

# BACHELOR OF TECHNOLOGY

In

## Computer Science and Engineering

**Submitted by:**

Kusuma M

(2023443738) CSE-F G2



**SHARDA**  
**UNIVERSITY**  
*Beyond Boundaries*



**Department of Computer Science and Engineering**

**School of Computing Science & Engineering**

**Sharda University, Greater Noida**

**Jan-June 2026**

## Working of Code

### Overview

This document explains the **working of the notebook Fine\_tune-BLIP\_on\_an\_image\_captioning\_dataset.ipynb** in a clear, step-by-step way. The goal of the notebook is to **fine-tune a BLIP (Bootstrapped Language-Image Pretraining) model** so it can generate captions for images using a custom dataset.

The explanation is written for documentation and understanding, not just execution.

### 1. Set-up Environment

#### What happens here:

- Required libraries are installed and imported.
- The main libraries include:
  - transformers → to load BLIP model and processor
  - torch → for deep learning and training
  - datasets → to load and handle image-caption datasets
  - PIL → for image handling

#### Why this step is important:

- BLIP is a Hugging Face model, so transformers is essential.
- PyTorch provides tensor operations and backpropagation.
- Without this setup, model loading and training won't work.

### 2. Load the Image Captioning Dataset

#### What happens here:

- An image-caption dataset is loaded using the Hugging Face datasets library.
- Each data sample typically contains:
  - An **image**
  - A **caption** describing that image

### **Why this step is important:**

- Fine-tuning requires paired data (image + text).
- The dataset acts as ground truth so the model learns how images map to captions.

### **Conceptually:**

“This image should produce this caption.”

## **3. Create a Custom PyTorch Dataset**

### **What happens here:**

- A custom dataset class is created.
- For each data sample:
  - The image is processed (resized, normalized)
  - The caption is tokenized (converted into numbers)
- The BLIP **processor** handles both image and text together.

### **Why this step is important:**

- PyTorch models cannot directly read raw images or text.
- Everything must be converted into tensors.
- This step bridges raw data → model-ready data.

### **Behind the scenes:**

- Images → pixel tensors
- Text → token IDs

## **4. Load BLIP Model and Processor**

### **What happens here:**

- A pre-trained BLIP model is loaded.
- A BLIP processor is also loaded.

### **Why both are needed:**

- **Model** → does the learning and caption generation
- **Processor** → prepares image + text in the exact format the model expects

### **Key idea:**

- Instead of training from scratch, the model already understands images and language.
- Fine-tuning only adapts it to your specific dataset.

## **5. Train the Model**

### **What happens here:**

- Training parameters are defined:
  - Learning rate
  - Batch size
  - Number of epochs
- The model is trained using backpropagation.

### **Training loop logic:**

1. Model receives image + caption
2. It predicts a caption
3. Loss is calculated (difference between prediction and actual caption)
4. Gradients are computed
5. Model weights are updated

### **Why this step is important:**

- This is where learning actually happens.
- The model slowly improves caption quality over time.

## **6. Inference (Generating Captions)**

### **What happens here:**

- A new image (not from training data) is given to the model.

- The model generates a caption for it.

**Why this step is important:**

- Confirms whether fine-tuning worked.
- Shows real-world usability of the trained model.

**Conceptually:**

“Given only an image, can the model describe it correctly?”

## **7. Load Model from Hugging Face Hub**

**What happens here:**

- The fine-tuned model is saved and optionally pushed to Hugging Face Hub.
- The model can later be reloaded without retraining.

**Why this step is important:**

- Saves time and computation
- Makes the model reusable and shareable
- Useful for deployment or future experiments

## **Overall Workflow Summary**

1. Install and import libraries
2. Load image-caption dataset
3. Convert data into PyTorch format
4. Load pre-trained BLIP model
5. Fine-tune model on custom dataset
6. Test model using inference
7. Save and reload trained model

## **Key Takeaway**

This notebook demonstrates **transfer learning in multimodal AI**, where:

- Vision + Language models are adapted
- Pre-trained knowledge is reused
- Custom datasets improve task-specific performance

This makes BLIP powerful, efficient, and practical for real-world image captioning tasks.