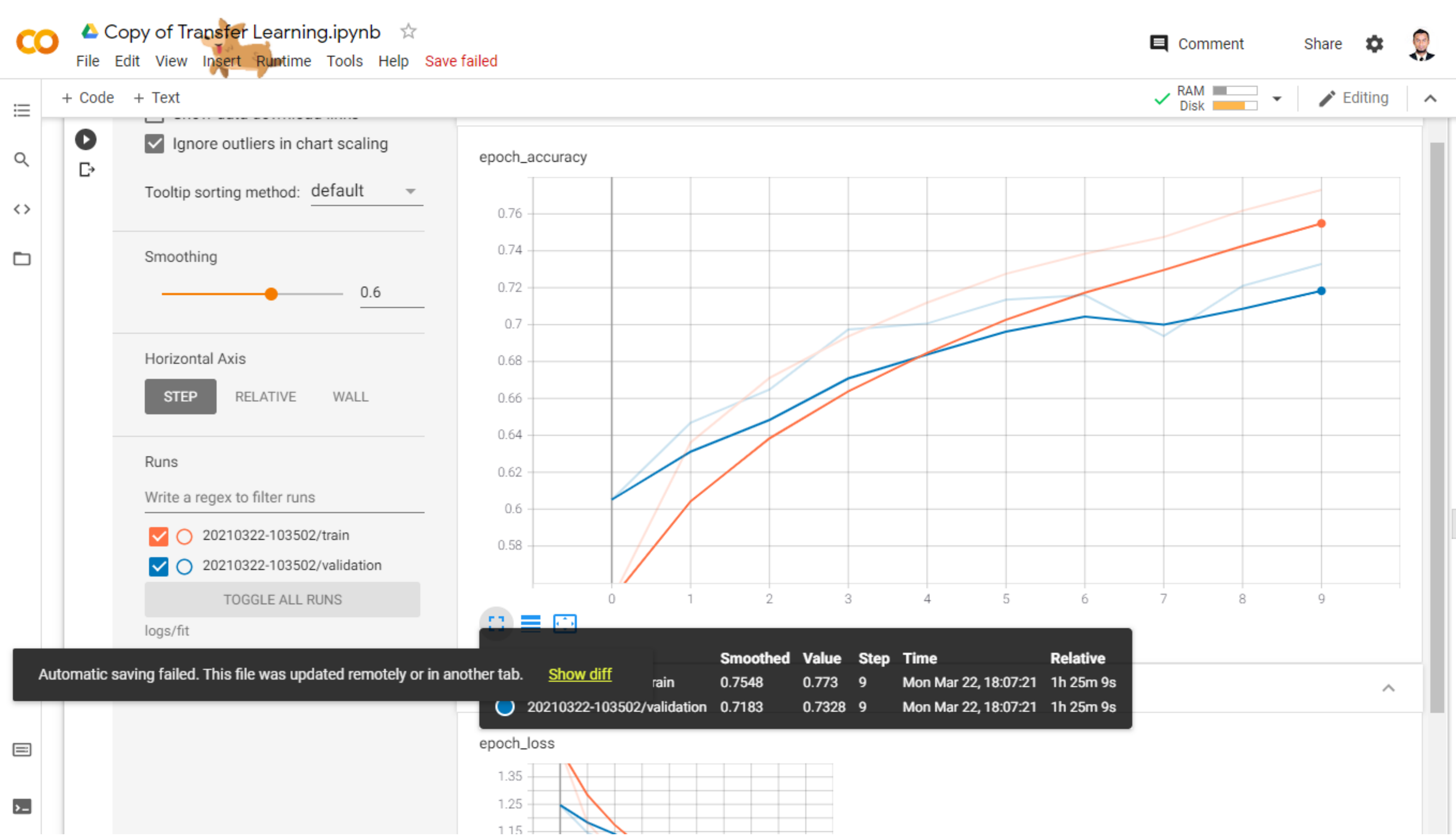
```
# remove all previous logs
!rm -rf logs
# remove all saved models
!rm -rf model_save
#compile the model
model_1.compile(optimizer=tf.keras.optimizers.Adam(),
                  loss = 'sparse_categorical_crossentropy',
                  # loss=tf.keras.losses.sparse_categorical_crossentropy(), # use sparse categorical cross entropy for nor
                  # loss='categorical_crossentropy', # use categorical cross entropy for one-hot encoded labels
                  metrics=['accuracy'])

# fit the model
model_1.fit(x = train_dataset,
              validation_data=validation_dataset,
              epochs=10,
              callbacks=[callback_list]
              )
```

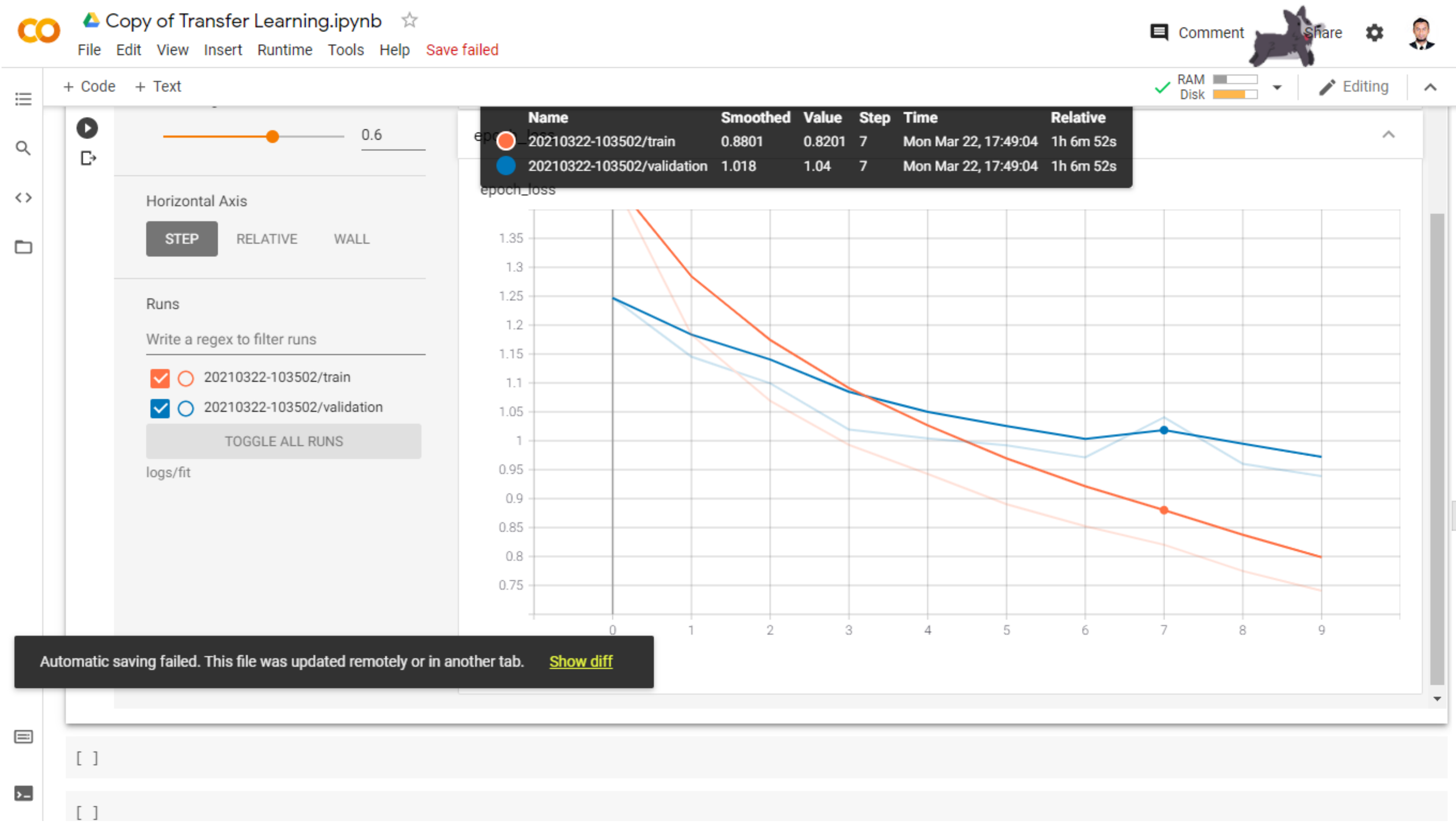
```
Epoch 1/10
0
Learning rate is : 0.0010000000474974513
1200/1200 [==============================] - 595s 495ms/step - loss: 1.6186 - accuracy: 0.4966 - val_loss: 1.2469 - va
l_accuracy: 0.6051

Epoch 00001: val_accuracy improved from -inf to 0.60510, saving model to model_save/weights-01-0.6051.hdf5
Epoch 2/10
1
Learning rate is : 0.0010000000474974513
1200/1200 [==============================] - 597s 497ms/step - loss: 1.2189 - accuracy: 0.6247 - val_loss: 1.1454 - va
l_accuracy: 0.6468

Epoch 00002: val_accuracy improved from 0.60510 to 0.64677, saving model to model_save/weights-02-0.6468.hdf5
Epoch 3/10
2
Learning rate is : 0.0010000000474974513
1200/1200 [==============================] - 583s 486ms/step - loss: 1.0793 - accuracy: 0.6675 - val_loss: 1.0993 - va
l_accuracy: 0.6648

Epoch 00003: val_accuracy improved from 0.64677 to 0.66479, saving model to model_save/weights-03-0.6648.hdf5
Epoch 4/10
3
Learning rate is : 0.0009500000160187483
1200/1200 [==============================] - 575s 479ms/step - loss: 0.9874 - accuracy: 0.6945 - val_loss: 1.0194 - va
l_accuracy: 0.6974

Epoch 00004: val_accuracy improved from 0.66479 to 0.69740, saving model to model_save/weights-04-0.6974.hdf5
```

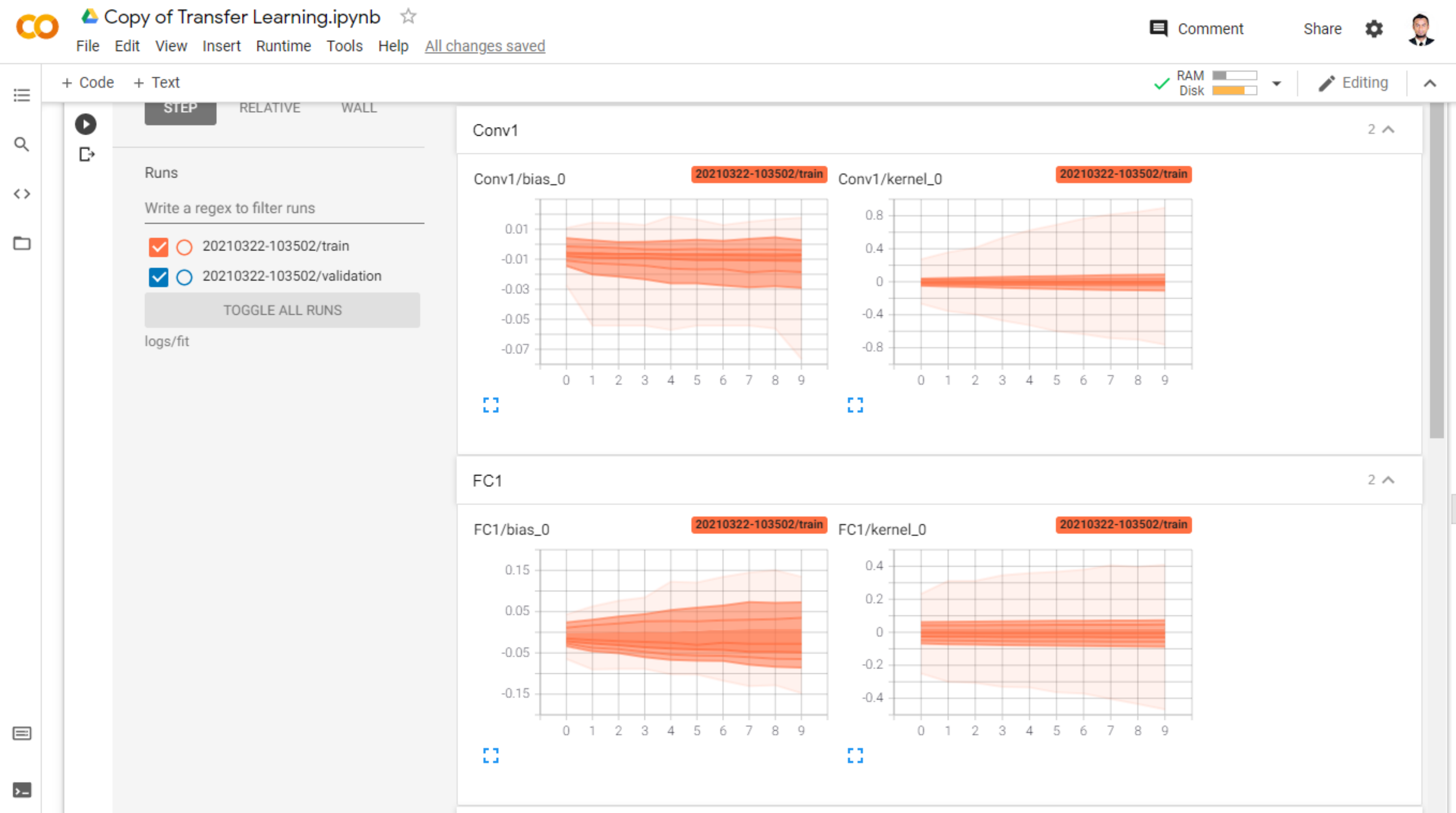
```
Epoch 5/10
4
Learning rate is : 0.0009500000160187483
1200/1200 [==============================] - 576s 480ms/step - loss: 0.9304 - accuracy: 0.7140 - val_loss: 1.0042 - va
l_accuracy: 0.7006

Epoch 00005: val_accuracy improved from 0.69740 to 0.70063, saving model to model_save/weights-05-0.7006.hdf5
Epoch 6/10
5
Learning rate is : 0.0009500000160187483
1200/1200 [==============================] - 569s 474ms/step - loss: 0.8877 - accuracy: 0.7273 - val_loss: 0.9920 - va
l_accuracy: 0.7134

Epoch 00006: val_accuracy improved from 0.70063 to 0.71344, saving model to model_save/weights-06-0.7134.hdf5
Epoch 7/10
6
Learning rate is : 0.0009025000035762787
1200/1200 [==============================] - 558s 465ms/step - loss: 0.8421 - accuracy: 0.7407 - val_loss: 0.9713 - va
l_accuracy: 0.7159

Epoch 00007: val_accuracy improved from 0.71344 to 0.71594, saving model to model_save/weights-07-0.7159.hdf5
Epoch 8/10
7
Learning rate is : 0.0009025000035762787
1200/1200 [==============================] - 546s 455ms/step - loss: 0.8108 - accuracy: 0.7478 - val_loss: 1.0405 - va
l_accuracy: 0.6938

Epoch 00008: val_accuracy did not improve from 0.71594
Epoch 9/10
8
Learning rate is : 0.000812250014860183
1200/1200 [==============================] - 548s 457ms/step - loss: 0.7658 - accuracy: 0.7648 - val_loss: 0.9602 - va
l_accuracy: 0.7210

Epoch 00009: val_accuracy improved from 0.71594 to 0.72104, saving model to model_save/weights-09-0.7210.hdf5
Epoch 10/10
9
Learning rate is : 0.0007716375403106213
1200/1200 [==============================] - 546s 455ms/step - loss: 0.7304 - accuracy: 0.7759 - val_loss: 0.9391 - va
l_accuracy: 0.7328

Epoch 00010: val_accuracy improved from 0.72104 to 0.73281, saving model to model_save/weights-10-0.7328.hdf5
```

Out[51]:
```
<tensorflow.python.keras.callbacks.History at 0x7f827c118dd0>
```

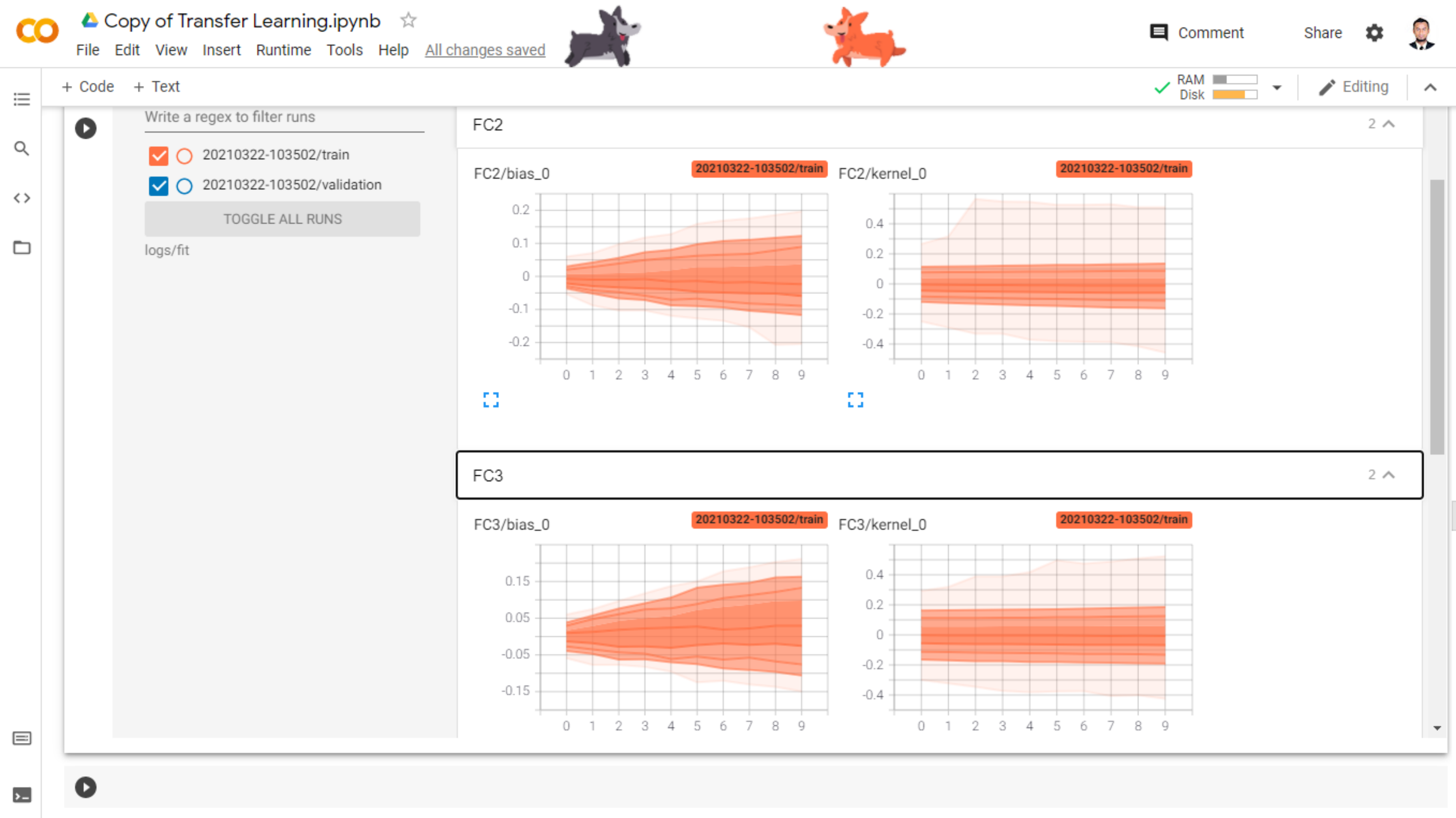
Tensorboard screenshot of Model-1


Copy of Transfer Learning.ipynb
File Edit View Insert Runtime Tools Help Save failed
Comment Share
+ Code + Text
RAM
Disk
Editing
Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing
0.6
Horizontal Axis
STEP RELATIVE WALL
Runs
Write a regex to filter runs
20210322-103502/train
20210322-103502/validation
TOGGLE ALL RUNS
logs/fit
epoch_accuracy
Smoothed Value Step Time Relative
rain 0.7548 0.773 9 Mon Mar 22, 18:07:21 1h 25m 9s
20210322-103502/validation 0.7183 0.7328 9 Mon Mar 22, 18:07:21 1h 25m 9s
Automatic saving failed. This file was updated remotely or in another tab. Show diff
epoch_loss

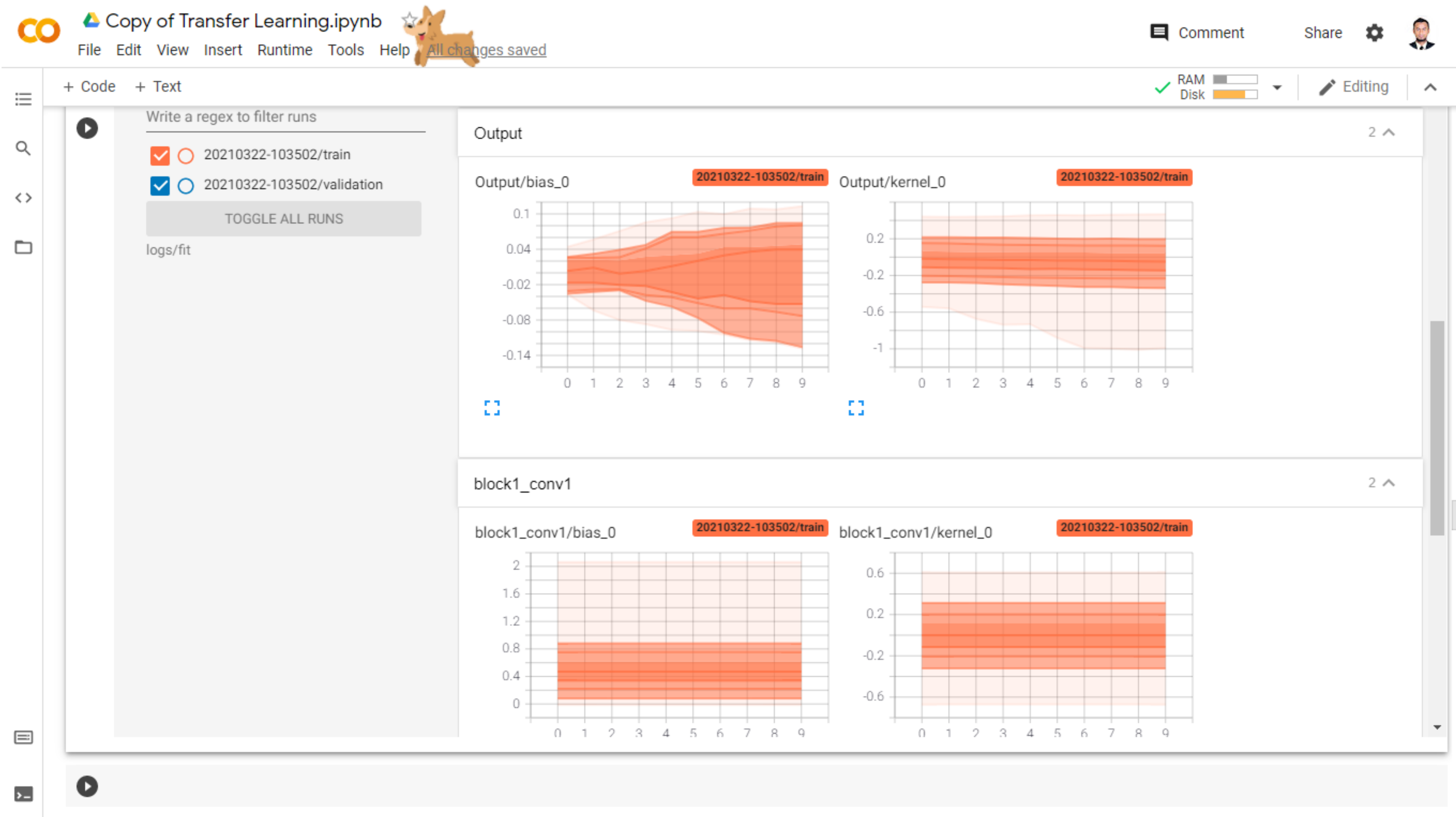

Copy of Transfer Learning.ipynb
File Edit View Insert Runtime Tools Help Save failed
Comment
Share
+ Code + Text
RAM
Disk
Editing
0.6
Horizontal Axis
STEP RELATIVE WALL
Runs
Write a regex to filter runs
20210322-103502/train
20210322-103502/validation
TOGGLE ALL RUNS
logs/fit
Name Smoothed Value Step Time Relative
20210322-103502/train 0.8801 0.8201 7 Mon Mar 22, 17:49:04 1h 6m 52s
20210322-103502/validation 1.018 1.04 7 Mon Mar 22, 17:49:04 1h 6m 52s
epoch_loss
1.35
1.3
1.25
1.2
1.15
1.1
1.05
1
0.95
0.9
0.85
0.8
0.75
0 1 2 3 4 5 6 7 8 9
Automatic saving failed. This file was updated remotely or in another tab. Show diff
[ ]
[ ]

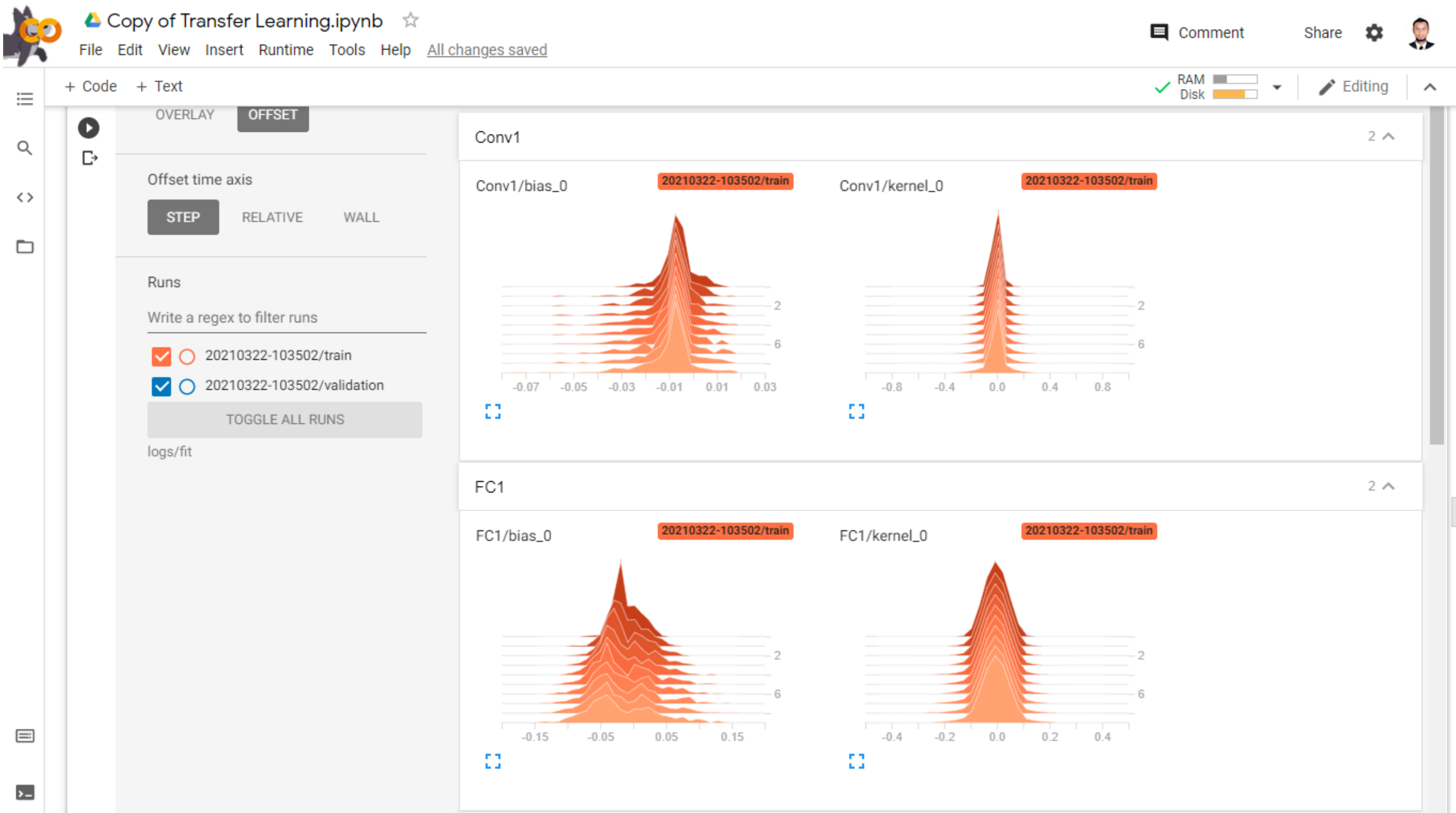


Copy of Transfer Learning.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
20210322-103502/train
20210322-103502/validation
TOGGLE ALL RUNS
logs/fit
Conv1
Conv1/bias_0
20210322-103502/train
Conv1/kernel_0
20210322-103502/train
FC1
FC1/bias_0
20210322-103502/train
FC1/kernel_0
20210322-103502/train

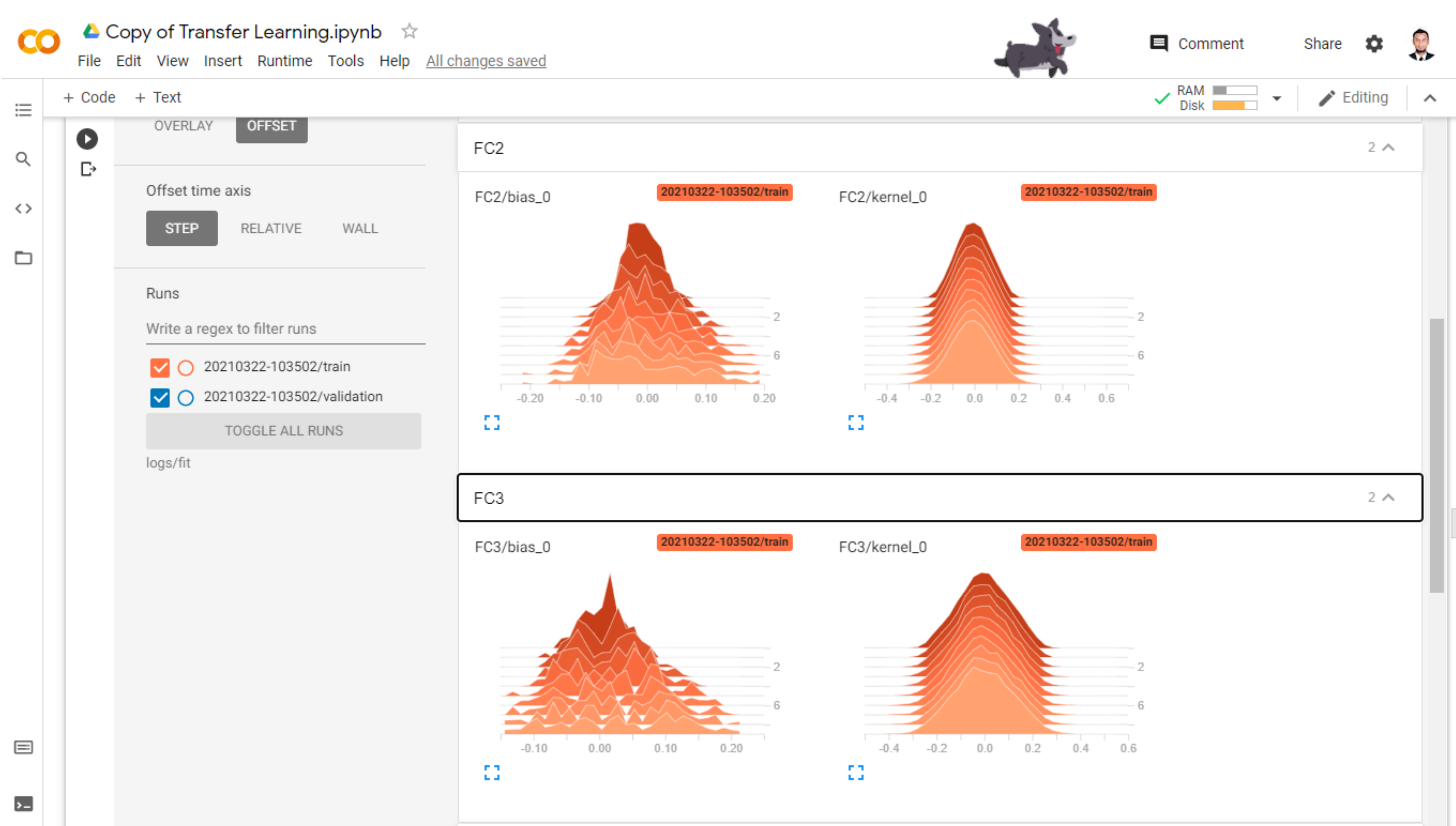


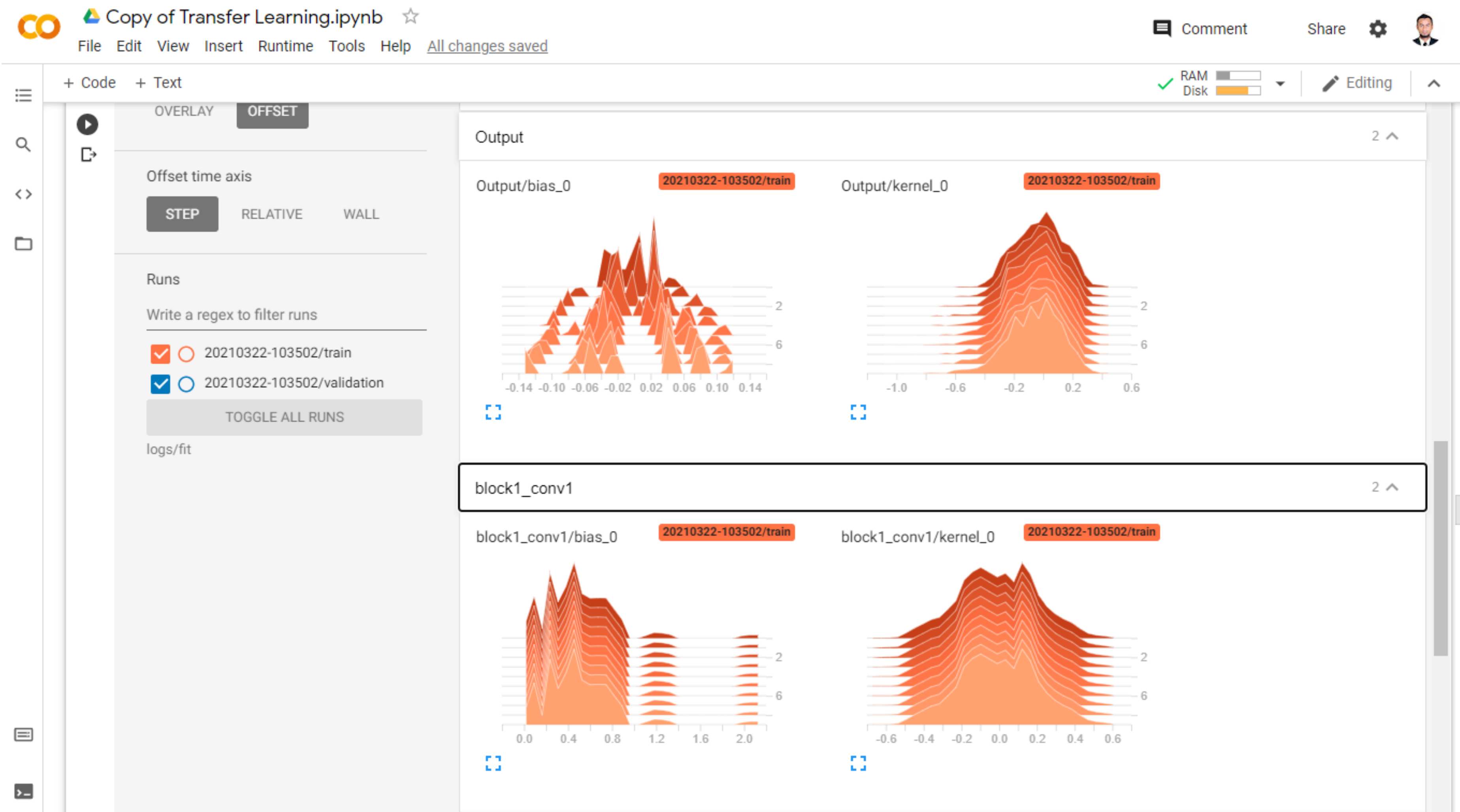
Copy of Transfer Learning.ipynb
File
Edit
View
Insert
Runtime
Tools
Help
All changes saved
Comment
Share
RAM
Disk
Editing
+ Code
+ Text
Write a regex to filter runs
20210322-103502/train
20210322-103502/validation
TOGGLE ALL RUNS
logs/fit
FC2
FC2/bias_0
20210322-103502/train
FC2/kernel_0
20210322-103502/train
FC3
FC3/bias_0
20210322-103502/train
FC3/kernel_0
20210322-103502/train


Copy of Transfer Learning.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
Write a regex to filter runs
20210322-103502/train
20210322-103502/validation
TOGGLE ALL RUNS
logs/fit
Output
Output/bias_0
20210322-103502/train
Output/kernel_0
20210322-103502/train
block1_conv1
block1_conv1/bias_0
20210322-103502/train
block1_conv1/kernel_0
20210322-103502/train



Copy of Transfer Learning.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
OVERLAY
OFFSET
Offset time axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
20210322-103502/train
20210322-103502/validation
TOGGLE ALL RUNS
logs/fit
Conv1
Conv1/bias_0
20210322-103502/train
Conv1/kernel_0
20210322-103502/train
FC1
FC1/bias_0
20210322-103502/train
FC1/kernel_0
20210322-103502/train


Copy of Transfer Learning.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
OVERLAY
OFFSET
Offset time axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
20210322-103502/train
20210322-103502/validation
TOGGLE ALL RUNS
logs/fit
FC2
FC2/bias_0
20210322-103502/train
FC2/kernel_0
20210322-103502/train
FC3
FC3/bias_0
20210322-103502/train
FC3/kernel_0
20210322-103502/train



In [51]:

In [52]:

```
%load_ext tensorboard
```

In [53]:

```
%tensorboard --logdir logs/fit
```

<IPython.core.display.Javascript object>

In [44]:

In [44]:

In [44]:

```

```

Model-2

1. Use VGG-16 (https://www.tensorflow.org/api_docs/python/tf/keras/applications/VGG16) pretrained network without Fully Connected layers and initilize all the weights with Imagenet trained weights.
2. After VGG-16 network without FC layers, don't use FC layers, use conv layers only as Fully connected layer. any FC layer can be converted to a CONV layer. This conversion will reduce the No of Trainable parameters in FC layers. For example, an FC layer with K=4096 that is looking at some input volume of size 7×7×512 can be equivalently expressed as a CONV layer with F=7,P=0,S=1,K=4096. In other words, we are setting the filter size to be exactly the size of the input volume, and hence the output will simply be 1×1×4096 since only a single depth column "fits" across the input volume, giving identical result as the initial FC layer. You can refer this (http://cs231n.github.io/convolutional-networks/#convert) link to better understanding of using Conv layer in place of fully connected layers.
3. Final architecture will be VGG-16 without FC layers(without top), 2 Conv layers identical to FC layers, 1 output layer for 16 class classification. **INPUT --> VGG-16 without Top layers(FC) --> 2 Conv Layers identical to FC --> Output Layer**
3. Train only last 2 Conv layers identical to FC layers, 1 output layer. Don't train the VGG-16 network.

In [84]:

```
# clear all previous sessions
tf.keras.backend.clear_session()
# initializing sequential model
model_2 = tf.keras.models.Sequential()
# load VGG16
model = tf.keras.applications.VGG16(
                          include_top = False,
                          weights = 'imagenet',
                          input_shape=(h,w,3)
                          )
# making all layers.trainable True
for layer in model.layers:
  layer.trainable = False

model_2.add(model) # adding VGG model to our Sequential model
model_2.add(Conv2D(filters=4096,
                         kernel_size=(7,7),
                         strides = (1,1),
                         padding='valid', # valid means no padding
                         data_format = 'channels_last',
                         activation='relu',
                         kernel_initializer=tf.keras.initializers.he_uniform(seed=42),
                         name = 'fc_Conv1'
                         #input_shape = model.output_shape[1:]
                        ))
model_2.add(Conv2D(
    filters=4096,
    kernel_size=(1,1),
    padding='valid',
    data_format='channels_last',
    activation='relu',
    kernel_initializer=tf.keras.initializers.he_uniform(seed=42),
    name='fc_conv2'
))
model_2.add(Flatten(data_format='channels_last',name='Flatten_last'))
model_2.add(Dense(units=16,
                  activation='softmax',
                  kernel_initializer=tf.keras.initializers.glorot_normal(seed=3), name='Output'
                  ))
```

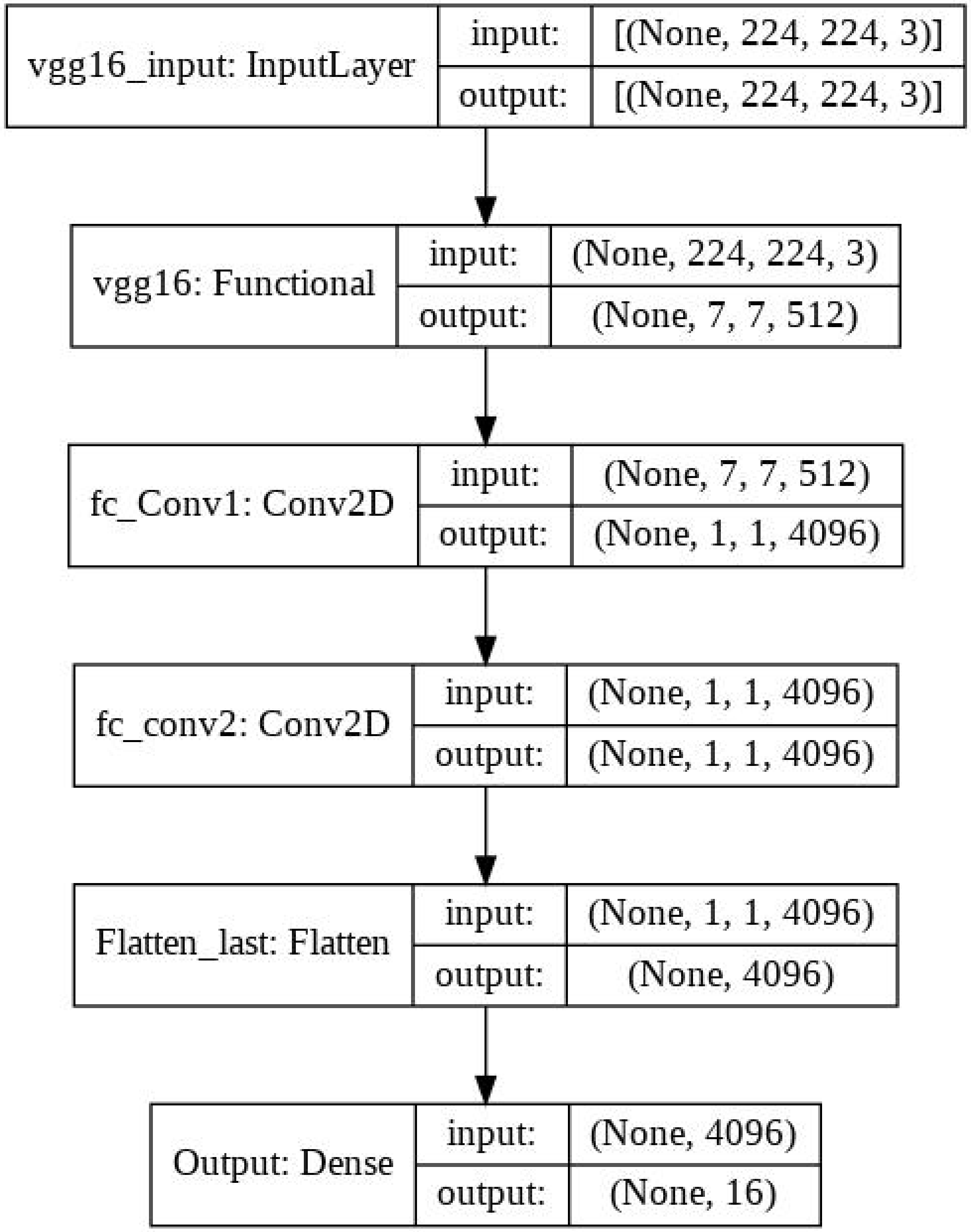
In [85]:
```
model_2.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 7, 7, 512)         14714688
_________________________________________________________________
fc_Conv1 (Conv2D)            (None, 1, 1, 4096)        102764544
_________________________________________________________________
fc_conv2 (Conv2D)            (None, 1, 1, 4096)        16781312
_________________________________________________________________
Flatten_last (Flatten)       (None, 4096)              0
_________________________________________________________________
Output (Dense)               (None, 16)                65552
=================================================================
Total params: 134,326,096
Trainable params: 119,611,408
Non-trainable params: 14,714,688
_________________________________________________________________
```

In [86]:

```
tf.keras.utils.plot_model(model_2, to_file='model1.jpeg',show_shapes=True)
```

Out[86]:



In [87]:

```
# train_df.head(32)
```

In [88]:
```
# remove previous logs
!rm -rf logs
# remove previous saved models
!rm -rf model_save
#compile the model
model_2.compile(optimizer=tf.keras.optimizers.Adam(),
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
#fit the model
model_2.fit(x = train_dataset,
              validation_data=validation_dataset,
              epochs=10,
              callbacks=[callback_list]
              )
```

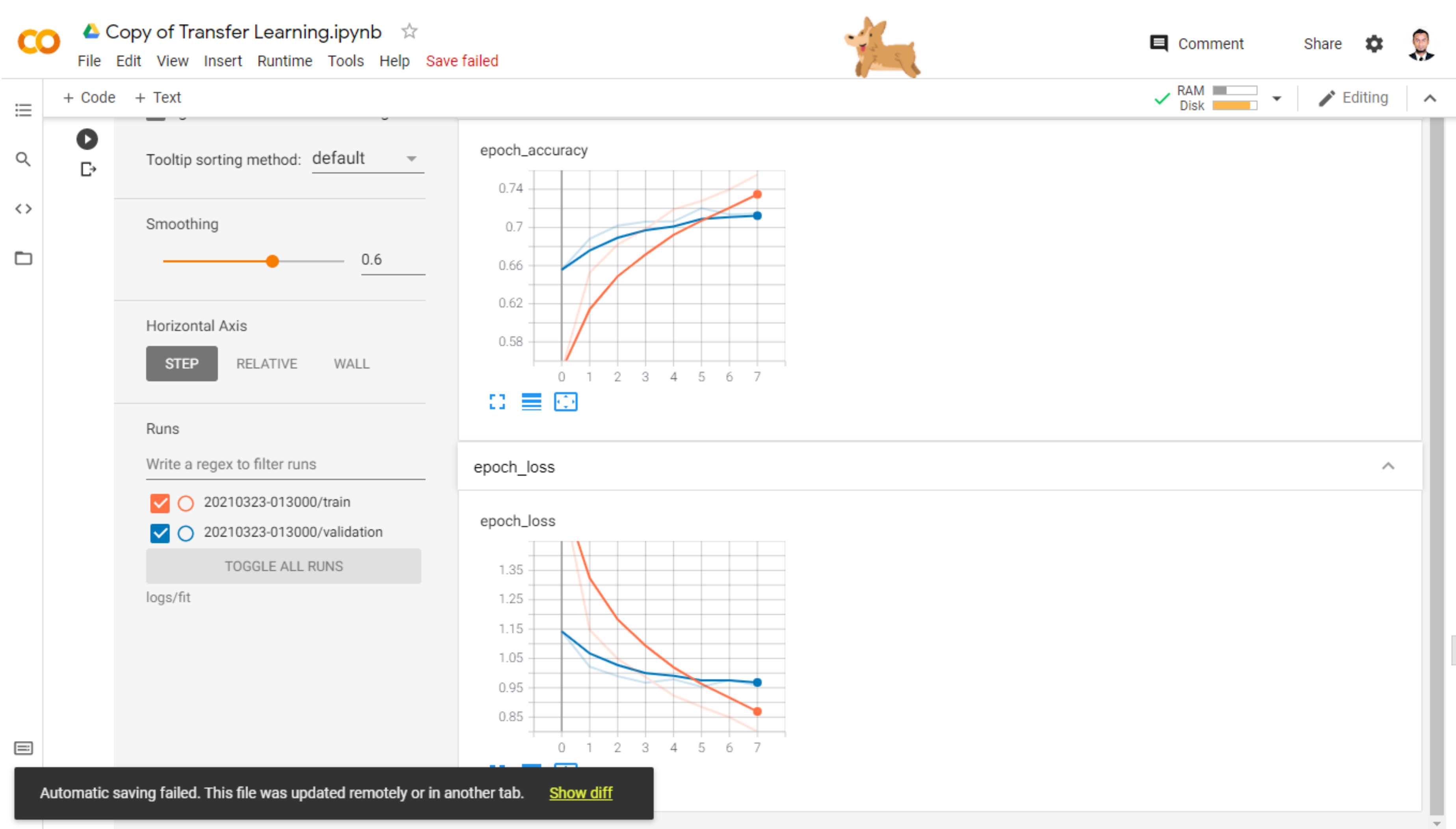
```
Epoch 1/10
0
Learning rate is : 0.0010000000474974513
1200/1200 [==============================] - 666s 554ms/step - loss: 2.6802 - accuracy: 0.4623 - val_loss: 1.1420 - val_
accuracy: 0.6556

Epoch 00001: val_accuracy improved from -inf to 0.65562, saving model to model_save/weights-01-0.6556.hdf5
Epoch 2/10
1
Learning rate is : 0.0010000000474974513
1200/1200 [==============================] - 677s 564ms/step - loss: 1.1569 - accuracy: 0.6468 - val_loss: 1.0220 - val_
accuracy: 0.6881

Epoch 00002: val_accuracy improved from 0.65562 to 0.68813, saving model to model_save/weights-02-0.6881.hdf5
Epoch 3/10
2
Learning rate is : 0.0010000000474974513
1200/1200 [==============================] - 671s 559ms/step - loss: 1.0431 - accuracy: 0.6815 - val_loss: 0.9893 - val_
accuracy: 0.7019

Epoch 00003: val_accuracy improved from 0.68813 to 0.70187, saving model to model_save/weights-03-0.7019.hdf5
Epoch 4/10
3
Learning rate is : 0.0009500000160187483
1200/1200 [==============================] - 674s 561ms/step - loss: 0.9872 - accuracy: 0.6946 - val_loss: 0.9673 - val_
accuracy: 0.7061

Epoch 00004: val_accuracy improved from 0.70187 to 0.70615, saving model to model_save/weights-04-0.7061.hdf5
Epoch 5/10
4
Learning rate is : 0.0009500000160187483
1200/1200 [==============================] - 674s 561ms/step - loss: 0.9195 - accuracy: 0.7198 - val_loss: 0.9786 - val_
```

```
accuracy: 0.7063

Epoch 00005: val_accuracy improved from 0.70615 to 0.70625, saving model to model_save/weights-05-0.7063.hdf5
Epoch 6/10
5
Learning rate is : 0.0009500000160187483
1200/1200 [==============================] - 675s 562ms/step - loss: 0.8758 - accuracy: 0.7335 - val_loss: 0.9532 - val_
accuracy: 0.7200

Epoch 00006: val_accuracy improved from 0.70625 to 0.72000, saving model to model_save/weights-06-0.7200.hdf5
Epoch 7/10
6
Learning rate is : 0.0009025000035762787
1200/1200 [==============================] - 665s 554ms/step - loss: 0.8424 - accuracy: 0.7395 - val_loss: 0.9762 - val_
accuracy: 0.7138

Epoch 00007: val_accuracy did not improve from 0.72000
Epoch 8/10
7
Learning rate is : 0.0008122500014860183
1200/1200 [==============================] - 648s 540ms/step - loss: 0.7893 - accuracy: 0.7574 - val_loss: 0.9574 - val_
accuracy: 0.7144

Epoch 00008: val_accuracy did not improve from 0.72000
```

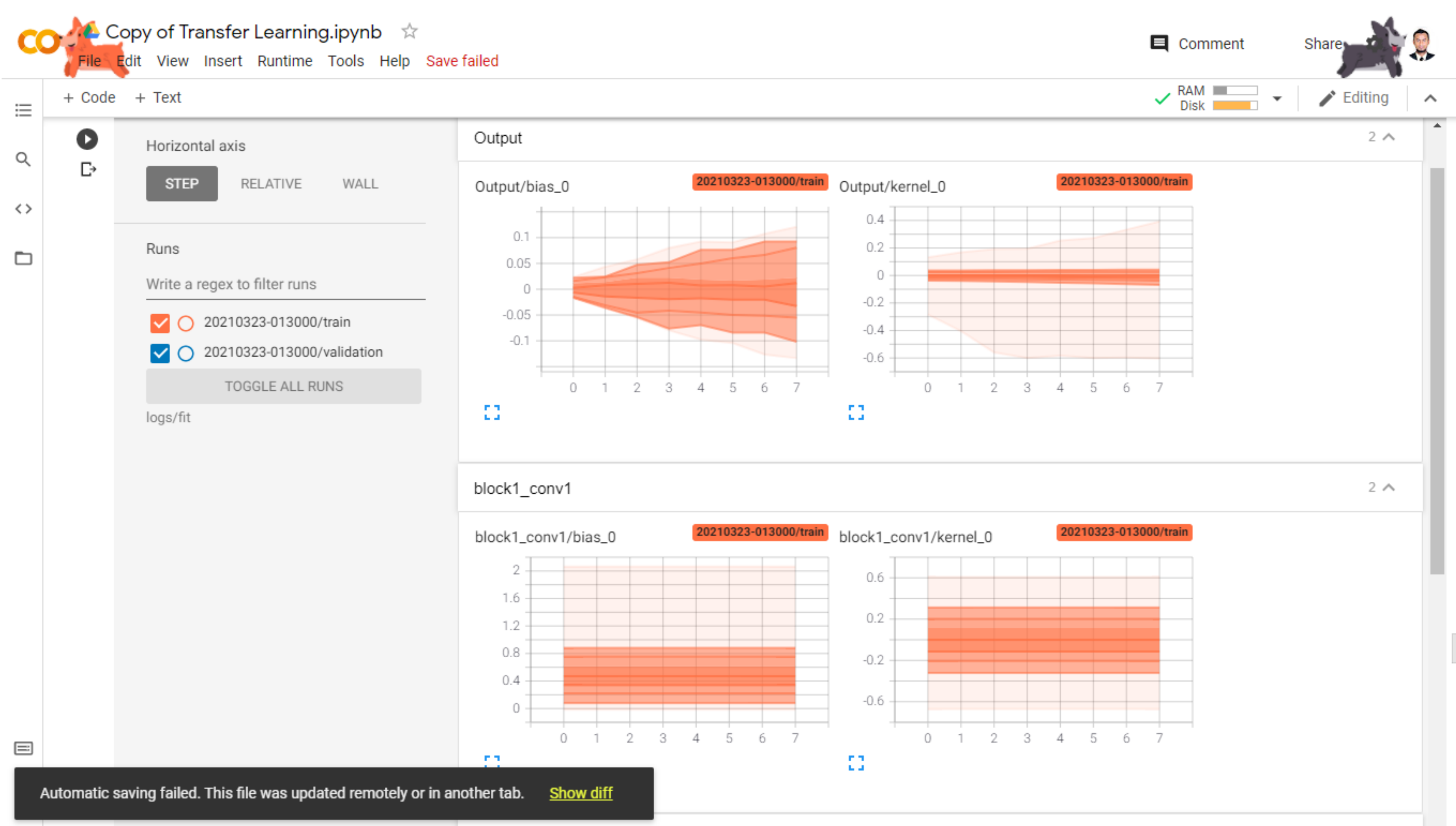
Out[88]:

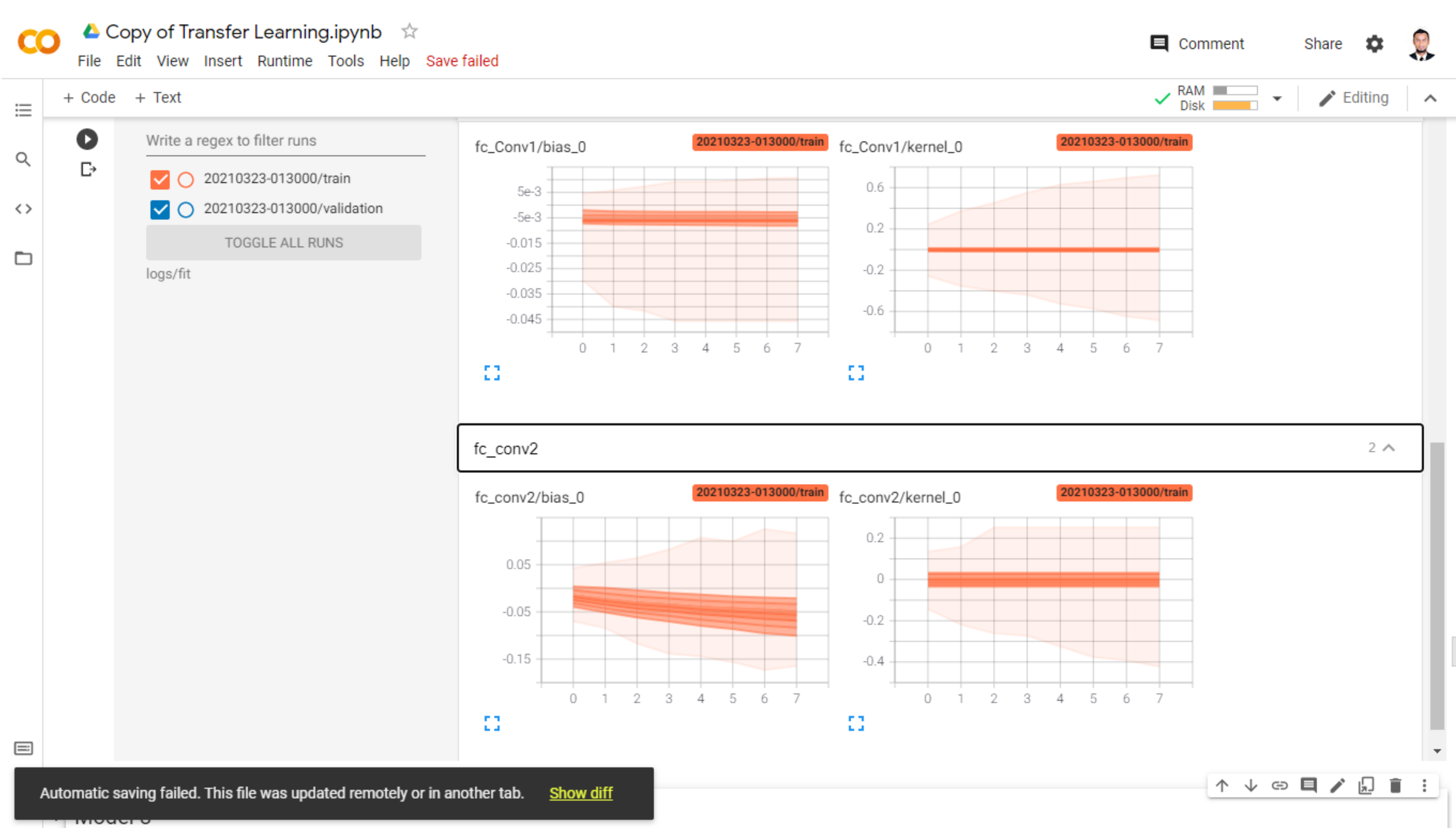
```
<tensorflow.python.keras.callbacks.History at 0x7ff281687750>
```

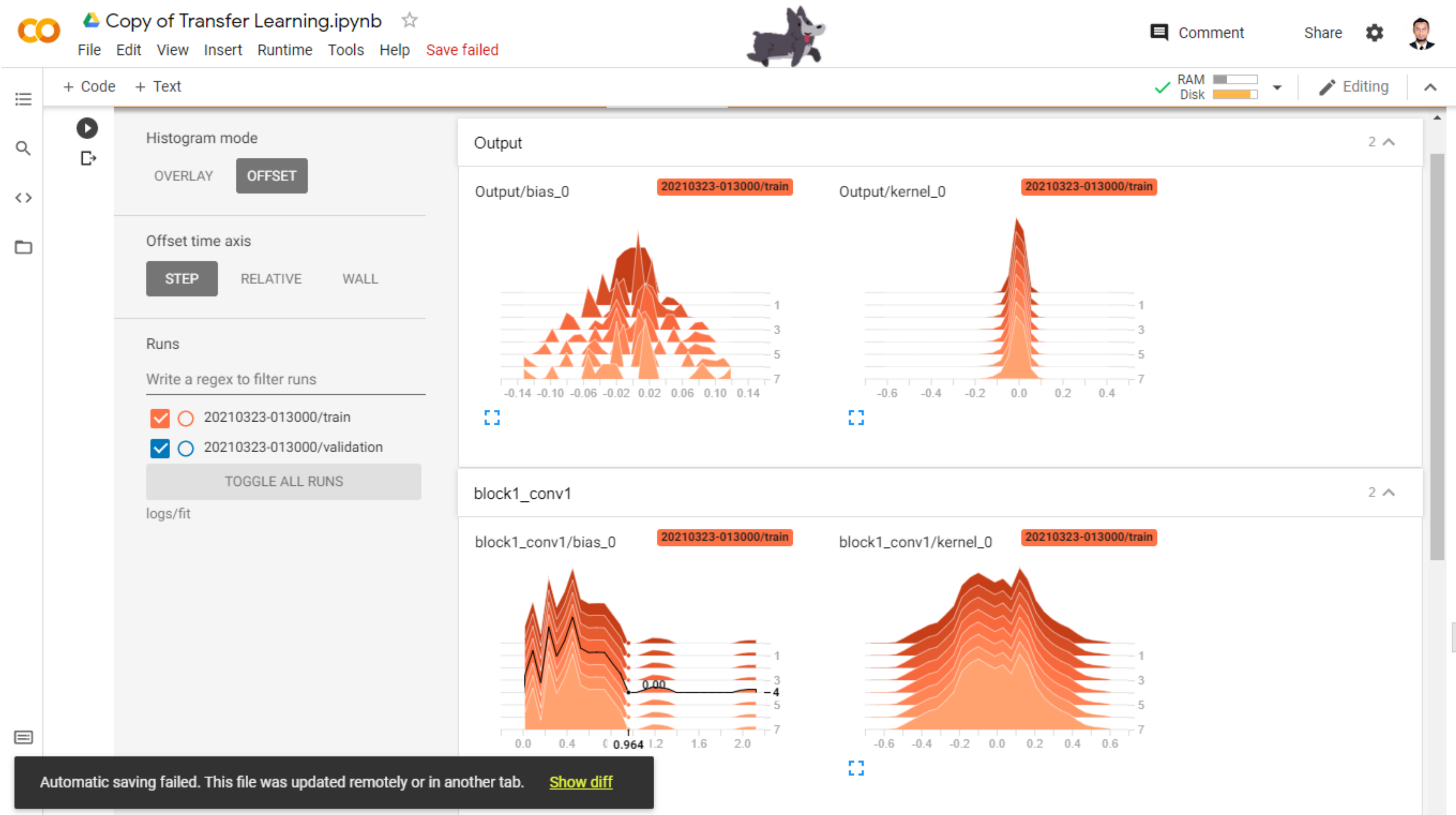
Tensorboard screenshots of Model-2

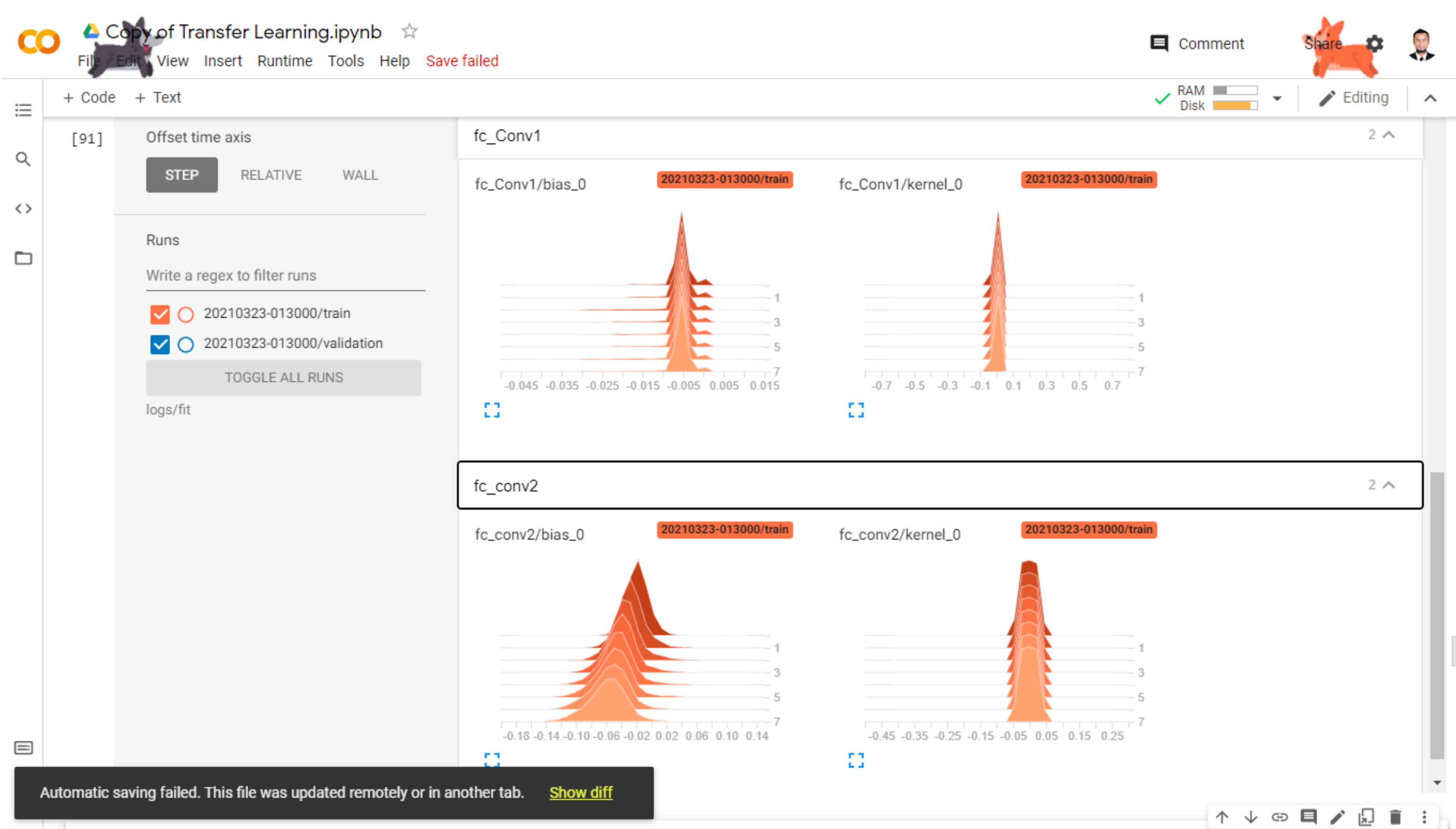


Copy of Transfer Learning.ipynb
File Edit View Insert Runtime Tools Help Save failed
Comment
Share
+ Code + Text
RAM
Disk
Editing
Tooltip sorting method: default
Smoothing
0.6
Horizontal Axis
STEP RELATIVE WALL
Runs
Write a regex to filter runs
20210323-013000/train
20210323-013000/validation
TOGGLE ALL RUNS
logs/fit
epoch_accuracy
0.74
0.7
0.66
0.62
0.58
0 1 2 3 4 5 6 7
epoch_loss
epoch_loss
1.35
1.25
1.15
1.05
0.95
0.85
0 1 2 3 4 5 6 7
Automatic saving failed. This file was updated remotely or in another tab. Show diff


Copy of Transfer Learning.ipynb
File Edit View Insert Runtime Tools Help Save failed
Comment
Share
+ Code + Text
RAM
Disk
Editing
Horizontal axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
20210323-013000/train
20210323-013000/validation
TOGGLE ALL RUNS
logs/fit
Output
Output/bias_0
20210323-013000/train
Output/kernel_0
20210323-013000/train
block1_conv1
block1_conv1/bias_0
20210323-013000/train
block1_conv1/kernel_0
20210323-013000/train
Automatic saving failed. This file was updated remotely or in another tab.
Show diff


Copy of Transfer Learning.ipynb
File Edit View Insert Runtime Tools Help Save failed
Comment
Share
+ Code + Text
RAM
Disk
Editing
Write a regex to filter runs
20210323-013000/train
20210323-013000/validation
TOGGLE ALL RUNS
logs/fit
fc_Conv1/bias_0
20210323-013000/train
fc_Conv1/kernel_0
20210323-013000/train
fc_conv2
fc_conv2/bias_0
20210323-013000/train
fc_conv2/kernel_0
20210323-013000/train
Automatic saving failed. This file was updated remotely or in another tab.
Show diff


Copy of Transfer Learning.ipynb
File Edit View Insert Runtime Tools Help Save failed
Comment
Share
+ Code + Text
RAM
Disk
Editing
Histogram mode
OVERLAY
OFFSET
Offset time axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
20210323-013000/train
20210323-013000/validation
TOGGLE ALL RUNS
logs/fit
Output
Output/bias_0
20210323-013000/train
Output/kernel_0
20210323-013000/train
block1_conv1
block1_conv1/bias_0
20210323-013000/train
block1_conv1/kernel_0
20210323-013000/train
Automatic saving failed. This file was updated remotely or in another tab.
Show diff



In [ ]:

In [90]:
```
%load_ext tensorboard
```

In [91]:
```
%tensorboard --logdir logs/fit
```

```
<IPython.core.display.Javascript object>
```

Model-3

1. Use same network as Model-2 'INPUT --> VGG-16 without Top layers(FC) --> 2 Conv Layers identical to FC --> Out

put Layer' and train only Last 6 Layers of VGG-16 network, 2 Conv layers identical to FC layers, 1 output layer.

In [120]:

```python
model = tf.keras.applications.VGG16(
                          include_top = False,
                          weights = 'imagenet',
                          input_shape=(h,w,3)
                          )
# making last 6 layers.trainable True
print(len(model.layers)-5)
for i in range(len(model.layers)-5):
  layer = model.layers[i]
  layer.trainable = False
model.summary()
```

```
14
Model: "vgg16"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
input_3 (InputLayer)        [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0
_________________________________________________________________
```

```
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 7,079,424
Non-trainable params: 7,635,264
_________________________________________________________________
```

In [120]:

In [120]:

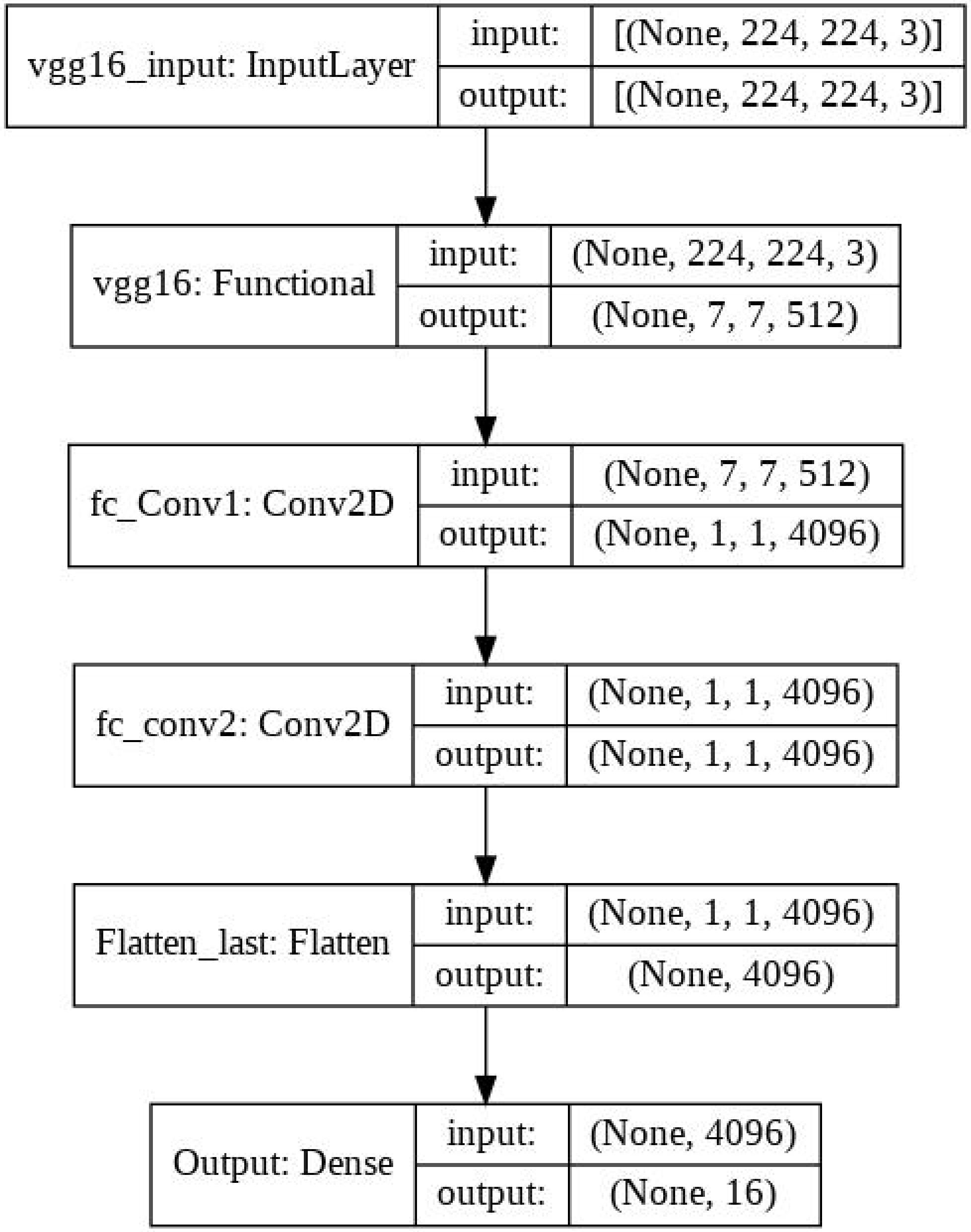
In [92]:

```python
# clear all previous sessions
tf.keras.backend.clear_session()
# initializing sequential model
top_model = tf.keras.models.Sequential()
# load VGG16
model = tf.keras.applications.VGG16(
                          include_top = False,
                          weights = 'imagenet',
                          input_shape=(h,w,3)
                          )
# making last 6 layers.trainable True
for i in range(len(model.layers)-5):
  layer = model.layers[i]
  layer.trainable = False

top_model.add(model) # adding VGG model to our Sequential model
top_model.add(Conv2D(filters=4096,
                         kernel_size=(7,7),
                         strides = (1,1),
                         padding='valid', # valid means no padding
                         data_format = 'channels_last',
                         activation=tf.keras.layers.LeakyReLU(),
                         kernel_initializer=tf.keras.initializers.he_normal(seed=0),
                         name = 'fc_Conv1'
                         #input_shape = model.output_shape[1:]
                        ))
top_model.add(Conv2D(
    filters=4096,
    kernel_size=(1,1),
    padding='valid',
    data_format='channels_last',
    activation=tf.keras.layers.LeakyReLU(),
    kernel_initializer=tf.keras.initializers.he_normal(seed=0),
    name='fc_conv2'
))
top_model.add(Flatten(data_format='channels_last',name='Flatten_last'))
top_model.add(Dense(units=16,
                  activation='softmax',
                  kernel_initializer=tf.keras.initializers.glorot_normal(seed=3), name='Output'
                  ))
```

In [93]:
```
top_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 7, 7, 512)         14714688
_________________________________________________________________
fc_Conv1 (Conv2D)            (None, 1, 1, 4096)        102764544
_________________________________________________________________
fc_conv2 (Conv2D)            (None, 1, 1, 4096)        16781312
_________________________________________________________________
Flatten_last (Flatten)       (None, 4096)              0
_________________________________________________________________
Output (Dense)               (None, 16)                65552
=================================================================
Total params: 134,326,096
Trainable params: 126,690,832
Non-trainable params: 7,635,264
_________________________________________________________________
```

In [95]:
```
tf.keras.utils.plot_model(top_model, to_file='model3.jpeg',show_shapes=True)
```
Out[95]:

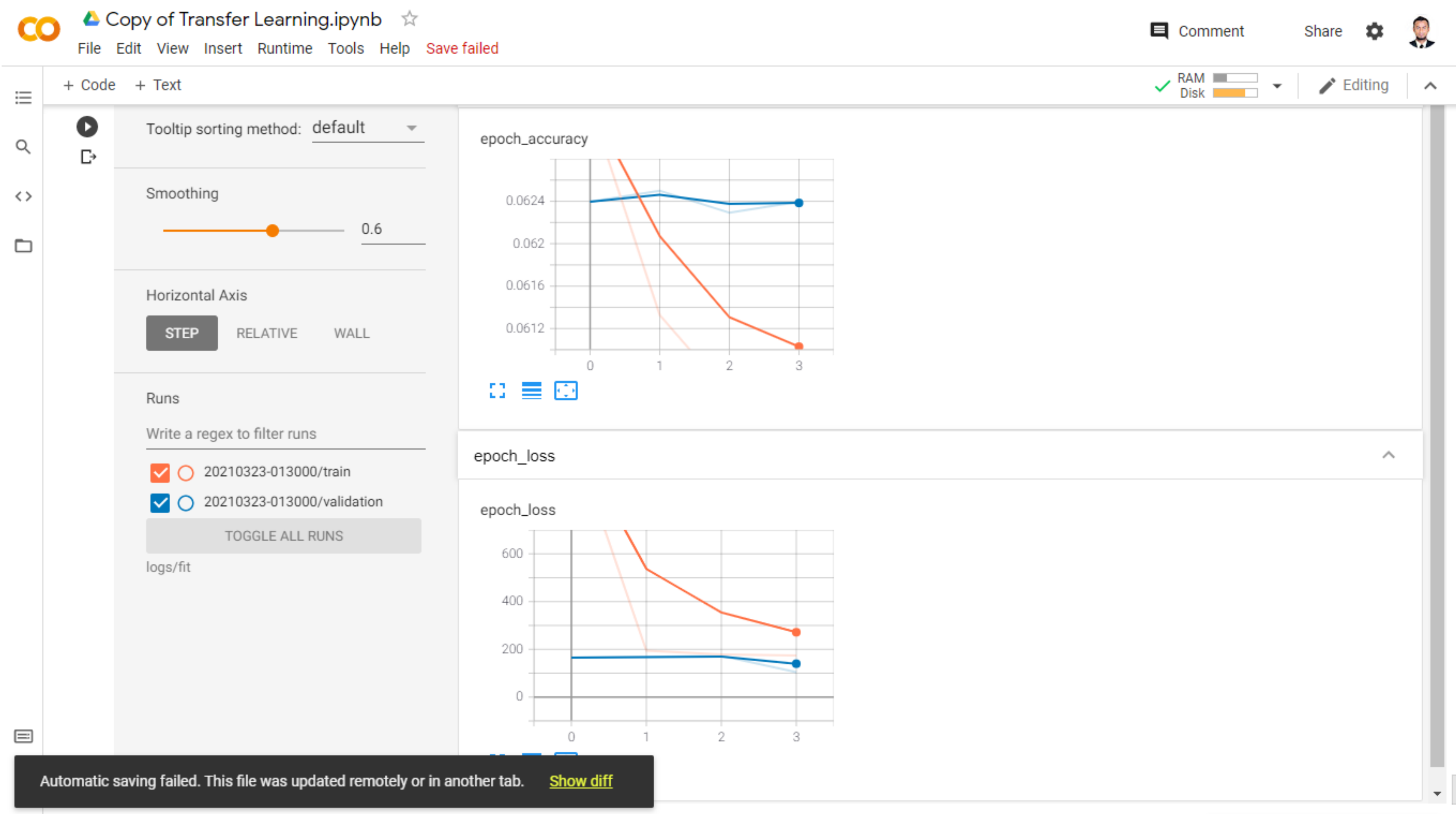


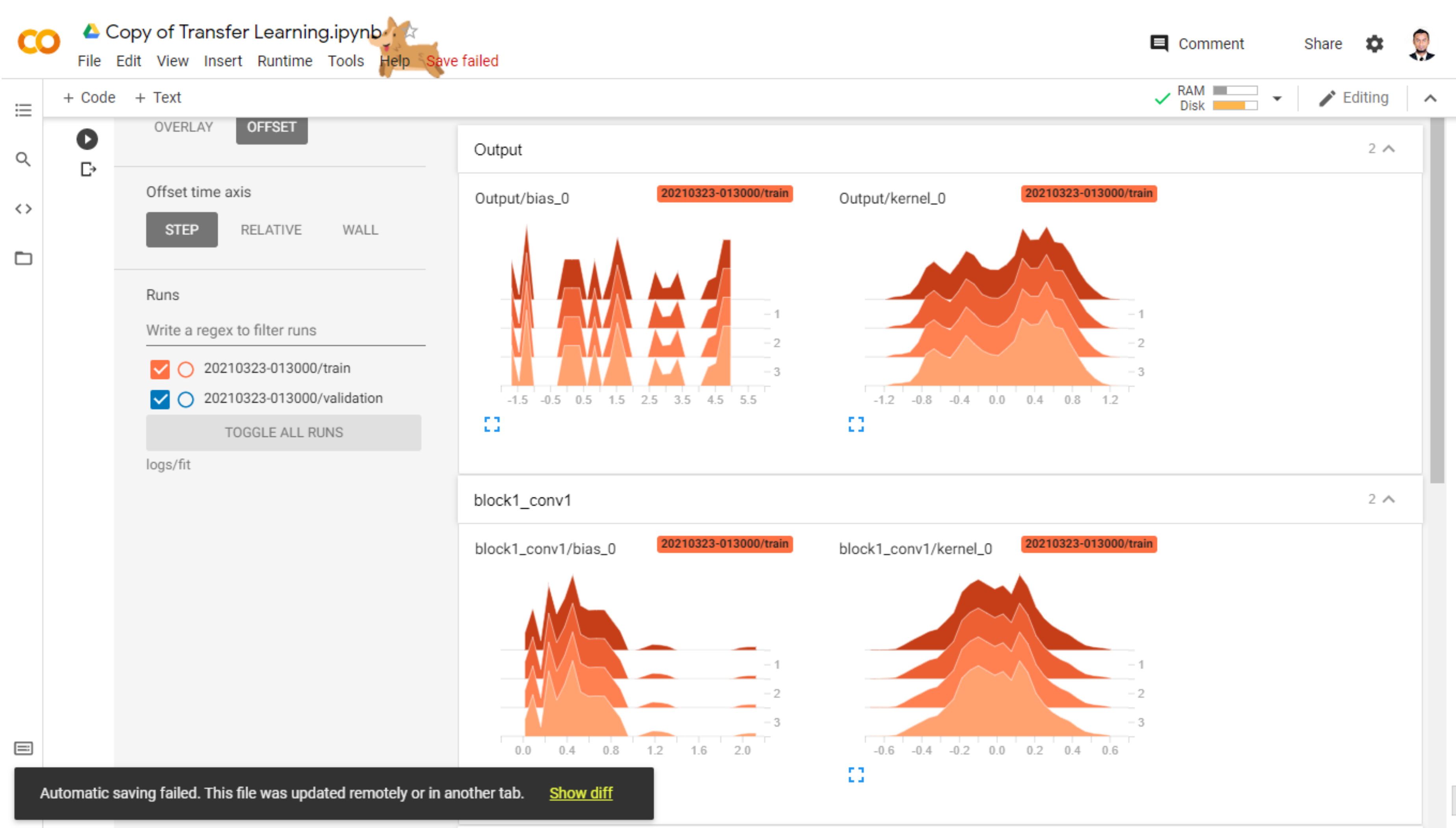
In [97]:
```python
# remove previous logs
!rm -rf logs
# remove previous saved models
!rm -rf model_save
#compile the model
top_model.compile(optimizer=tf.keras.optimizers.Adam(),
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

#fit the model
top_model.fit(x = train_dataset,
              validation_data=validation_dataset,
              epochs=10,
              callbacks=[callback_list]
              )
```

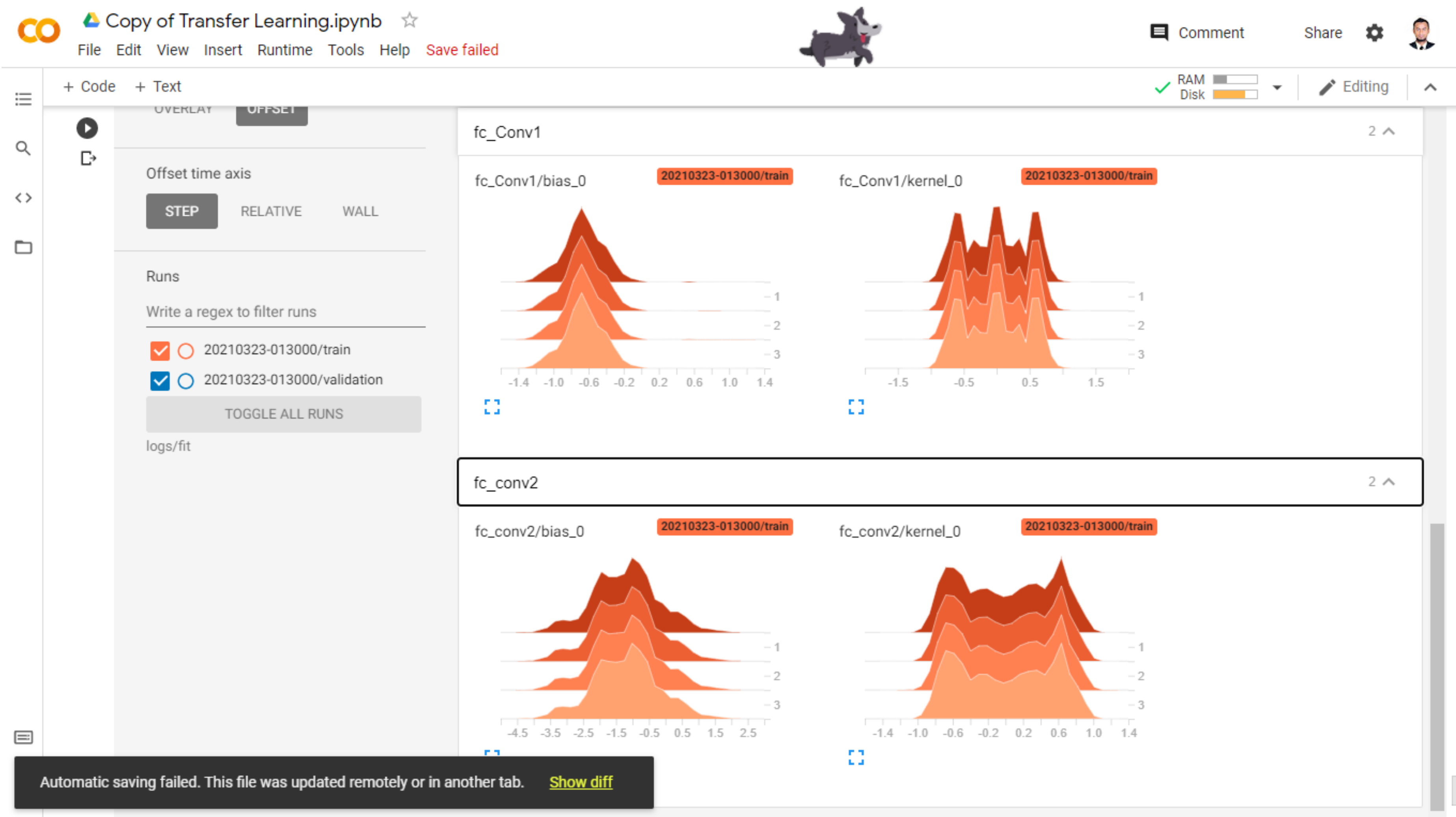
```
Epoch 1/10
0
Learning rate is : 0.0010000000474974513
1200/1200 [==============================] - 736s 612ms/step - loss: 1295.5925 - accuracy: 0.0628 - val_loss: 165.3107 - val_accuracy: 0.0624

Epoch 00001: val_accuracy did not improve from 0.72000
Epoch 2/10
1
Learning rate is : 0.0010000000474974513
1200/1200 [==============================] - 723s 602ms/step - loss: 192.8512 - accuracy: 0.0612 - val_loss: 168.8053 - val_accuracy: 0.0625

Epoch 00002: val_accuracy did not improve from 0.72000
Epoch 3/10
2
Learning rate is : 0.0010000000474974513
1200/1200 [==============================] - 721s 600ms/step - loss: 174.8353 - accuracy: 0.0615 - val_loss: 172.5757 - val_accuracy: 0.0623

Epoch 00003: val_accuracy did not improve from 0.72000
Epoch 4/10
3
Learning rate is : 0.0008549999911338091
1200/1200 [==============================] - 717s 597ms/step - loss: 182.1918 - accuracy: 0.0598 - val_loss: 104.1381 - val_accuracy: 0.0624

Epoch 00004: val_accuracy did not improve from 0.72000
```

Out[97]:
```
<tensorflow.python.keras.callbacks.History at 0x7ff2204ba790>
```

Tensorboard screenshots of Model-3




Copy of Transfer Learning.ipynb
File Edit View Insert Runtime Tools Help Save failed
Comment
Share
+ Code + Text
RAM
Disk
Editing
OVERLAY
OFFSET
Offset time axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
20210323-013000/train
20210323-013000/validation
TOGGLE ALL RUNS
logs/fit
Output
Output/bias_0
20210323-013000/train
Output/kernel_0
20210323-013000/train
block1_conv1
block1_conv1/bias_0
20210323-013000/train
block1_conv1/kernel_0
20210323-013000/train
Automatic saving failed. This file was updated remotely or in another tab.
Show diff



In [ ]:

In [98]:
```
%tensorboard --logdir logs/fit
```

```
<IPython.core.display.Javascript object>
```

In [ ]:

Result:

Model -- Accuracy(%)

Model-1 ==================== 73.28

Model-2 ==================== 72.00

Model-3 ==================== 06.25

```
In [ ]:
```