**Report**

**Pick and Click**

**Abstract:**

The main aim of this project is to design a smart camera system for always-on edge devices. As we all know edge device (low power device) consume more energy and drain the battery very soon. So, to overcome this problem, we have dived the energy consumption in edge devices in 2 components

1. Sensing energy – Continuously sense data from real world
2. Transmission energy – Processing on the cloud

Overall goal of this project is to add machine learning concepts on the edge device to minimize sensing and transmission energy.

**Introduction:**

The Shot we take is not always the shot we want, so when we try to click pictures with the smart camera it will not click picture immediately instead it will sense series of images from the screen and then predicts the right frame from which we get the best shot of any object. At this point it trigger high resolution camera in order to capture the best image. To do this we have design classifier- regressor pipeline which predicts the frame with least obstruction or the best image of the scene.

**Dataset used:**

* Fruit 360 dataset
* Total number of images: 65000
* Training set size: 45000images
* Test set size: 17000 images
* Number of classes: 95 different types
* Image size: 100x100 pixels.

**Data Preparation**:

Original Dataset was not very related to our project, so we had to redesign dataset according to our requirements. We had to create stream of video from the images. To achieve maximum generality, we introduce 6 levels of randomness in dataset preparation

* We had to Create a white canvas of 100x100
* Resize the fruit image to a random size and place it at a random position in the canvas which was generated
* Create black-box of random size, which would be obstruction which moved randomly on the canvas
* Determine a random trajectory, which direction will the black box move.
* We made our trajectory to move from Left-to-right and made it to pick random angle (-45 to 45 degree)
* We had to determine a random speed for the trajectory, either fast or slow or medium Number of frames the obstacle takes to pass through the trajectory
* Move the black box of the trajectory to create the video

**Labelling the dataset:**

Since, we know the positions of the image and black-box the % of overlap can be ascertained analytically. We label each frame based on the % of overlap. We also add meta data for positions of image and black-box. Positoin of the image and the black box on the white canvas is measured on the x-axis and y-axis and also the width and the height and the height of the black box and the image is noted down into the csv folder which will be our label data set.

|  |  |
| --- | --- |
| **% overlap of the obstruction on the image** | **Label the class** |
| 50 + | 4 |
| 25 to 50 | 3 |
| 12.5 to 25 | 2 |
| 6 to 12.5 | 1 |
| 6 or less | 0 |

**Classifier Design:**

After get the accurate dataset we had to predicts the degree of overlap between the object and the black-box Since classification happens on a low-resolution image, to check the least obstruction or less overlap of the black box, we downsized the 100x100 input frame by 16 times. We applied different machine learning algorithms like Random forest, Decision Tree, Logistic Regression, Support vector classifier, Bagging Classifier to check accuracy. Among all the algorithm only Random forest has performed very well. With the accuracy score on training data set is 99% and test data set is 75%. When checked with confusion matrix for the misclassification, on training data set no misclassification and that fits perfectly well but on testing data set only few were misclassified. The classifier gets confused and classifies some images on the previous class or the neighborhood class.

We have also tried using Keras and CNN.

1. **Implementation using Keras:**

As we know that Keras is a powerful Python library for developing and evaluating deep learning algorithms. It wraps the efficient numerical computation libraries like TensorFlow and allows to define and train neural network models. We have just tried with the simple neural network. Basis steps followed are:

1. Load Data.

training dataset and testing dataset is loaded along with the label data file. No of rows, cols, channel(rgb) is initialization and also here will resize the images

1. Define Model.

create a Sequential model and add layers, set the input parameter to 1875 as per the no of cols.

* Fully connected layers are defined using the Dense class.
* first argument will be no of neurons in the layer
* initialization method as the second argument as init and the activation function
* network weights are in between 0 and 0.05 because that is the default uniform weight initialization in Keras.
* Relu activation function on the first two layers and the sigmoid function in the output layer.
* The first layer has 100 neurons and expects 1875 input variables.
* The second hidden layer has 50 neurons and third 20 neurons
* the output layer has 1 neuron for predication

1. Compile Model.

here We must specify the loss function to use to evaluate a set of weights. The optimizer is used to search through different weights for the network. I have used loss as “binary\_crossentropy.” and optimizer as “Adam”.

1. Fit Model and Evaluate it.

here we need define the no od epochs and “batch\_size” and This will generate a prediction for each input and output pair and collect scores, including the average loss and any metrics.

1. **Implementation using CNN:**

CNN are used for the image classification main aim is to implement Convolutional Neural Network (CNN) on the dataset, feature extraction technique is used. Now we have the images captured using camera, we need check whether it is giving correct output or not, applying the different Dense layers to it.  
fruits can be predicted.

**Sequence Predictor:**

Given the overlap percentages of the previous frame(s), predict the % overlap of the next frame. We use online linear regression to achieve this and also need to maintain a linear model within the sequence predictor, as we get new points, we refit the linear model and predict the next frame. If the predicted overlap is less than a threshold, we click the next frame. Best frame will be the earliest frame with the least % overlap in the video

We look at 2 metrics:

* Frame difference between the clicked frame and the best frame
* Difference in % overlap between the clicked frame and best frame
* PicknClick clicked within +/- 1 frame of the best frame for 74.9% of the videos
* The average % overlap difference between best frame and frame clicked is 5%





**Future Enhancement**:

More challenging trajectories like moving in all the different direction or in zic-zac pattern. Second-order projectiles, sinusoids etc. It takes a few frames for PicknClick to start accurately predicting %overlap of the next frame – if the least overlap occurs in the initial frames. LSTMs for sequence prediction