Implications of Image classifier enhancement for the video classification.

As per the discussion the experimentation was started with building the personal model that classifies images. It was built in my university days.

Now the main goal of the enhancement is:

1. Whether the classifier performs well on the videos when they are fragmented into frames and provided to the model?
2. How the personal model can be improved or modified?

This entire report is divided in two sections:

Section 1: Experimentation and Observation of model's output.

Section 2: Discussion and Future Scope

Section 1: Experimentation and observation.

For this experiment, both the dataset and the model, designed specifically for image classification, were already available. The dataset consisted of images sourced from Google Earth, with nearly 600 images of cities and forests uploaded to Google Drive. Of these, 400 images were used for training and 200 for testing, with each set containing an equal number of city and forest images.

The model architecture included three convolutional layers and three max pooling layers, arranged alternately. The output from the final max pooling layer was passed to a flatten layer, converting the tensor output into a list of inputs for subsequent layers, ultimately leading to the classified output the image below illustrates the actual structure of the architecture.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 16)      448

 max_pooling2d (MaxPooling2  (None, 127, 127, 16)      0
 D)

 conv2d_1 (Conv2D)           (None, 125, 125, 32)      4640

 max_pooling2d_1 (MaxPoolin  (None, 62, 62, 32)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 60, 60, 16)        4624

 max_pooling2d_2 (MaxPoolin  (None, 30, 30, 16)        0
 g2D)

 flatten (Flatten)           (None, 14400)             0

 dense (Dense)               (None, 256)               3686656

 dense_1 (Dense)             (None, 1)                 257

=================================================================
Total params: 3696625 (14.10 MB)
Trainable params: 3696625 (14.10 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

This can be viewed as “folding” the input space or creating new functions that are clipped and recombined at each layer.

Now moving further to enhancement, I have used two videos for the same, each of an approximately minute duration. Here are the below details of those videos:

Video name(As uploaded on Google drive)	Video Duration(in minutes)	Frames per second	The Entire video should be classified as
1_MinuteNEWYORK.mp4	1:00	25	City
1_minute_relaxing_video_with_nature_A_minute_with_nature_Flowing_River.mp4	0:54	26	Forest

Table1: Details of the video used in the classification

In order to extract video into frames, I have used open cv library and to perform preprocessing, I have used Tensorflow library. Using the function called video capture, each video is divided into frames depending on the frame rate I provide. After that, the frames are extracted and appended as list of tensors. Then each frame is provided to the model after which the model classifies each image and put the label as “City” or “Forests”. Post that, the labeled/edited frames are recombined to form a video using another function provided by open CV named as video writer.

Observations

So the observations are taken for different number of frames rate and according to each video. The video that I initially provided was the video of the city and the below table is the observations for each frame rate:

Frame rate(FPS)	Original frames per second (FPS)	Frame interval	Average inference time(ms)	Total Number of frames	Number of frames classified as cities	Number of frames classified as forests	Overall Accuracy
4	25	6	25	289	280	9	96%
16	25	1	10	1729	1495	234	86%
25	25	1	10	1729	1494	234	86%

Table 2: Readings for the video of city

Over here you can see as I am increasing the resolution (Frames Rate) the accuracy gets reduced. This is the final reading I have got. Before that some experimentation and modifications have been done which will be explained in the next section.

Now, the similar experimentation has been performed on the video of forests and here are the below readings for the same.

Frame rate(FPS)	Original frames per second (FPS)	Frame interval	Average inference time(ms)	Total Number of frames	Number of frames classified as cities	Number of frames classified as forests	Overall Accuracy
4	26	6	10.43	235	171	64	27%
16	26	1	11.46	1409	1033	376	26.68%
26	26	1	10.46	1409	1033	376	26.68%

Table 3: Readings for the video of Forest

From the above reading it is clear indication that the model is classifying the videos of the city properly, but it performs worse when it comes to classifying the data of forests.

The implications now can be explored in the next section.

Section 2: Implications, Discussions and the future scope.

After training the model with existing data, it came under the picture that the model is not performing well. Here are the performance parameters for each model:

Video Name	Frames per second	Accuracy for Forests	Accuracy for Cities
1_MinuteNEWYORK.mp4	25	NA	79%
1_minute_relaxing_video_with_nature_A_minute_with_nature_Flowing_River.mp4	26	15%	NA

Table 4: Performance of Model for different videos after training on existing dataset.

After the deep investigation, I came to realize that the data should be more diversified. Hence I explored and extracted other images of the city(for this particular experiment, I chose two cities other than the cities of which I have the data i.e. Glasgow and New York). For Forests data, I followed the same process where I took the images of the forests from United States and United Kingdom. By keeping this in consideration, I am showing the breakdown of training and testing datasets through the below table.

Number of Images from the city of Glasgow, UK	Number of Images from the city of New York, USA	Number of Images from the city of Accra, Ghana	Number of Images from the city of Pune, India
50	50	50	50

Table 5: Number of images from different Cities for training data

Number of Images from the UK Forests	Number of Images from the American Forests	Number of images from African Forests	Number of Images from Indian Forests
50	50	50	50

Table 6: Number of images from different Forests for training data

By keeping the above breakdown in consideration, overall, there are 400 images provided as the training data.

The below table shows the breakdown for testing data:

Number of Images from Stirling city	Number of Images taken from Scottish forests
100	100

Table 7: Number of images from Stirling city and Scottish Forests

So overall, there are 200 Images provided as the testing data.

I tried training and testing this model on the updated dataset. The results are as per the below table which shows a slight improvement in the performance.

Video Name	Frames per second	Accuracy for Forests	Accuracy for Cities
1_MinuteNEWYORK.mp4	25	NA	83%
1_minute_relaxing_video_with_nature_A_minute_with_nature_Flowing_River.mp4	26	20%	NA

Table 8: Model's performance after training with the new dataset

For a moment, I thought my model is over fitting. To address this problem, I decided to change the optimizer from stochastic gradient descent (SGD) to Adam. The reason behind doing this is in SGD, model's parameters are updated based on the single data points or small batch of data points and it takes the random path to converge towards optimal parameters. As the path to reach local minima is random the time required for the model to learn the data will be more compared to Adam(Adaptive moment estimation) which incorporates momentum to accumulate past gradients and accelerate convergence towards optimal parameters. Plus it also integrates with RMSprop (Root Mean Squared propagation) technique for adapting the learning rate for each parameter based on historical gradients to address issues with sparse or noisy gradients.

Along with this, I changed loss function from categorical cross entropy to binary cross entropy as the type of problem was of binary classification. After making necessary changes, this is the result I have got:

Video Name	Frames per second	Accuracy for Forests	Accuracy for Cities
1_MinuteNEWYORK.mp4	25	NA	86%
1_minute_relaxing_video_with_nature_A_minute_with_nature_Flowing_River.mp4	26	26%	NA

Table 9: Model's performance after changing the optimizer and loss function

Other techniques to avoid over fitting were data pre processing for instance normalizing images and resizing it.

Further discussion involves following subsections.

2.1. Flaws of the model

Despite providing diversified data, there were some flaws involved. The following points are the flaws of the model which can be mitigated in the future.

Biasness: After analyzing the figures and outputs of the models, it became evident that the model was more biased towards classifying cities than forests. There were several exceptional frames of forests that the model failed to classify correctly. This occurred because these frames were novel to the model, and many forest frames contained sunlight. The training dataset comprised images of forests with little to no sunlight,

leading the model to learn that frames with no sunlight and trees were indicative of forests. The primary reason for this bias is detailed in the next section.

Amount of data: Due to time constraints, I was unable to gather more diversified data. With a larger and more varied dataset, the model could have been more adept at handling different kinds of videos, which brings us to the next point.

Less or zero flexibility to new videos: Experimented on different sorts of videos related to forests and cities but the result for the model was same as mentioned in first flaw.

Processing time and memory: Although I managed to train and test the models on the updated dataset, and save the new model to implement it on video classification, as I started increasing the frame rate, the pipeline was consuming more RAM. Nearly 80% of the RAM used to be consumed just because I was performing video classification.

2.2. Knowledge requirements

When it come to knowledge requirements, I think learning more about Open cv and Tensor flow framework would be more beneficial. It is because I knew I can do lots of image processing techniques using the Open CV and tensorflow, but video processing and classification using these libraries was new to me.

2.3. Further Enhancement

Further enhancements for this project include expanding and diversifying the dataset to mitigate model bias. The personal model I built utilizes a Convolutional Neural Network (CNN), which performs convolution operations on each pixel. This process has a computational complexity of $O(N^2)$, consuming significant memory.

By employing the Fast Fourier Transform (FFT), we can optimize this. Given two long number series for convolution, we first compute their FFT, then multiply the results pointwise, and finally perform the inverse FFT. This approach reduces the number of operations to O(N log N), thereby saving memory and processing time. Implementing the FFT algorithm instead of the conventional convolution method would be more efficient.

Additionally, this project can be enhanced by using semantic segmentation to identify cities based on building structures. Another improvement could involve using R-CNN to detect and classify objects characteristic of forests and cities.

2.4. Pipeline improvement:

For pipeline improvement I think we can download the pre-trained model and fine tune it for instance LeNet(I wont say alexnet because the computation time for the same is exceptionally huge.). We can even build small model and implement transfer learning algorithm which will also save a lot of memory and computation time.

2.5. Use Case:

As mentioned in enhancement, this classification can be further expanded from binary to categorical, where model can identify cities based on the structure of building using semantic segmentation. This can help us to find the missing person if he/she lost in particular city For example, if ABC person lost in XYZ city, the drone with the computer vision model can be sent which will first Identify the XYZ city and then find the person.