# Code Snippets

## Seed to ensure reproducibility

| tf.random.set\_seed(42) # extra code - ensures reproducibility |
| --- |

## 

## Link to Git Repository:

| cd /Users/shuchimishra/Library/CloudStorage/GoogleDrive-shuchimishra@g.ucla.edu/My\ Drive/Data\ Science\ \&\ Machine\ Learning |
| --- |

## GitSearching:

<https://git-scm.com/book/en/v2/Git-Tools-Searching>

| cd /Users/shuchimishra/Library/CloudStorage/GoogleDrive-shuchimishra@g.ucla.edu/My\ Drive/Data\ Science\ \&\ Machine\ Learning\Tensorflow\Certification/Repository/Tensorflow\_projects  git grep -n copyfile |
| --- |

# 

## Download files from Google colab:

| *# Import files utilities in Colab* try:  from google.colab import files except ImportError:  pass  *# Download the files* else:  files.download('vecs.tsv')  files.download('meta.tsv') |
| --- |

## 

## Snippet to read csv file

| def parse\_data\_from\_input\_v1(filename):  ""*"*  Parses the images and labels from a CSV file   Args:  filename (string): path to the CSV file   Returns:  images, labels: tuple of numpy arrays containing the images and labels  ""*"*  with open(filename) as file:  ### START CODE HERE  # Use csv.reader, passing in the appropriate delimiter  # Remember that csv.reader can be iterated and returns one line in each iteration  csv\_reader = csv.reader(file, delimiter=',')   next(csv\_reader, None) # skip the headers  label = []  image = []   for row in csv\_reader:  label\_col = str(row[0])  label.append(label\_col)   #Parsing pixel values of image  pixel\_value = row[1:]  image.append(list(np.reshape(pixel\_value, (28,28))))   #Convert list to numpy array  labels = np.asarray(label, dtype=np.float32)  images = np.asarray(image, dtype=np.float32)   ### END CODE HERE   return images, labels |
| --- |

## 

## Snippet to read json file

| Import json  filename = './sarcasm.json'  *## Opening JSON file* f = open(filename)  *# Load the json file* datastore = json.load(f)  *# Initializing the lists* urls, texts, labels = [],[],[]  *#Iterating through json data* for row in datastore:  urls.append(row['article\_link'])  texts.append(row['headline'])  labels.append(row['is\_sarcastic'])  *# Closing file* f.close() |
| --- |

## 

## Retrieve image,labels in dataset with metadata:

## 

| get\_label\_name = metadata.features['label'].int2str  image, label = next(iter(train\_ds)) \_ = plt.imshow(image) \_ = plt.title(get\_label\_name(label)) |
| --- |

## 

## Plot accuracy and loss graphs

| import matplotlib.pyplot as plt %matplotlib inline %config InlineBackend.fugure\_format = 'retina' def plot\_loss\_acc(history):  *#-----------------------------------------------------------*  *# Retrieve a list of list results on training and test data*  *# sets for each training epoch*  *#-----------------------------------------------------------*  acc = history.history[ 'accuracy' ]  val\_acc = history.history[ 'val\_accuracy' ]  loss = history.history[ 'loss' ]  val\_loss = history.history['val\_loss' ]  epochs = range(len(acc)) *# Get number of epochs*  *#------------------------------------------------*  *# Plot training and validation accuracy per epoch*  *#------------------------------------------------*  plt.plot ( epochs, acc, label='Training accuracy' )  plt.plot ( epochs, val\_acc, label='Validation accuracy' )  plt.title ('Training and validation accuracy')  plt.grid()  plt.legend()  plt.xlabel("Epochs")  plt.ylabel("Accuracy")  plt.figure()  *#------------------------------------------------*  *# Plot training and validation loss per epoch*  *#------------------------------------------------*  plt.plot ( epochs, loss, label='Training loss' )  plt.plot ( epochs, val\_loss, label='Validation loss' )  plt.grid()  plt.legend()  plt.xlabel("Epochs")  plt.ylabel("Loss")  plt.title ('Training and validation loss' )  *# Plot training results* plot\_loss\_acc(history) |
| --- |

## Enable Tensorboard on google colab

| *# !pip install tensorboardcolab* from tensorboardcolab import TensorBoardColab, TensorBoardColabCallback  tbc=TensorBoardColab() |
| --- |

## 

### Add to Keras callback

| history = model.fit(  train\_generator,  *# validation\_data=val\_ds,*  epochs=epochs,  steps\_per\_epoch = 115,  verbose = 2,  callbacks = [TensorBoardColabCallback(tbc)] ) |
| --- |

## Plot to view images

| import matplotlib.pyplot as plt  plt.figure(figsize=(10, 10)) for images, labels in train\_dataset.take(1):  for i in range(9):  ax = plt.subplot(3, 3, i + 1)  plt.imshow(images[i].numpy().astype("uint8"))  plt.title(class\_names[labels[i]])  plt.axis("off") |
| --- |

### Validate image and label size

| for image\_batch, labels\_batch in train\_ds:  print(image\_batch.shape)  print(labels\_batch.shape)  break |
| --- |

## Set number of characters per row when printing

| import os import tensorflow as tf from tensorflow import keras import numpy as np *#set number of characters per row when printing* np.set\_printoptions(linewidth=320) np.set\_printoptions(suppress=True) |
| --- |

## Clean up:

### Clears Global state of Keras memory

| import tensorflow as tf tf.keras.backend.clear\_session() |
| --- |

### Unassign CPU using Google colab

| from google.colab import runtime runtime.unassign() |
| --- |

### Clean up kernel and memory

| import os, signal os.kill(os.getpid(), signal.SIGKILL) |
| --- |

## Windowed dataset function for multivariate series:

## You can use the below code if you have univariate output (one target). This code can work for multiple features but the target should be one. Both features and target should be Numpy arrays and have a 2D shape. The result will be of (batch\_size, window\_size, features) and (batch\_size, window\_size, target) and can be feed to LSTM.

| def windowed\_dataset(features, target, window\_size, batch\_size):  dataset = tf.data.Dataset.from\_tensor\_slices((features, target))  dataset = dataset.window(window\_size + 1, shift=1, drop\_remainder=True)  dataset = dataset.flat\_map(lambda x, y: tf.data.Dataset.zip((x.batch(window\_size + 1), y.batch(window\_size + 1))))  dataset = dataset.map(lambda x, y: (x[:-1], y[1:]))   dataset = dataset.batch(batch\_size).prefetch(1)  return dataset |
| --- |

## 

## Inferring Pixel of images

### Method 1

| *# importing the module*  import PIL  from PIL import Image    *# loading the image*  img = PIL.Image.open(os.path.join(train\_cats\_dir, os.listdir(train\_cats\_dir)[9]))   *# fetching the dimensions*  wid, hgt = img.size  *#print the pixels of image* print(wid, hgt) |
| --- |

### Method 2

| *# Load the first example of a horse* sample\_image = load\_img(f"{os.path.join(train\_horses\_dir, os.listdir(train\_horses\_dir)[0])}") *# Convert the image into its numpy array representation* sample\_array = img\_to\_array(sample\_image) print(f"Each image has shape: {sample\_array.shape}") |
| --- |

## Load model

| import tensorflow as tf import tensorflow\_hub as hub new\_model = tf.keras.models.load\_model('/content/drive/MyDrive/Data Science & Machine Learning/Tensorflow Certification/exam/4/mymodel (1).h5',  custom\_objects={'KerasLayer':hub.KerasLayer})  # Show the model architecture new\_model.summary() |
| --- |

## 

## Evaluating the correct steps\_per\_epoch & validation\_steps

| training\_instance = train\_generator.n #total count of files in train\_generator set validation\_instance = validation\_generator.n #total count of files in validation\_generator set steps\_per\_epoch = training\_instance*//train\_generator.batch\_size* validation\_steps = validation\_instance*//validation\_generator.batch\_size*  print("steps\_per\_epoch :",steps\_per\_epoch) print("validation\_steps :",validation\_steps) |
| --- |

# Medium Certification preps

* **Github -** [**https://github.com/Enzofali/The-Ultimate-Google-TensorFlow-Developer-Certificate-Exam-Study-Guide?tab=readme-ov-file**](https://github.com/Enzofali/The-Ultimate-Google-TensorFlow-Developer-Certificate-Exam-Study-Guide?tab=readme-ov-file)
* [**https://github.com/kolasniwash/tensorflow-certification-study-guide**](https://github.com/kolasniwash/tensorflow-certification-study-guide)
* **Practice Tutorial on keras website - https://www.tensorflow.org/tutorials/quickstart/advanced**
  + Viraj Datt Certification -<https://kvirajdatt.medium.com/tips-and-tricks-for-the-tensorflow-developer-certificate-exam-4a555f1adb38>
  + Daniel Bourke Video - https://www.mrdbourke.com/how-i-got-tensorflow-developer-certified/
  + <https://www.youtube.com/watch?v=ya5NwvKafDk>

<https://www.youtube.com/watch?v=ya5NwvKafDk&t=500s>

* + Harshit Tyagi -<https://www.freecodecamp.org/news/how-i-passed-the-certified-tensorflow-developer-exam/>
  + Computer vision engineer -<https://www.youtube.com/watch?v=6DAhtNJI88Q>
  + Pere Martra\*\* -<https://pub.towardsai.net/my-journey-to-becoming-a-tensorflow-certified-developer-407ca899c1b4>
* [Intro to Deep Learning MIT](https://www.youtube.com/watch?v=njKP3FqW3Sk&list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI): These are free lectures by MIT on Deep Learning. The recommended lecturers are first three, each with 40 minutes long. They are very practical, quick and helpful in understanding the basics of deep learning, convolutional neural networks and sequence models.
  + Binod Suman Academy: [https://www.youtube.com/playlist?list=PLIRnO\_sdVuEfau\_eJKVhiaLaqIXCT0F-\_](https://www.youtube.com/watch?v=Y1qxI-Df4Lk)
    - Convolutional Neural Networks | CNN | Kernel | Stride | Padding | Pooling | Flatten | Formula -<https://www.youtube.com/watch?v=Y1qxI-Df4Lk>
* <https://mlnoobs.hashnode.dev/everything-you-need-to-know-about-tensorflow-certification-exam>
* **Getting Started Question**A basic regression question dataset will be given and we have to train a neural network to match the x to the y.
* **Basic Datasets Question**Create and train a classifier for the Tensorflow Image dataset. Note that the test will expect it to classify n classes and that the input shape should be the native size of the TF dataset which is NxN monochrome. Do not resize the data. Your input layer should accept NxN as the input shape only. If you amend this, the tests will fail.
* **Computer vision with CNNs**Create and train a classifier for CV Image Dataset using the provided data. The test will use images that are NxN with N bytes color depth so be sure to design your neural network accordingly
* **NLP QUESTION**Build and train a classifier for the NLP Dataset. It will be tested against a number of sentences that the network hasn't previously seen and you will be scored on whether prediction was correctly detected in those sentences.
* **TIME SERIES QUESTION**Build and train a neural network to predict the time-indexed variable of the dataset\_name Using a window of past N observations of N feature, train the model to predict the next N observations of that feature.
* **Reference Notes:**
  + Understanding the [tf.data](https://www.tensorflow.org/api_docs/python/tf/data/Dataset). This is the modern way of handling data in your TensorFlow projects. The methods mentioned here can be quite exhaustive while beginning here I have shortlisted a few tf.data API’s that I feel are used frequently [github](https://github.com/Virajdatt/TensorFlow_Cert_Learning/blob/main/minor_projects/tf_dataapi.py) . Here is a complete example on keras [tutorial](https://keras.io/examples/vision/image_classification_from_scratch/) for an image classification task that demonstrates the usage of image\_dataset\_from\_directory which creates the image data in tf.data format which leads to building better data cleaning and transforming pipeline for your data in TensorFlow.
  + The Udacity course Intro to TensorFlow for [Deep Learning](https://www.udacity.com/course/intro-to-tensorflow-for-deep-learning--ud187) really helped to better my understanding of Time Series Analysis and how seq-seq and seq-vectors work in TF and Keras.
  + Understanding the [tensorflow\_dataset](https://www.tensorflow.org/datasets/overview) module. TensorFlow Dataset(tfds) is an amazing way for loading datasets during your work and especially during practice. The following code demonstrates how powerful tfds is for loading, splitting your data during practice.

*import tensorflow\_datasets as tfds*

*# 1. Load your data through a dictonary*

*data, metadata = tfds.load('mnist',as\_supervised=True,with\_info=True)*

*train, test = data['train'], data['test']*

*# 2. Load the train and test data seperately*

*(train, test), metadata = tfds.load('mnist',split=['train', 'test'], as\_supervised=True, with\_info=True)*

*# 3. Load the data as train, validation , test*

*(data, metadata) = tfds.load('mnist',split=['train[:90%]', 'train[90%:]', 'test'],as\_supervised=True,with\_info=True)*

*train\_data = data[0]*

*valid\_data = data[1]*

*test\_data = data[2]*

* Develop the habit of using tensorboard to look at your accuracy and loss curves. It is easy with TensorBoard callback in keras. This eliminates the need to use matplotlib and you can vizualize the accuracy and loss of models of multiple runs during your fine-tuning.The details for this Callback API are present at [TensorBoard](https://keras.io/api/callbacks/tensorboard/)

# Code Snippet for TensorBoard

TB = keras.callbacks.TensorBoard('<path-to-save-tensorboard-logs>')

Here is the [github](https://github.com/Virajdatt/TensorFlow_Cert_Learning/tree/main/handy_snippets) link which lists all these callbacks.

* Review handy snippets of code needed during exam - [*TF\_HANDY\_SNIPPETS*](https://github.com/Virajdatt/TensorFlow_Cert_Learning/tree/main/handy_snippets)
* Data Preprocessing article - https://jonathan-hui.medium.com/tensorflow-keras-preprocessing-layers-dataset-performance-considera-e9fc11de7bc7
* **Practice:**
  + a. **tf.keras.callbacks.ModelCheckpoint** :- This callback will save the best model based on the criteria you have mentioned. The details for this Callback API are present at [ModelCheckpoint](https://keras.io/api/callbacks/model_checkpoint/).

# Code Snippet for ModelCheckpoint

MC = tf.keras.callbacks.ModelCheckpoint('<path-to-saving-model>',

monitor='val\_loss',

save\_best\_only='True',

verbose=1)

* + b. **tf.keras.callbacks.EarlyStopping**:- This callback will stop training based on the criteria you provide. This is a really handy callback in the exam as you can run your training for extended amout of epochs and not worry about overfitting as the training will be cutshort when the stop criteria is reached.The details for this Callback API are present at [EarlyStopping](https://keras.io/api/callbacks/early_stopping/).

# Code Snippet for EarlyStopping

ES = tf.keras.callbacks.EarlyStopping(monitor='val\_loss', patience=5,

verbose=1,

restore\_best\_weights='True')

* c. **tf.keras.callbacks.LearningRateScheduler**:- This callback lets you dynamically update the learning rate of an optimizer during training. The details for this Callback API are present at [LearningRateScheduler](https://keras.io/api/callbacks/learning_rate_scheduler/).

# Code Snippet for LearningRateScheduler

LR = keras.callbacks.LearningRateScheduler(lambda epoch: 1e-5 \* 10 \*\* (epoch/2), verbose=1)

* At the same time, finding the Appropriate Learning Rate is an experimental task. Here, ReduceLROnPlateau can be used. When the loss does not decrease with a certain patience coefficient, it decreases the LR at the determined rate.

# Practice following types of problem:

## Linear Regression

* Snippet to make a single-layer **linear regression** model.

### Practice:

* ~~Univariate and multivariate Regression~~

### Samples:

* Data-Set:-
  + [~~https://github.com/Virajdatt/TensorFlow\_Cert\_Learning/blob/main/revision/simple\_reg.py~~](https://github.com/Virajdatt/TensorFlow_Cert_Learning/blob/main/revision/simple_reg.py)
  + [~~https://github.com/Virajdatt/TensorFlow\_Cert\_Learning/blob/main/tf\_test.py~~](https://github.com/Virajdatt/TensorFlow_Cert_Learning/blob/main/tf_test.py)

### Pending:

* Cali Housing Dataset -
  + https://captum.ai/tutorials/House\_Prices\_Regression\_Interpret
* Student Performance
  + <https://archive.ics.uci.edu/dataset/320/student+performance>
  + https://www.kaggle.com/datasets/nikhil7280/student-performance-multiple-linear-regression/code
* Wine Quality

## **CNN Tasks**

a. Splitting data from disk into multiple folders (train, test, and validation)  
b. ImageDataGenerator with its various transformations (Image Augmentation).  
c. Transfer Learning snippet (using InceptionV3, ResNet52, etc..)  
d. A few custom CNN models based on the practice you do.

### Practice:

* Binary and Multi-Class Image Classification

### Samples:

* + ~~MNIST,~~ 
    - ~~https://github.com/Virajdatt/TensorFlow\_Cert\_Learning/blob/main/revision/revision\_1.py~~
  + ~~Fashion-MNIST,~~ 
    - ~~https://github.com/Virajdatt/TensorFlow\_Cert\_Learning/blob/main/revision/revision\_2.py~~
  + ~~Sign-MNIST,~~ 
    - ~~https://github.com/Virajdatt/TensorFlow\_Cert\_Learning/blob/main/revision/cnn\_multi.py~~
  + ~~Digit Recognizer~~
    - [**~~Digit Recognizer~~**](https://www.kaggle.com/competitions/digit-recognizer)
      * https://www.kaggle.com/code/peremartramanonellas/mnist-digit-recognigtion-0-9974-with-tensorflow?source=post\_page-----407ca899c1b4--------------------------------
  + [Flower Dataset](https://www.tensorflow.org/datasets/catalog/tf_flowers),
    - Using data augmentation using ImageDataGenerator: <https://github.com/Virajdatt/TensorFlow_Cert_Learning/blob/main/revision/flower_dataset.py>
    - Using ​​image\_dataset\_from\_directory: <https://github.com/Virajdatt/TensorFlow_Cert_Learning/blob/main/revision/flowers_cnn_tf_dataset.py>
    - Configure for performance & tfds.load module: <https://github.com/Virajdatt/TensorFlow_Cert_Learning/blob/main/revision/flowers_tfds.py>
    - Using tensor dataset: <https://github.com/Virajdatt/TensorFlow_Cert_Learning/blob/main/revision/flower_tfds2.py>
    - Using InceptionV3 transfer learning: <https://github.com/Virajdatt/TensorFlow_Cert_Learning/blob/main/revision/flowers_tl_cnn.py>
  + ~~Cats vs Dogs~~
    - [~~https://github.com/Virajdatt/TensorFlow\_Cert\_Learning/blob/main/revision/cat\_dog\_split\_data.py~~](https://github.com/Virajdatt/TensorFlow_Cert_Learning/blob/main/revision/cat_dog_split_data.py)
  + Pet Data
    - https://github.com/Virajdatt/TensorFlow\_Cert\_Learning/blob/main/revision/transfer\_learning\_revision.py
  + <https://github.com/georgesung/traffic_sign_classification_german/blob/master/Traffic_Signs_Recognition.ipynb>
  + <https://towardsdatascience.com/input-pipeline-for-images-using-keras-and-tensorflow-c5e107b6d7b9>
  + Gem dataset:
    - https://www.kaggle.com/code/lsind18/gemstones-multiclass-classification-cnn#III.-Fit-the-train-generator
  + CIFAR
    - <https://www.kaggle.com/code/bhavinmoriya/cifar10-multiclass-classification-using-cnn>
    - https://www.kaggle.com/code/bhuvanchennoju/cifar-10-image-classification-with-cnn
  + Bird species classification
    - https://www.kaggle.com/datasets/kedarsai/bird-species-classification-220-categories/code

### Pending:

* Tensorflow Tutorials
  + <https://www.tensorflow.org/tutorials/load_data/images>
  + <https://www.tensorflow.org/tutorials/images/data_augmentation>
* ~~TensorBoard setup~~
  + ~~https://neptune.ai/blog/how-to-manage-track-visualize-hyperparameters~~
* ~~Callback implementation~~
* Change Pixel to image and vice versa
* Code snippet View image
* Research data preprocessing needed in model layer or outside; practice accordingly
* KerasTuner
  + \*\* V Imp <https://www.analyticsvidhya.com/blog/2021/08/easy-hyperparameter-tuning-in-neural-networks-using-keras-tuner/>
  + \*V. Imp <https://pythonprogramming.net/keras-tuner-optimizing-neural-network-tutorial/>
  + \*https://medium.com/swlh/hyperparameter-tuning-in-keras-tensorflow-2-with-keras-tuner-randomsearch-hyperband-3e212647778f
  + <https://pyimagesearch.com/2021/06/07/easy-hyperparameter-tuning-with-keras-tuner-and-tensorflow/>
  + <https://medium.com/@viniciusqroz/using-keras-tuner-to-find-the-best-parameters-for-your-neural-network-model-2dc02e0a1203>
  + <https://huggingface.co/learn/nlp-course/chapter3/3?fw=tf>
  + <https://www.analyticsvidhya.com/blog/2021/06/tuning-hyperparameters-of-an-artificial-neural-network-leveraging-keras-tuner/>
  + https://kegui.medium.com/a-few-pitfalls-for-kerastuner-beginner-users-13116759435b#:~:text=Hyperas%20is%20great%2C%20very%20intuitive,if%20it%20has%20Bayesian%20Optimization.
  + Hyperas:
    - <https://towardsdatascience.com/a-guide-to-an-efficient-way-to-build-neural-network-architectures-part-ii-hyper-parameter-42efca01e5d7>
    - https://medium.com/@hwu4/tips-for-hyperas-a058b27b8aa6

## NLP Tasks

a. Tokenizer code for NLP (text to sequence and padded sequence).  
b. Embedding layers (how to create them and visualize )  
c. A few RNN, LSTM, GRU, and Conv1D architectures that you used during practice.  
d. Loading Text data from CSV and JSON.

### Practice:

* Binary and Multi-Class Text Classification
* Text Generation

### Samples:

* + **Text Classification:-**Data-Set:-
    - [~~https://github.com/Virajdatt/TensorFlow\_Cert\_Learning/blob/main/revision/text\_revision\_text\_preprocess.py~~](https://github.com/Virajdatt/TensorFlow_Cert_Learning/blob/main/revision/text_revision_text_preprocess.py)
    - ~~https://github.com/Virajdatt/TensorFlow\_Cert\_Learning/blob/main/revision/text\_using\_existing\_embeddings.py~~
  + ~~IMDB,~~ 
    - ~~https://github.com/Virajdatt/TensorFlow\_Cert\_Learning/blob/main/revision/text\_revision\_word\_embeddings.py~~
  + ~~BBC,~~ 
    - ~~https://github.com/Virajdatt/TensorFlow\_Cert\_Learning/blob/main/revision/text\_revision\_text\_preprocess\_csv.py~~
  + ~~Scarasm dataset~~ 
    - ~~https://github.com/Virajdatt/TensorFlow\_Cert\_Learning/blob/main/revision/text\_revision\_text\_preprocess\_json.py~~
  + Tweets
    - [**Natural Language Processing with disaster tweets**](https://www.kaggle.com/competitions/nlp-getting-started)**.**
    - Natural Language Processing with Disaster Tweets ([dataset](https://www.kaggle.com/c/nlp-getting-started)),
      * Try GLoVE
      * Try BeRT
      * <https://www.kaggle.com/code/peremartramanonellas/guide-tweet-analysis-with-transfer-learning?source=post_page-----407ca899c1b4--------------->
      * <https://www.kaggle.com/competitions/nlp-getting-started/code?competitionId=17777&sortBy=voteCount&excludeNonAccessedDatasources=true----------------->
      * <https://www.kaggle.com/code/shahules/basic-eda-cleaning-and-glove/notebook#Making-our-submission>
      * <https://www.kaggle.com/code/mariapushkareva/nlp-disaster-tweets-with-glove-and-lstm>
  + Shakesphere dataset for text generation

### Pending:

* If time permits, improve exam final model using - https://www.kaggle.com/code/quadeer15sh/transformers-for-text-classification/notebook
* Download sarcasm dataset and preprocess data to convert to json
* hyperas with nlp
  + <https://medium.com/@fiona.s.feng/hyperparameter-tuning-for-text-classification-in-keras-tensorflow-with-hyperas-86668a7e732b>
  + <https://www.linkedin.com/pulse/hyperas-prateek-khanna/>
  + <https://medium.com/@hwu4/tips-for-hyperas-a058b27b8aa6>
  + https://statisticalinterference.wordpress.com/2017/06/15/hyperas-for-lstms/
  + Try passing intermediate function
    - https://stackoverflow.com/questions/53777457/cant-use-intermediate-function-in-hyperas
* Setup Python and environment dependencies:
  + <https://medium.com/the-owl/changing-the-python-version-on-google-colab-245fd510d3ae>
  + <https://netraneupane.medium.com/how-to-install-libraries-permanently-in-google-colab-fb15a585d8a5>
    - <https://github.com/np-n/blog_code_snippets/blob/master/Installing%20Dependencies%20Permanently%20on%20Google%20Colab/installing_libraries_permanently_in_google_colab.ipynb>
  + <https://www.linkedin.com/pulse/working-google-colab-python-machine-learning-guide-rany/>
  + <https://www.reddit.com/r/GoogleColab/comments/13605i3/how_to_change_python_version_in_google_colab_env/>
    - <https://stackoverflow.com/questions/60775160/install-python-3-8-kernel-in-google-colaboratory/71511943#71511943>
  + <https://www.squash.io/how-to-use-different-python-versions-with-virtualenv/>
  + <https://github.com/Tahira2910/ETG_Machine_Learning_Projects/blob/main/Creating_Virtual_Environment_On_Google_Colab.ipynb>
* Buy medium subscription to view this article: <https://towardsdatascience.com/multiclass-text-classification-using-keras-to-predict-emotions-a-comparison-with-and-without-word-5ef0a5eaa1a0>

## Time-Series Tasks

a. Snippet to create data for time-series prediction.  
b. A few RNN, LSTM, GRU, and Conv1D architectures that you used during practice.

### Practice:

* Single Step Time Series Predictions
  + ~~univariate inputs, univariate outputs~~
  + univariate inputs, multivariate outputs
  + ~~multivariate inputs, univariate outputs~~
  + ~~multivariate inputs, multivariate outputs~~
* Multi-Step Time Series predictions
  + ~~univariate inputs, univariate outputs~~
  + univariate inputs, multivariate outputs
  + ~~multivariate inputs, univariate outputs~~
  + multivariate inputs, multivariate outputs
* Single and Multi-Step Time Series Predictions with multi-variate outputs

### Samples:

* + **~~Time-Series:-~~**~~Data-Set:-~~ [~~TimeSeries-Data~~](https://www.tensorflow.org/tutorials/structured_data/time_series)
    - ~~the most difficult was the time series prediction question, (probably due to my lack of experience in the field) and I’d recommend you read the chapter on that in the Geron textbook. Something I read in the chapter really came to my aid during the exam.~~

### Pending

* Multivariate:
  + https://www.kaggle.com/code/nicholasjhana/multi-variate-time-series-forecasting-tensorflow#Forecasting-With-Tensorflow:-Comparing-Model-Architectures
* Multistep - Multivariate:
  + https://www.kaggle.com/code/nicapotato/keras-timeseries-multi-step-multi-output
  + [N Steps Multi-Variate Time Series Forcasting](https://github.com/Enzofali/TensorFlowDeveloperCertificateMaterial-/blob/main/Time%20Series%20Forecasting/4_Multi-VariateTimeSeries.ipynb) [(Time Series Forecasting Tutorial)](https://www.tensorflow.org/tutorials/structured_data/time_series)⭐
  + [**Univariate Autoregressive Multi Step Problem**](https://www.kaggle.com/code/iamleonie/time-series-forecasting-building-intuition#Length-of-Forecast)
  + <https://www.kaggle.com/code/anshulmehtakaggl/time-series-with-tensorflow>
  + [meh] <https://www.kaggle.com/code/debashis74017/eda-and-forecast-bajaj-finance-stock-price#Multivariate-Time-series-forecasting>
  + https://github.com/shuchimishra/Tensorflow\_projects/blob/main/Tensorflow\_Code/Timeseries/Household\_power\_consumption\_multivariate\_multistep.ipynb

## HyperParameter Tuning:

* Don’t panic if the model is over-fitting or under-fitting. Try different architectures based on the problem at hand. Please don’t start with a solution that uses a huge model (a large number of layers/more number of neurons):-
* a. Start with a baseline taught during the courses.  
  b. Incrementally add layers.  
  c. Change the optimizer and loss function. Keep in mind of the callbacks discussed to avoid overfitting.  
  d. Make sure you understand the input and output layers based on the task.

**Revision Strategy**

Tricks:

* My first trick is to use multiple [Gpu](https://towardsai.net/p/deep-learning/what-is-a-gpu-are-gpus-needed-for-deep-learning-7b315ed80f16)’s. This method does not give a big boost, but it seems useful.
* When I was reading blogs, it was written that submitting the same model more than once returned different results. This was most likely due to the constant changing of the test set. If you want to get a higher score, you can test the same model more than once.
* Using callbacks can be advantageous. For example, ModelCheckpoint continuously saves your best model and you will not have a problem even if your model in training is overfit or underfit.
* At the same time, finding the Appropriate Learning Rate is an experimental task. Here, ReduceLROnPlateau can be used. When the loss does not decrease with a certain patience coefficient, it decreases the LR at the determined rate.
* Also, I recommend that the compute capability of the [GPU](https://towardsai.net/p/deep-learning/what-is-a-gpu-are-gpus-needed-for-deep-learning-7b315ed80f16) you will use be above 6.1.
* Our team already has [Google Colab](https://towardsai.net/p/programming/google-colab-101-tutorial-with-python-tips-tricks-and-faq-7689bd4d24b4) Pro to use for the exam. But there was a problem. Colab was using Tensorflow 2.8.0, so it could create some problems. I downgraded Tensorflow to 2.7.0, but Cuda did not work.
* My last choice was to use AWS. AWS offers prebuilt DeepLearning environments. I set up a g4dn.xlarge, EC2 instance, and build Jupiter notebook that I can connect from my laptop. And our powerful GPU is ready g4 series has Tesla T4 GPU, which is compute capability 7.5.
* The version of Tensorflow and other libraries used by the exam changes over time. Colab or AWS can be used when versions match. Even just 1 GPU will do a lot of work.
* Review tips -<https://www.linkedin.com/pulse/tensorflow-developer-certification-vivek-bombatkar/>
* It’s good to have some helper functions ready, which can help with things like check-pointing the best trained model, early stopping, tensorboard callbacks, to name a few.
* The TF course by Andrei Neagoie & Daniel Bourke on [Udemy](https://www.udemy.com/course/tensorflow-developer-certificate-machine-learning-zero-to-mastery/). The accompanying resources are excellent — highly recommend to practise all chapters from their Github repo. There are many tips with TF and deep learning in general too.
* One of my major learnings from the exam:Don't overoptimize. Went crazy to build the model for one of the tasks and it performed ~15 times better than the expected result! But the model turned out to be so big that it could not be graded :( removed LR schedulers, a couple of LSTM layers, a couple of Dense layers, trained for 15 minutes (on a Tesla V100 😀 ), and voila, a perfect score!

Environments:

* We need Pycharm to install the exam plugin. That will create a virtual environment with the exercises to be solved. But I did 100% of the work in my Jupyter local and Google Colab.
* You must save the models in .h5 format and copy them to one of the directories created by the exam plug-in.
* Then, you can hit the evaluate button, and Pycharm uploads the model and evaluates it.
* The name of the model is not relevant, but you must have only one model in the directory. You can send the model for evaluate as many times as you want, so you can carry out tests and keep the best model.
* The models score from 0 to 5. I would recommend passing all the problems quickly with a 4 and then, if you have time left over, dedicate yourself to improving your grade.
* **You have to be careful when sending the exam to be evaluated. We must be sure that we have the desired models in each directory. Those who are currently in the directory are evaluated, and not those previously evaluated, even if they had a better grade.**
* <https://www.tensorflow.org/extras/cert/Setting_Up_TF_Developer_Certificate_Exam.pdf>