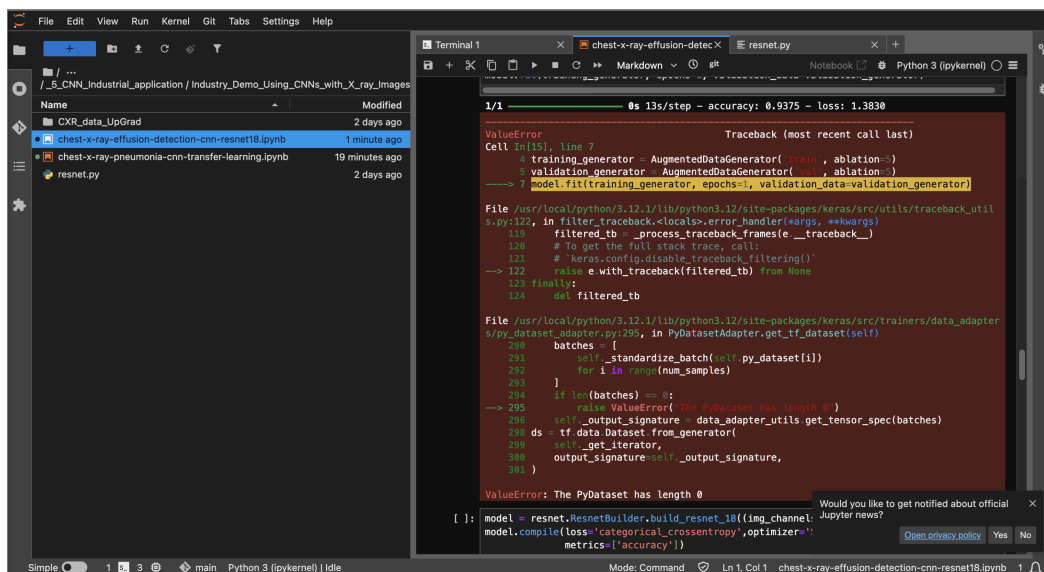


Pending items from Upgrad repo

As temporary workarounds, we took the following files / folders from existing code

-2-Exam-2

4. too many errors



The screenshot shows a Jupyter Notebook with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with files like 'CXR_data_UpGrad', 'chest-x-ray-effusion-detection-cnn-resnet18.ipynb', 'chest-x-ray-pneumonia-cnn-transfer-learning.ipynb', and 'resnet.py'. The code editor shows a Keras model training process. A red error message is displayed, indicating a 'ValueError: The PyDataset has length 0'. The error message includes a traceback showing the call to 'model.fit' and the subsequent 'PyDatasetAdapter.get_tf_dataset' method. A small dialog box at the bottom right asks 'Would you like to get notified about official Jupyter news?' with 'Open privacy policy', 'Yes', and 'No' buttons.

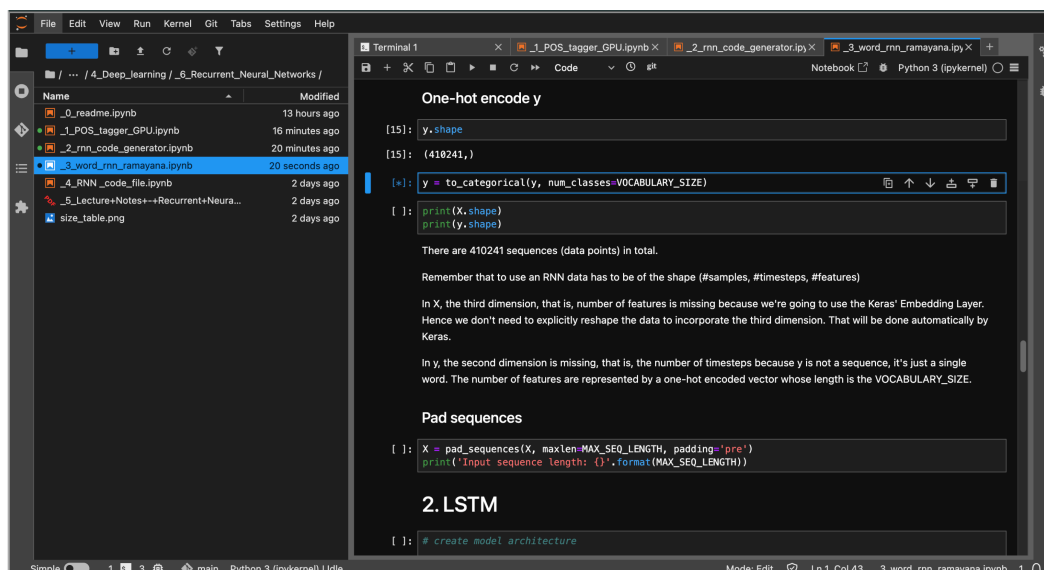
```
1/1 0s 13s/step - accuracy: 0.9375 - loss: 1.3830
ValueError                                Traceback (most recent call last)
Cell In[15], line 7
      4 training_generator = AugmentedDataGenerator(train, ablation=0)
      5 validation_generator = AugmentedDataGenerator(val, ablation=0)
----> 7 model.fit(training_generator, epochs=1, validation_data=validation_generator)

File /usr/local/python/3.12.1/lib/python3.12/site-packages/keras/src/utils/traceback_util
s.py:122, in filter_traceback.<locals>.error_handler(*args, **kwargs)
    119 filtered_tb = _process_traceback_frames(e.__traceback__)
    120 # to get the full stack trace, call:
    121 # `keras.config.disable_traceback_filtering()`
--> 122 raise e.with_traceback(filtered_tb) from None
    123 finally:
    124     del filtered_tb

File /usr/local/python/3.12.1/lib/python3.12/site-packages/keras/src/trainers/data_adapter
s/py_dataset_adapter.py:295, in PyDatasetAdapter.get_tf_dataset(self)
    290 batches = []
    291     self._standardize_batch(self.py_dataset[i])
    292     for i in range(num_samples)
    293 ]
    294 if len(batches) == 0:
--> 295     raise ValueError('The PyDataset has length 0')
    296 self._output_signature = data_adapter_utils.get_tensor_spec(batches)
    297 ds = tf.data.Dataset.from_generator(
    298     self._get_iterator,
    299     output_signature=self._output_signature,
    300 )
    301 )

ValueError: The PyDataset has length 0
Would you like to get notified about official Jupyter news?
[ ]: model = resnet.ResnetBuilder.build_resnet_18(img_channel=
model.compile(loss='categorical_crossentropy', optimizer='
metrics=['accuracy'])
```

8. Memory issue (MemoryError: Unable to allocate 54.0 GiB for an array with shape (410241, 17667) and data type float64)



The screenshot shows a Jupyter Notebook with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with files like '_0_readme.ipynb', '_1_POS_tagger_GPU.ipynb', '_2_rnn_code_generator.ipynb', and '_3_word_rnn_ramayana.ipynb'. The code editor shows a notebook titled 'One-hot encode y'. The code defines a function to convert a list of sequences into a one-hot encoded matrix. The output shows the shape of the matrix as (410241,). The notebook text explains that the third dimension (number of features) is missing and needs to be added. It also mentions that the second dimension (number of timesteps) is missing and needs to be added. The notebook concludes with a section titled '2. LSTM' and a comment to create the model architecture.

```
[15]: y.shape
[15]: (410241,)

[ ]: y = to_categorical(y, num_classes=VOCABULARY_SIZE)
[ ]: print(X.shape)
[ ]: print(y.shape)

There are 410241 sequences (data points) in total.

Remember that to use an RNN data has to be of the shape (#samples, #timesteps, #features)

In X, the third dimension, that is, number of features is missing because we're going to use the Keras' Embedding Layer.
Hence we don't need to explicitly reshape the data to incorporate the third dimension. That will be done automatically by Keras.

In y, the second dimension is missing, that is, the number of timesteps because y is not a sequence, it's just a single word.
The number of features are represented by a one-hot encoded vector whose length is the VOCABULARY_SIZE.

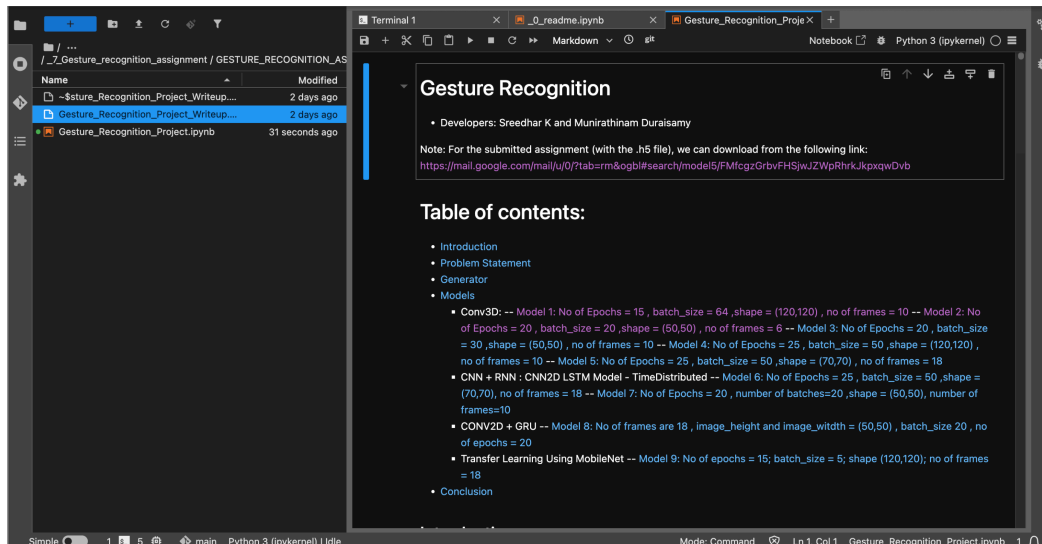
Pad sequences

[ ]: X = pad_sequences(X, maxlen=MAX_SEQ_LENGTH, padding='pre')
[ ]: print('Input sequence length: {}'.format(MAX_SEQ_LENGTH))

2. LSTM

[ ]: # create model architecture
```

9. Training and testing datasets not found

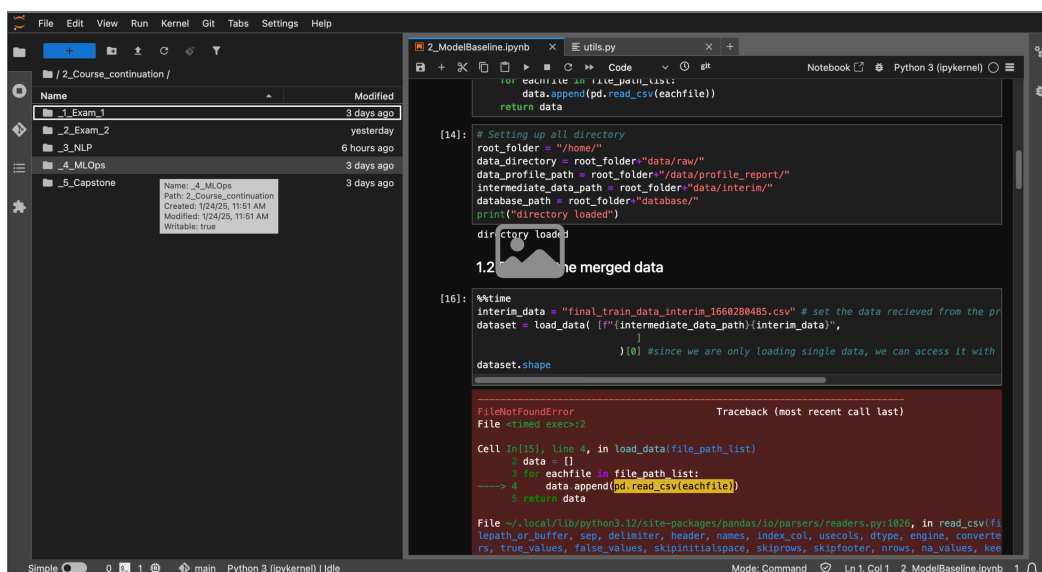


a

11.

1. Telecom Churn folder
2. **MLOps folder**
3. Individual files from above

4. MLOps (can be done only in Jarvis)



```
=====
==
=====
=====
==
```

git add . && git commit -m "C" && git push origin main