# Landmark Detection Project

# Name:TANIKONDA SRIVALLIKA

A step-by-step Process for building a landmark detection project using PyTorch and the 300W-LP dataset:

## 1.Install PyTorch and other required libraries:

In [ ]:

```
pip install torch torchvision opencv-python pandas numpy matplotlib
```

## 2.Uploading the 300W-LP dataset

## 3.Extract the zip file to a convenient location on your local machine.

## 4.Load the dataset and preprocess the images and labels:

In [ ]:

```
import cv2
import os
import numpy as np
import pandas as pd

# Load training data
train_df = pd.read_csv(os.path.join('300W_LP', 'landmarks','full_train.csv'))
train_images = []
train_labels = []
for _, row in train_df.iterrows():
    img = cv2.imread(os.path.join('300W_LP', row['img']))
    img = cv2.resize(img, (224, 224))
    train_images.append(img)
    landmarks = np.asarray(row[1:], dtype=np.float32).reshape(-1, 2)
    train_labels.append(landmarks)

# Load validation data
val_df = pd.read_csv(os.path.join('300W_LP', 'landmarks','full_test.csv'))
val_images = []
val_labels = []
for _, row in val_df.iterrows():
    img = cv2.imread(os.path.join('300W_LP', row['img']))
    img = cv2.resize(img, (224, 224))
    val_images.append(img)
    landmarks = np.asarray(row[1:], dtype=np.float32).reshape(-1, 2)
    val_labels.append(landmarks)

# Convert data to PyTorch tensors
train_images = torch.tensor(np.array(train_images), dtype=torch.float32)
train_labels = torch.tensor(np.array(train_labels), dtype=torch.float32)
val_images = torch.tensor(np.array(val_images), dtype=torch.float32)
val_labels = torch.tensor(np.array(val_labels), dtype=torch.float32)

# Normalize pixel values
train_images /= 255.
val_images /= 255.
```

## 5.Define the CNN model architecture:

In [ ]:

```python
import torch.nn as nn

class LandmarkDetectionModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 64, kernel_size=3, padding=1)
        self.bn1 = nn.BatchNorm2d(64)
        self.relu1 = nn.ReLU()
        self.maxpool1 = nn.MaxPool2d(kernel_size=2)
        self.conv2 = nn.Conv2d(64, 128, kernel_size=3, padding=1)
        self.bn2 = nn.BatchNorm2d(128)
        self.relu2 = nn.ReLU()
        self.maxpool2 = nn.MaxPool2d(kernel_size=2)
        self.conv3 = nn.Conv2d(128, 256, kernel_size=3, padding=1)
        self.bn3 = nn.BatchNorm2d(256)
        self.relu3 = nn.ReLU()
        self.maxpool3 = nn.MaxPool2d(kernel_size=2)
        self.conv4 = nn.Conv2d(256, 512, kernel_size=3, padding=1)
        self.bn4 = nn.BatchNorm2d(512)
        self.relu4 = nn.ReLU()
        self.maxpool4 = nn.MaxPool2d(kernel_size=2)
        self.fc1 = nn.Linear(512 * 7 * 7, 1024)
        self.relu5 = nn.ReLU()
        self.dropout1 = nn.Dropout(0.5)
        self.fc2 = nn.Linear(1024, 136)

    def forward(self, x):
        x = self.conv1(x)
        x = self.bn1(x)
        x = self.relu1(x)
        x = self.maxpool1(x)
        x = self.conv2(x)
        x = self.bn2(x)
        x = self.relu2(x)
        x = self.maxpool2(x)
        x = self.conv3(x)
        x = self.bn3(x)
        x = self.relu3(x)
        x = self.maxpool3(x)
        x = self.conv4(x)
        x = self.bn4(x)
        x = self.relu4(x)
        x = self.maxpool4(x)
        x = x.view(-1, 512 * 7 * 7)
        x = self.fc1(x)
        x = self.relu5(x)
        x = self.dropout1(x)
        x = self.fc2(x)
        return x
```

## 6.Define the loss function and optimizer:

In [ ]:

```
import torch.optim as optim
```