

**By** : ANNAKI OUSSAMA

: GUEMMOUT SALMA

**Deep Learning Assignement :**

MSc : Data Science in Economics & Finance

**Royaume du Maroc**

Université Abdelmalek ESSAÂDI

Faculté des sciences Juridiques, Economiques et Sociales

-Tanger-

[Global Temperature Time Series. 6](#_Toc157386850)

[1. The Model Used : RNN “LSTM” 6](#_Toc157386851)

[a. What are LSTM models: 6](#_Toc157386852)

[b. Why use LSTM for time series prediction: 6](#_Toc157386853)

[2. The Code : 8](#_Toc157386854)

[a. Libraries used: 8](#_Toc157386855)

[b. Data Preview : 8](#_Toc157386856)

[c. Preparing the Data : 8](#_Toc157386857)

[d. Building the model and Training : 9](#_Toc157386858)

[e. Making the predictions : 9](#_Toc157386859)

[f. Plotting the results : 9](#_Toc157386860)

[3. What has been done : 10](#_Toc157386861)

[4. Room for improvement : 11](#_Toc157386862)

[a. Loss function : 11](#_Toc157386863)

[b. GPU acceleration : 11](#_Toc157386864)

[Intel image classification : 12](#_Toc157386865)

[1. The model used : CNN 12](#_Toc157386866)

[a. What’s a CNN and why ? : 12](#_Toc157386867)

[2. The code : ( fully available in the video sent along this file) 14](#_Toc157386868)

[3. The results : 14](#_Toc157386869)

[4. Improvements and discussion : 15](#_Toc157386870)

This report concerns the two following assignments idea and planification behind their realisation, and ways and potentials of improvements:

**Task 1: Global Temperature Time Series**

https://datahub.io/core/global-temp?ref=hackernoon.com#readme

Data are included from the GISS Surface Temperature (GISTEMP) analysis and

the global component of Climate at a Glance (GCAG). You are expected to only

predict the global monthly mean temperature.

* Desing and train a model to predict the monthly mean temperature. (25 marks)
* Discuss how you would improve the prediction of your chosen model and implement/test the improvement. Briefly report your findings. (15marks)

**Task 2: Intel Image Classification**

https://www.kaggle.com/datasets/puneet6060/intel-image-classification

In this task the images are to be classified into one of six categories buildings,

forest, glacier, mountain, sea, or street.

* Desing and train a model to classify the images into the predict the monthly

mean temperature. (35 marks)

* Discuss how you would improve the prediction of your chosen model and

implement/test the improvement. Briefly report your findings. (25marks)

# Global Temperature Time Series.

## The Model Used : RNN “LSTM”

### What are LSTM models:

* **Recurrent Neural Network (RNN):**

A Recurrent Neural Network (RNN) is a type of neural network architecture designed for sequential data processing. Unlike traditional feedforward neural networks, RNNs have connections that form cycles, allowing them to maintain a memory of previous inputs. This makes RNNs well-suited for tasks involving sequences, such as time series prediction, natural language processing, and speech recognition.

* **Long Short-Term Memory (LSTM):**

Long Short-Term Memory (LSTM) is a specialized type of RNN designed to address the vanishing gradient problem, which is common in traditional RNNs. LSTMs include memory cells and gating mechanisms that enable them to capture and remember long-term dependencies in data. The architecture of LSTMs allows them to selectively update, read, and reset the information stored in their memory cells. This makes LSTMs particularly effective for modelling sequences with long-term dependencies.

### Why use LSTM for time series prediction:

* **Handling Long-Term Dependencies:**

LSTMs are designed to address the challenge of capturing dependencies in sequential data over extended periods. In time series data, where past observations often influence future values, LSTMs can effectively model long-term dependencies.

* **Memory Cell Mechanism:**

The memory cell architecture in LSTMs allows them to selectively remember or forget information over time. This enables the model to capture patterns in the data that occur at different time scales.

* **Vanishing Gradient Mitigation:**

The vanishing gradient problem is a common issue in training deep neural networks, especially RNNs. LSTMs include gating mechanisms that help mitigate this problem, allowing for more stable and efficient training of deep sequential models.

* **Flexibility in Sequence Length:**

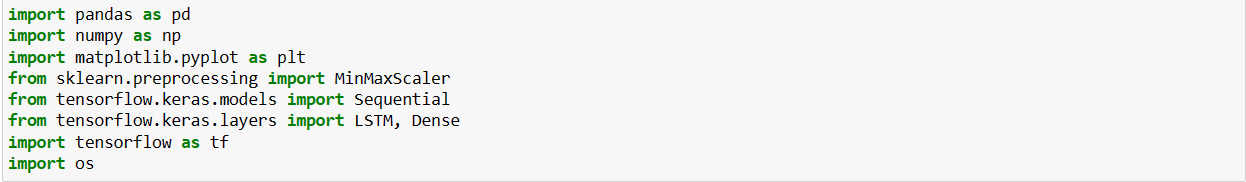
LSTMs can handle sequences of varying lengths, making them suitable for time series data with irregular intervals or missing observations.

* **Robustness to Noisy Data:**

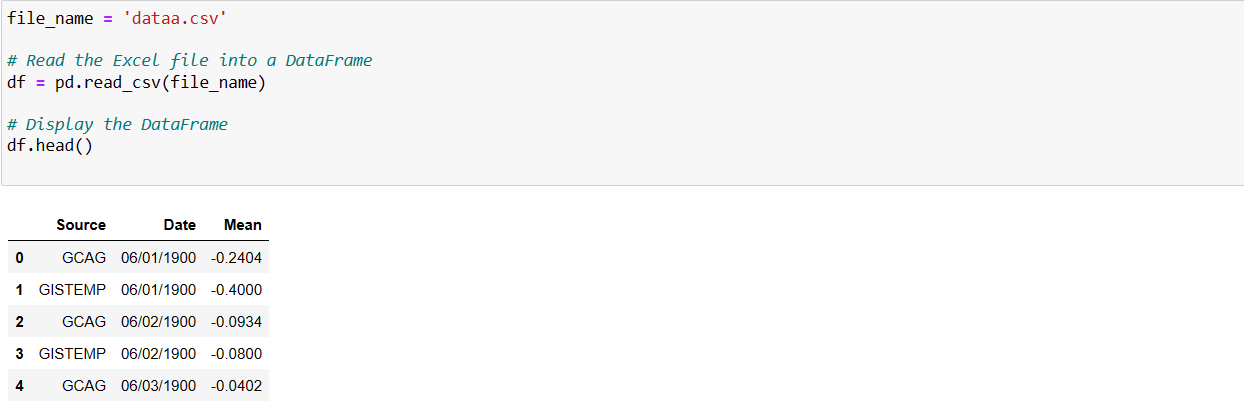
LSTMs are robust in handling noisy data and can learn to filter out irrelevant information through their memory cell mechanisms.

## The Code :

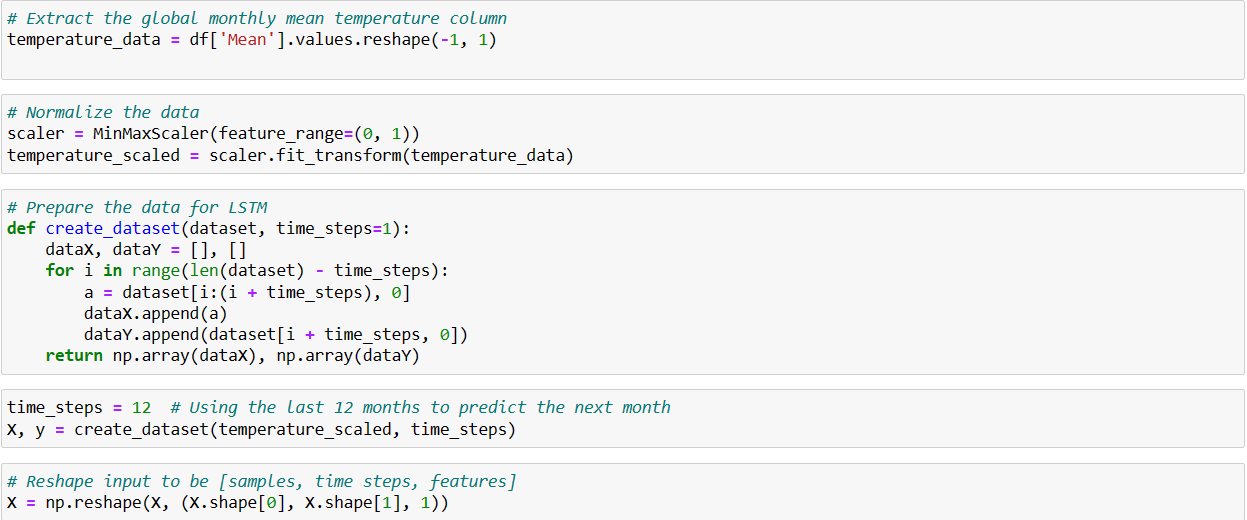
### Libraries used:



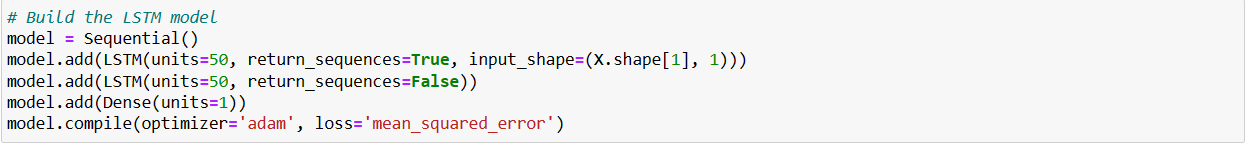
### Data Preview :



### Preparing the Data :

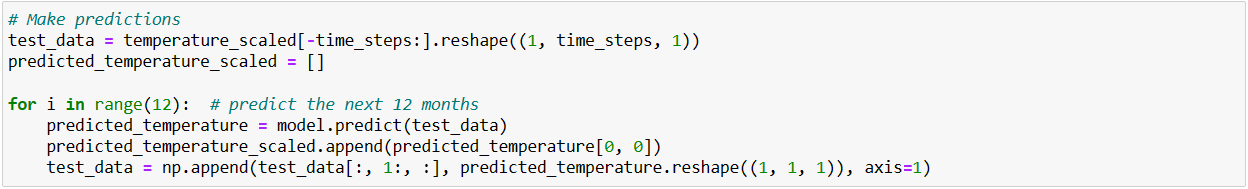


### Building the model and Training :

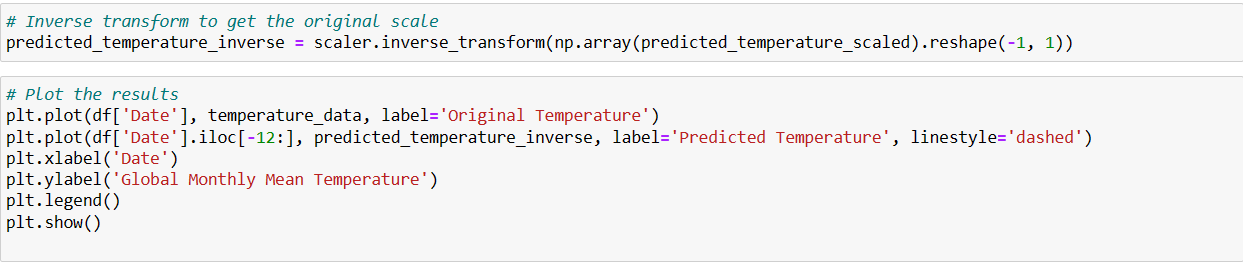


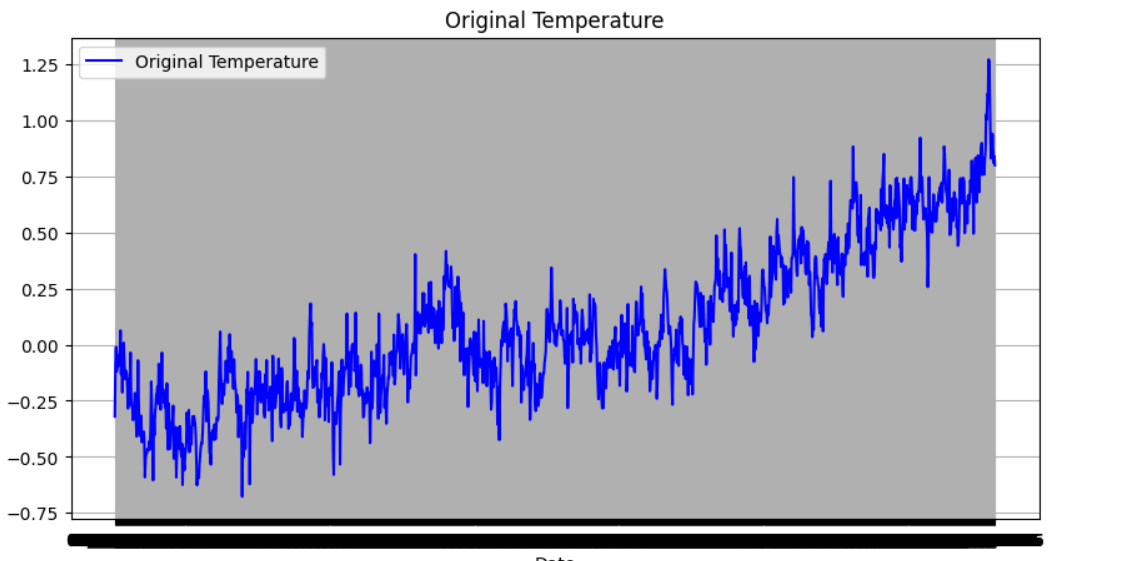
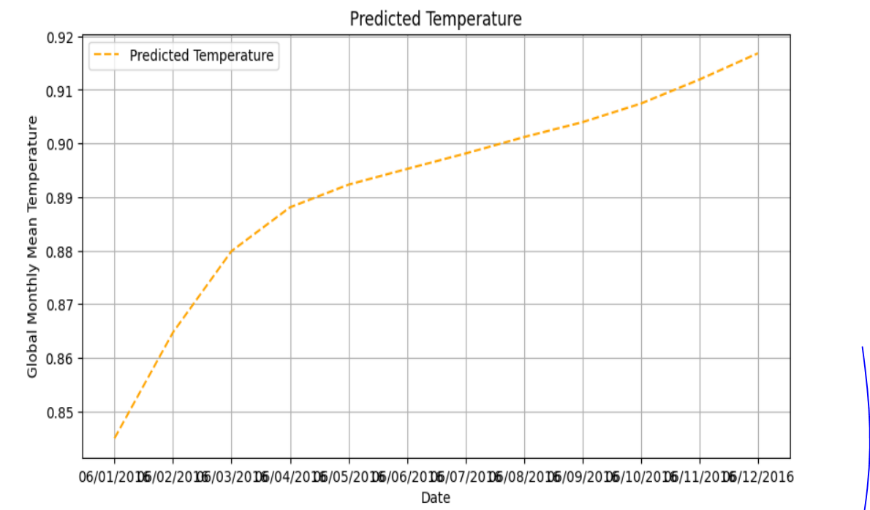


### Making the predictions :



### Plotting the results :





## What has been done :

* **Load Data:**

- Load global monthly mean temperature data from a CSV file into a Pandas DataFrame.

* **Extract and Reshape Data:**

- Extract the 'Mean' column from the DataFrame and reshape it into a 2D NumPy array.

* **Data Normalization:**

- Use Min-Max Scaling to normalize the temperature data to a range between 0 and 1.

* **Prepare Data for LSTM:**

- Define a function to create input-output pairs for training the LSTM model based on a specified number of time steps.

* **Define Time Steps and Create Dataset:**

- Set the number of time steps, indicating the past observations used to predict the next value.

- Create input (`X`) and output (`y`) sequences using the function defined in step 4.

* **Reshape Input Data:**

- Reshape the input data to be compatible with the LSTM input format.

* **Build and Compile LSTM Model:**

- Build a sequential LSTM model with multiple LSTM layers and a Dense layer.

- Compile the model using the Adam optimizer and mean squared error loss.

* **Train the Model:**

- Train the LSTM model using the prepared input and output data for a specified number of epochs and batch size.

* **Make Predictions:**

- Use the trained LSTM model to make predictions for future time steps based on the last observed values, 12 following month can be changed …

* **Inverse Transform:**

- Inverse transform the predicted scaled temperature data to obtain predictions in the original temperature scale.

* **Plot Results:**

- Visualize the original temperature data and the predicted values for future time steps using a plotting library (e.g., Matplotlib).

## Room for improvement :

At first, we went with the code screened above but as one might notice in the video showing the code working some tweaks have been made such as changing the batch size to 32 and going with and early stoppage function.

### Loss function :

The loss function stops improving with further epochs, an early stopping function can be introduced to the code, for optimizations.



The patience has been set to 10, such as for ten consecutive epochs the loss doesn’t improve the training stops.

### GPU acceleration :

If a GPU is at hand the training can be significantly speed up, in our case we had an mx450 in our disposal, training the model on the GPU instead of the CPU made quite the difference.

# Intel image classification :

## The model used : CNN

### What’s a CNN and why ? :

A Convolutional Neural Network (CNN) is a type of deep neural network designed specifically for tasks related to computer vision, particularly image analysis and recognition. CNNs are characterized by their ability to automatically and adaptively learn hierarchical feature representations directly from raw pixel data.

Key components of a CNN include:

* Convolutional Layers:

These layers apply convolution operations to the input data, which involves sliding a set of filters (kernels) over the input to detect features like edges, textures, or patterns.

* Pooling Layers:

Pooling layers downsample the spatial dimensions of the input by aggregating information. Common pooling operations include max pooling and average pooling.

* Activation Functions:

Activation functions, such as ReLU (Rectified Linear Unit), introduce non-linearity to the model, allowing it to learn complex relationships in the data.

* Fully Connected Layers:

At the end of the network, fully connected layers are often used for classification tasks. These layers connect every neuron to every neuron in the subsequent layer.

CNNs are particularly effective for image-related tasks due to their ability to automatically learn hierarchical features. They can identify low-level features like edges in early layers and progressively learn high-level features and spatial hierarchies in deeper layers.

**Why CNN for Image Classification:**

* Local Feature Learning:

CNNs are designed to automatically learn and identify local features in images, making them highly effective at capturing patterns and structures.

* Parameter Sharing:

CNNs use parameter sharing through convolutional kernels, allowing the network to learn a relatively small number of parameters that are reused across the entire image, making them computationally efficient.

* Translation Invariance:

The use of convolutional operations provides translation invariance, meaning that a learned pattern can be recognized in different parts of an image.

* Hierarchical Representation:

CNNs naturally learn hierarchical representations of features, where early layers capture simple features and deeper layers capture complex, abstract features.

* Reduced Sensitivity to Local Variations:

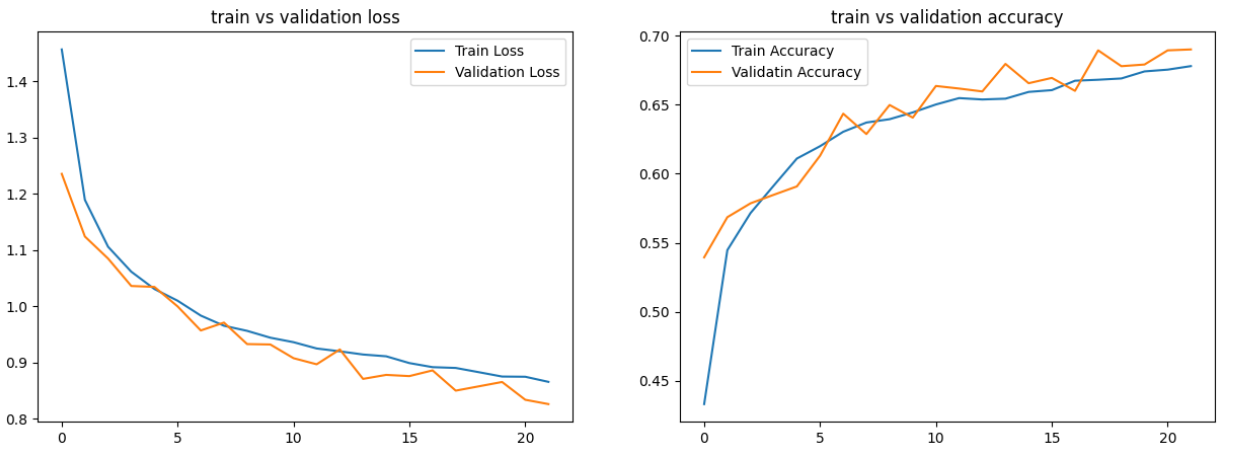
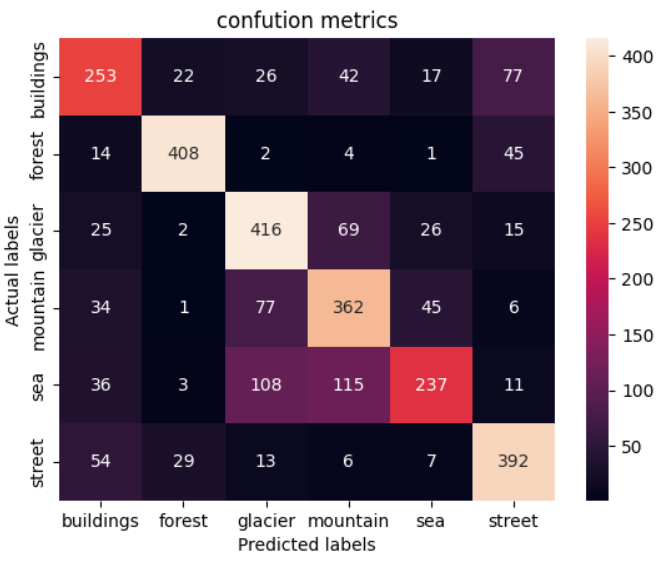
Pooling layers help reduce sensitivity to small variations in the input, making CNNs robust to changes in scale and orientation.

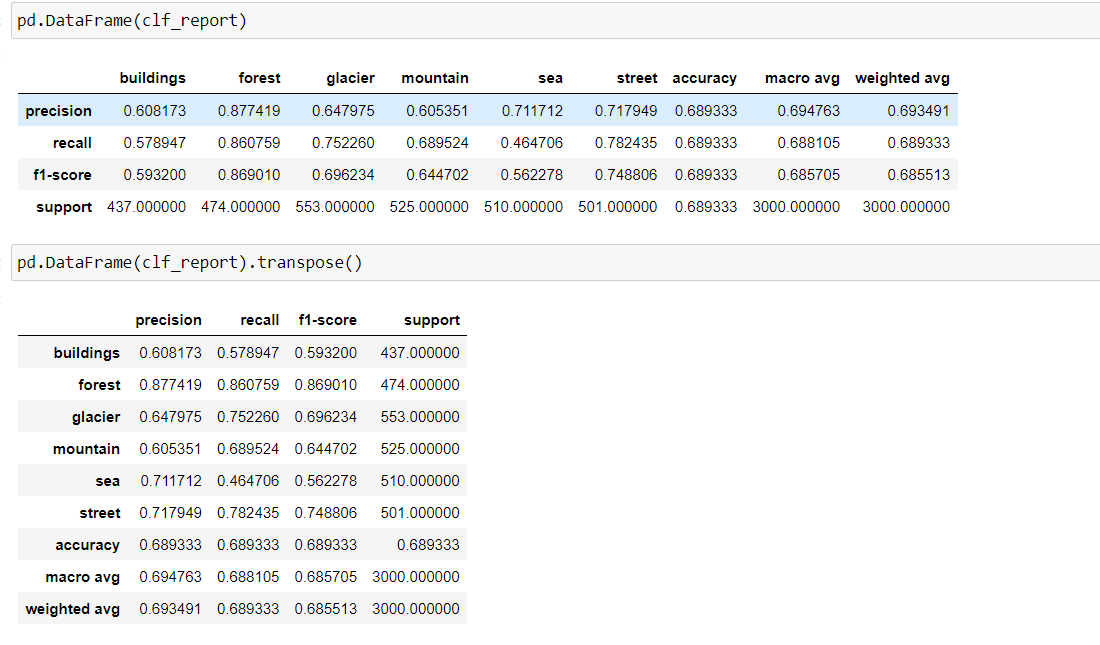
* State-of-the-Art Performance:

CNNs have consistently demonstrated state-of-the-art performance in various computer vision tasks, including image classification, object detection, and segmentation.

## The code : ( fully available in the video sent along this file)

## The results :





## Improvements and discussion :

This code is fully capable of giving satisfying results, it has been made by the help of the original code of Mohammed Hassan Ali <https://www.linkedin.com/in/mohamed-hassan-6b629715b>.

If one would want to further have more accurate results, we can add more epoch and not to forget to increase the “patience” for the early stoppage or the results would be the same, also we could test it by inserting random images from the web and seeing how the model can classify it, and lastly not forgetting the obvious, having a gpu proves to be very helpful, if one would’ve run this on cpu it would’ve took lots of time, the as a suggestion one can go for a Transfer learning model, one that has already been pre trained…