**Project Title: Train a CNN on the SVHN Dataset for Classification**

**Requirements:**

1. Create account on GitHub
2. Share your profile link in the project report (to be submitted at the end of your project)
3. Upload your code to GitHub

Prepare a short report:

**Title:** Street View Housing Number

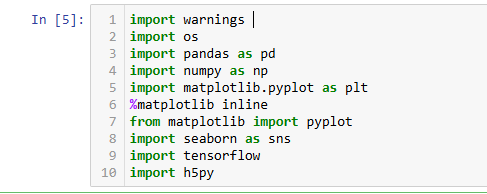
**Your name:** Muhammad Hasnain

**Email address:** muhammadhassain55@gmail.com

**GitHub profile link:**

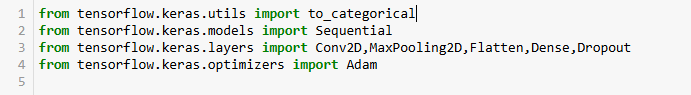
Report Summary

Detailing the project, explain each line of code step by step.



Explanation of above each line.

1. **Import warnings** statement in Python is used to manage warning messages. It helps alert users about potential issues like deprecate features, resource leaks, or configuration problems without halting the program.
2. **import os** in Python allows interaction with the operating system.
3. **import pandas as pd** imports the **Pandas** library with the alias pd. This library is widely used for data manipulation and analysis.
4. **import numpy as np** imports the **NumPy** library, widely used for numerical computing in Python.
5. **import Matplotlib. Pyplot as plt** imports the **Matplotlib** library's pyplot module, used for creating visualizations in Python
6. **%matplotlib inline** is a **magic command** in Jupyter Notebooks. It ensures that plots created with **Matplotlib** are displayed directly **within the notebook** instead of in a separate window. This makes it convenient for interactive data analysis.
7. **from matplotlib import pyplot** imports the pyplot module from the **Matplotlib** library, used for creating visualizations. This is functionally equivalent to import Matplotlib. Pyplot as plt, but without the alias
8. **import seaborn as sns** imports the **Seaborn** library, built on top of Matplotlib, for advanced and visually appealing statistical graphics. It simplifies complex visualizations and works well with **Pandas Data Frame.**
9. **import tensorflow** imports the **TensorFlow** library, an open-source framework developed by Google for **machine learning** and **deep learning**. It provides tools for building and training models, especially neural networks.
10. **import h5py** imports the **h5py** library, which provides an interface to read and write **HDF5** (Hierarchical Data Format version 5) files in Python. HDF5 is a popular format for storing large amounts of data, particularly in scientific computing.

****

Explanation of above each line.

1. **from tensorflow.keras.utils import to\_categorical** imports the to\_categorical function from TensorFlow’s Keras utilities. This function converts **integer labels** (e.g., 0, 1, 2) into **one-hot encoded vectors**, which are often required for classification tasks in machine learning.
2. **from tensorflow.keras.models import Sequential** imports the Sequential class, used to build neural networks layer-by-layer in a linear stack.
3. **from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout** imports commonly used layers for building **convolutional neural networks (CNNs)** in TensorFlow's Keras API. Here's a brief explanation of each.

* **Conv2D**: Applies 2D convolution to extract spatial features
* **MaxPooling2D**: Downsamples feature maps using max-pooling, reducing dimensionality and computation.
* **Flatten**: Converts multi-dimensional data (e.g., images) into a 1D vector for input to dense layers.
* **Dense**: A fully connected layer for learning high-level features.
* **Dropout**: Randomly drops a fraction of neurons during training to prevent overfitting.

1. **from tensorflow.keras.optimizers import Adam** imports the **Adam optimizer**, one of the most popular optimization algorithms for training deep learning models

****

The command print(tensorflow.\_\_version\_\_) displays the installed **TensorFlow version**.

****

warnings.filterwarnings('ignore') is used to **suppress warning messages** in Python. It ensures that the program runs without printing warnings, which can be useful when you know the warnings are safe to ignore (e.g., deprecated functions) and don't want them cluttering the output.

****

pd.options.display.max\_columns = None ensures that **all columns of a DataFrame** are displayed when printed, without truncation. By default, Pandas may limit the number of columns shown if there are too many, adding ... in the middle or end.

****

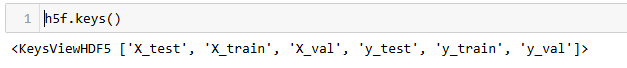
pd.options.display.float\_format = '{:.7f}'.format configures how **floating-point numbers** are displayed in Pandas DataFrames. It ensures that all floats are shown with **7 decimal places.**

****

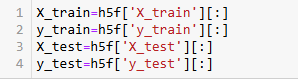
pd.options.display.max\_rows = None ensures that **all rows of a DataFrame** are displayed when printed, without truncation. By default, Pandas limits the number of rows shown to avoid excessive output.

****

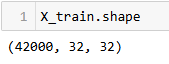
**Reading the .h5 file and assign to a variable**

****

In Python, if you're working with an HDF5 file using the h5py library, h5f.keys() returns a list of all the top-level datasets or groups stored within the file.

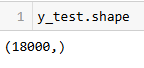
****

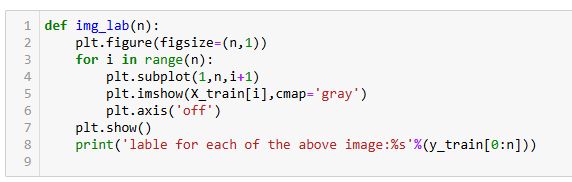
Split the data into X\_train,X\_test,y\_train,y\_test. load the training, test and validation set

****

****

****

****

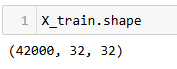
****

The img\_lab(n) function **displays the first n images** from the X\_train dataset with **grayscale coloring** and **prints their labels** from y\_train.

### Key Steps:

1. **Figure setup**:  
   plt.figure(figsize=(n, 1)) creates a horizontal figure with n subplots.
2. **Loop through images**:  
   Each image is plotted using plt.imshow() with cmap='gray'.
3. **Axis off**:  
   Hides axes for cleaner visualization.
4. **Show plot**:  
   plt.show() displays the images.
5. **Print labels**:  
   Prints the labels for the displayed images from y\_train.

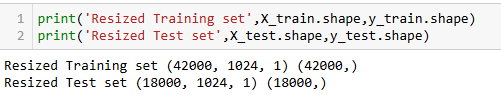
****

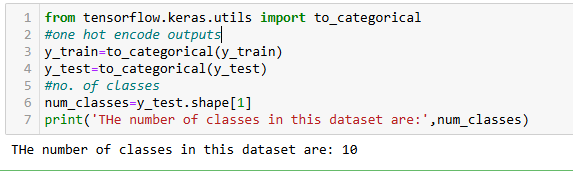
****

****

****

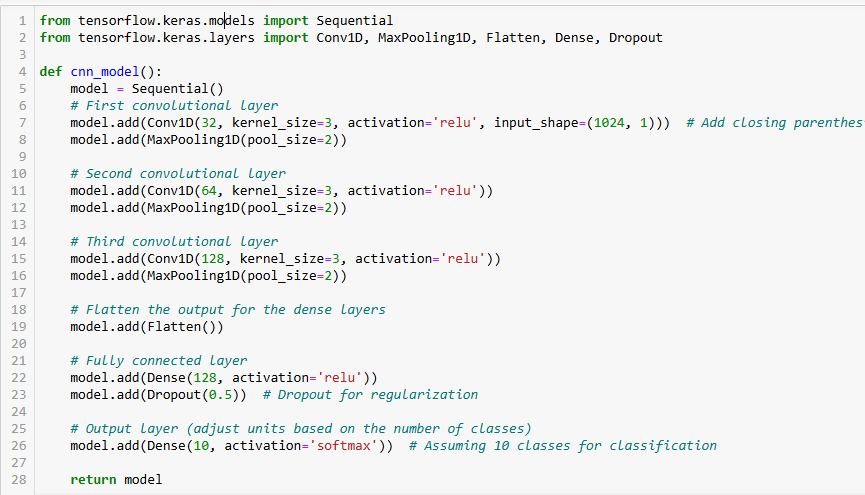
normalize inputs from 0 -255 to 0 to 1

****

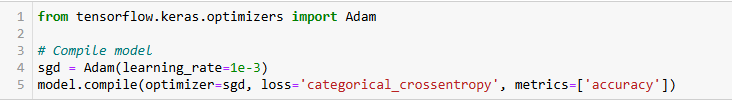
****

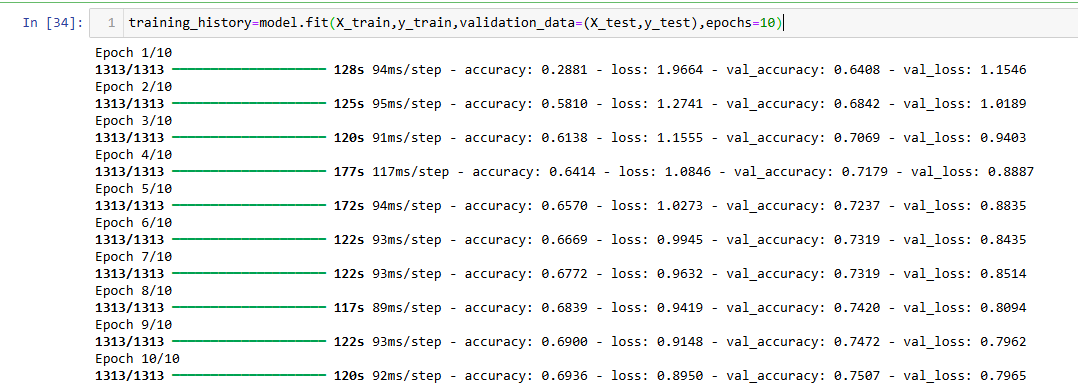
The code performs the following tasks:

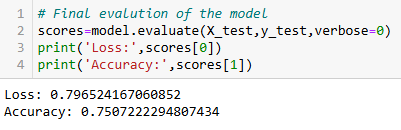
1. **Imports to\_categorical** from TensorFlow to convert labels into a one-hot encoded format.
2. **One-hot encodes** the training (y\_train) and testing (y\_test) labels, transforming them from integer values to binary vectors.
3. **Determines the number of classes** in the dataset by checking the second dimension of the one-hot encoded y\_test.
4. **Prints the number of classes** present in the dataset.

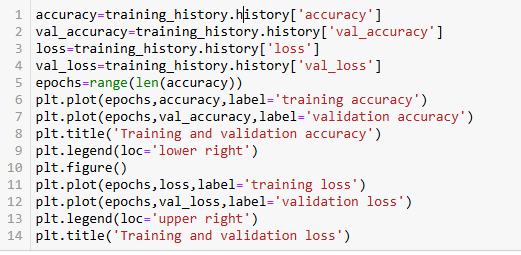
****



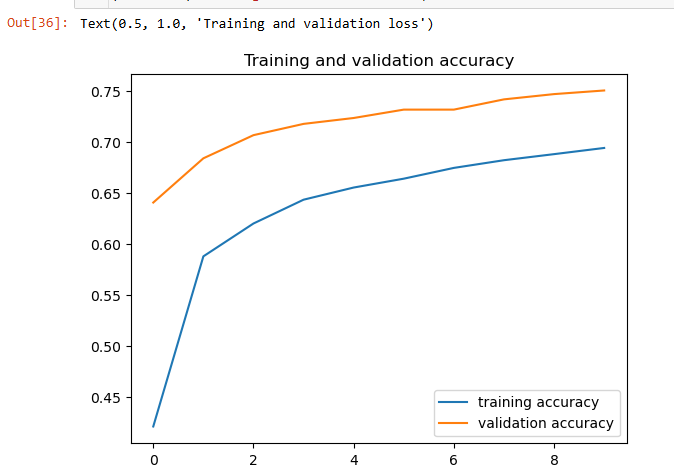


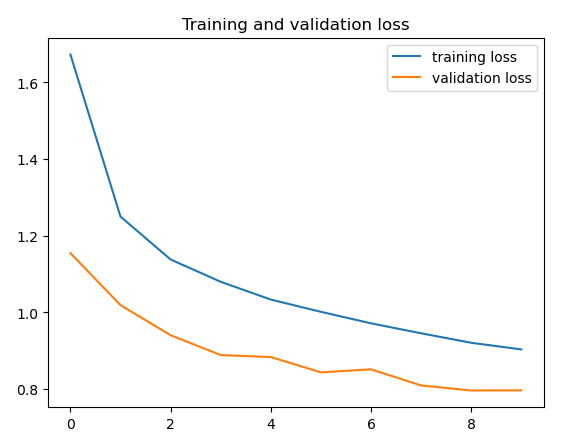






This code snippet is used to **visualize the training process** of a machine learning model by plotting both the training and validation accuracy, as well as the training and validation loss over epochs.



****