BVRIT HYDERABAD
College of Engineering for Women

Department of Computer Science and Engineering

A Deep Learning based System for Landmark and tourist place recognition

Under the Guidance of:
Name: Ms. S.Vidyullata
Designation: Assistant Professor

Team No: C_5
19WH1A05F0- S.Meghana
19WH1A05G1 – G. Varsha priya
20WH5A0517- S. Malavika

VISHNU
UNIVERSAL LEARNING

AGENDA

- Introduction
- Existing system
- Problem statement
- Proposed system
- Tools and Technology
- Societal impact
- Project timeline
- References

INTRODUCTION

- More and more information about tourist attractions is being portrayed visually rather than textually.
- As a result, tourists who are interested in a specific attraction represented in photographs may not know how to conduct a text search to learn more about the intriguing tourist places.
- In light of this issue, and in order to improve the tourism industry's competitiveness, this study proposes an innovative tourist spot identification mechanism based on deep learning-based object detection technology for real-time detection and identification of tourist spots by taking pictures on location or retrieving images from the Internet.

EXISTING SYSTEM

- Currently most of the searches are about tourist are done using Text. Although there are several instances where text information is not available and or is not enough.
- There are several Image based system which can identify tourist spots using Images which uses algorithms like SVM which are not accurate and are not well suited.

VISHNU
UNIVERSAL LEARNING

DISADVANTAGES OF EXISTING SYSTEM

- Support vector machine algorithm is not acceptable for large data sets.
- It does not execute very well when the data set has more sound i.e. target classes are overlapping.
- In cases where the number of properties for each data point outstrips the number of training data specimens, the support vector machine will underperform.

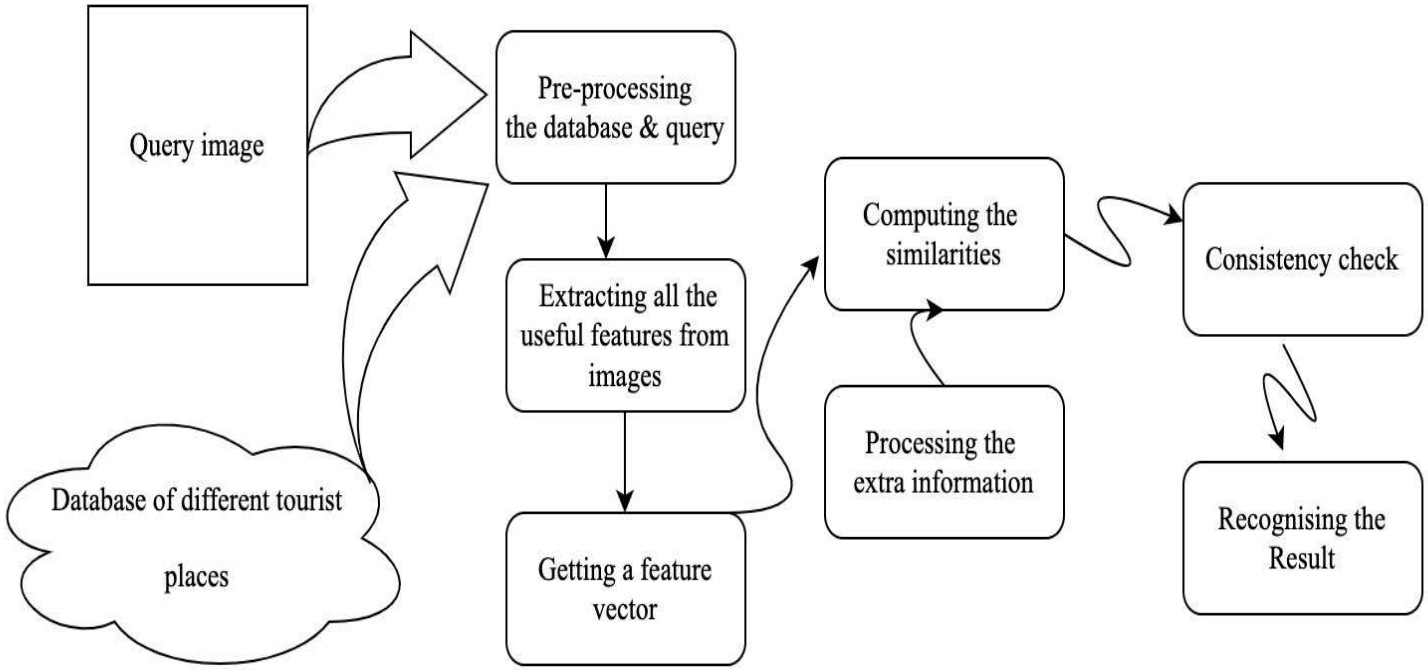
PROBLEM STATEMENT

This project creates a tourist spot recognition system, which is a Deep Learning AI framework that is used to identify tourist destinations by providing photographs and Images.

PROPOSED SYSTEM

- We propose a Advance Deep Learning System which will be able to detect the Name and Details about a Tourist Spot just by using the Image provided. Currently the System will be trained to recognise some major Tourist Spots with a Web App based intuitive Interface.

PROPOSED SYSTEM FLOW CHART



Data Normalization

- Step 2 - Data normalization

```
images = images.astype(np.float16)
labels = labels.astype(np.int8)
images /= 255
print("Images shape after normalization = ",images.shape)
```

Python

VISHNU
UNIVERSAL LEARNING

Splitting Dataset for testing and training

- Split dataset for training and testing

```
x_train, x_test, y_train, y_test = train_test_split(images, labels, test_size = 0.2, random_state = RANDOM_SEED)


print("x_train shape = ",x_train.shape)
print("y_train shape = ",y_train.shape)
print("\nx_test shape = ",x_test.shape)
print("y_test shape = ",y_test.shape)
```

Python

```
x_train shape =  (14400, 224, 224, 3)
y_train shape =  (14400,)

x_test shape =  (3600, 224, 224, 3)
y_test shape =  (3600,)
```

```
del images
del labels
```

Python

```
display_random_images(x_train, y_train)
```

Python

VISHNU
UNIVERSAL LEARNING

Epochs and Batch Size

```
EPOCHS = 30
BATCH_SIZE = 32 #32
```

[ ] Python

VISHNU
UNIVERSAL LEARNING

Initializing the Resnet-50 V2 model

RESNET 50 V2

```python
resnet_50_v2 = ResNet50V2(input_shape=IMAGE_SIZE , weights='imagenet', include_top=False)
```

Python

```python
#do not train the pre-trained layers of VGG-19
for layer in resnet_50_v2.layers:
    layer.trainable = False
```

Python

```python
x = Flatten()(resnet_50_v2.output)

x = Dense(100, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.25)(x)
x = Dense(100, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.25)(x)
# x = Dense(100, activation='relu')(x)
# x = BatchNormalization()(x)
# x = Dropout(0.25)(x)

#adding output layer.Softmax classifier is used as it is multi-class classification
prediction = Dense(TOTAL_CATEGORIES, activation='softmax')(x)
resnet_50_v2_model = Model(inputs=resnet_50_v2.input, outputs=prediction)
# view the structure of the model
resnet_50_v2_model.summary()
```

Python

VISHNU
UNIVERSAL LEARNING

Continued…

```python
x = Flatten()(resnet_50_v2.output)

x = Dense(100, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.25)(x)
x = Dense(100, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.25)(x)
# x = Dense(100, activation='relu')(x)
# x = BatchNormalization()(x)
# x = Dropout(0.25)(x)

#adding output layer.Softmax classifier is used as it is multi-class classification
prediction = Dense(TOTAL_CATEGORIES, activation='softmax')(x)
resnet_50_v2_model = Model(inputs=resnet_50_v2.input, outputs=prediction)
# view the structure of the model
resnet_50_v2_model.summary()
```

Python

```python
resnet_50_v2_model.compile(loss='sparse_categorical_crossentropy', optimizer="adam", metrics=['acc'])
```

Python

```python
#Early stopping to avoid overfitting of model
early_stop=EarlyStopping(monitor='val_accuracy',mode='max', verbose=1, patience=15, restore_best_weights=True)
```

Python

```python
# fit the model
resnet_50_v2_history = resnet_50_v2_model.fit(x_train, y_train, validation_data=(x_test, y_test), steps_per_epoch = x_train.shape[0]//BATCH_SIZE, ep
```

Python

VISHNU
UNIVERSAL LEARNING

Training the Model

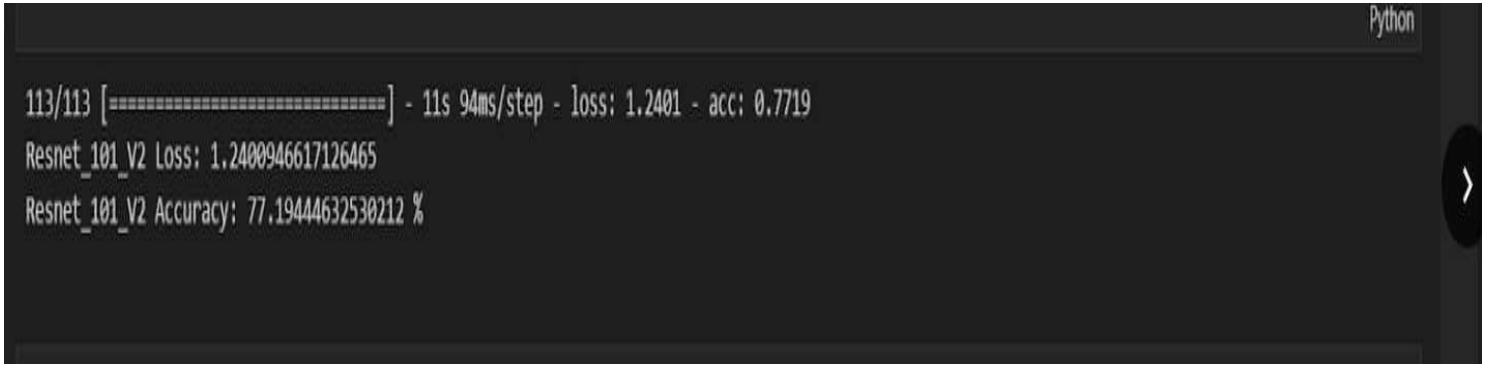
```
resnet_101_v2_history = resnet_101_v2_model.fit(x_train, y_train, validation_data=(x_test, y_test), steps_per_epoch = x_train.shape[0]//BATCH_SIZE,
```

Python

```
Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/30
450/450 [==============================] - 72s 127ms/step - loss: 1.3459 - acc: 0.6221 - val_loss: 0.9088 - val_acc: 0.7483
Epoch 2/30
450/450 [==============================] - 51s 114ms/step - loss: 0.2547 - acc: 0.9449 - val_loss: 0.6947 - val_acc: 0.7949
Epoch 3/30
450/450 [==============================] - 51s 114ms/step - loss: 0.0614 - acc: 0.9924 - val_loss: 0.6881 - val_acc: 0.8008
Epoch 4/30
450/450 [==============================] - 51s 114ms/step - loss: 0.0247 - acc: 0.9985 - val_loss: 0.6652 - val_acc: 0.8103
Epoch 5/30
450/450 [==============================] - 51s 114ms/step - loss: 0.0125 - acc: 0.9992 - val_loss: 0.6368 - val_acc: 0.8200
Epoch 6/30
450/450 [==============================] - 51s 114ms/step - loss: 0.0114 - acc: 0.9989 - val_loss: 0.8078 - val_acc: 0.7821
Epoch 7/30
450/450 [==============================] - 51s 114ms/step - loss: 0.0696 - acc: 0.9809 - val_loss: 1.3345 - val_acc: 0.6892
Epoch 8/30
450/450 [==============================] - 51s 114ms/step - loss: 0.1462 - acc: 0.9528 - val_loss: 0.9478 - val_acc: 0.7536
Epoch 9/30
450/450 [==============================] - 51s 114ms/step - loss: 0.0581 - acc: 0.9835 - val_loss: 0.8396 - val_acc: 0.7796
Epoch 10/30
450/450 [==============================] - 51s 114ms/step - loss: 0.0303 - acc: 0.9924 - val_loss: 0.8465 - val_acc: 0.7824
Epoch 11/30
450/450 [==============================] - 51s 114ms/step - loss: 0.0163 - acc: 0.9969 - val_loss: 0.8241 - val_acc: 0.7921
Epoch 12/30
450/450 [==============================] - 51s 114ms/step - loss: 0.0112 - acc: 0.9975 - val_loss: 0.8041 - val_acc: 0.7969
Epoch 13/30
...
Epoch 29/30
450/450 [==============================] - 51s 114ms/step - loss: 0.0111 - acc: 0.9969 - val_loss: 1.1623 - val_acc: 0.7810
Epoch 30/30
450/450 [==============================] - 51s 114ms/step - loss: 0.0159 - acc: 0.9955 - val_loss: 1.2453 - val_acc: 0.7709
```

VISHNU
UNIVERSAL LEARNING

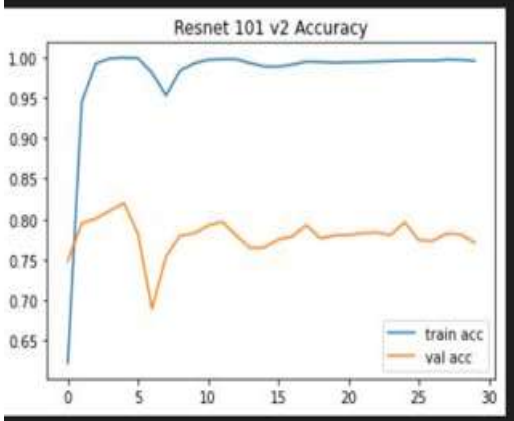
Model Evaluation



VISHNU
UNIVERSAL LEARNING

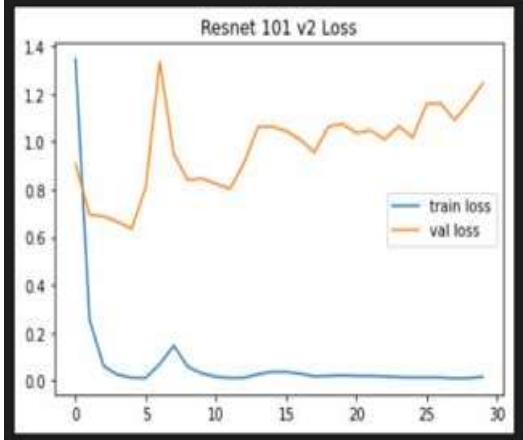
Model Accuracy and Loss

```python
# accuracies
plt.plot(resnet_101_v2_history.history['acc'], label='train acc')
plt.plot(resnet_101_v2_history.history['val_acc'], label='val acc')
plt.title("Resnet 101 v2 Accuracy")
plt.legend()
plt.show()
```




```python
# loss
plt.plot(resnet_101_v2_history.history['loss'], label='train loss')
plt.plot(resnet_101_v2_history.history['val_loss'], label='val loss')
plt.title("Resnet 101 v2 Loss")
plt.legend()
plt.show()
```

```python
# loss
plt.plot(resnet_101_v2_history.history['loss'], label='train loss')
plt.plot(resnet_101_v2_history.history['val_loss'], label='val loss')
plt.title("Resnet 101 v2 Loss")
plt.legend()
plt.show()
```




```python
#predict
y_pred=resnet_101_v2_model.predict(x_test)
y_pred=np.argmax(y_pred,axis=1)
#get classification report
print(classification_report(y_pred,y_test))
```

VISHNU
UNIVERSAL LEARNING

Evaluation Metrics

Output exceeds the size limit. Open the full output data in a text editor

	precision	recall	f1-score	support
0	0.53	0.65	0.59	92
1	0.62	0.61	0.62	118
2	0.86	0.88	0.87	123
3	0.73	0.57	0.64	126
4	0.78	0.92	0.84	111
5	0.92	0.75	0.83	127
6	0.86	0.84	0.85	116
7	0.99	0.99	0.99	110
8	0.97	0.62	0.76	174
9	0.65	0.64	0.64	113
10	0.79	0.78	0.78	129
11	0.68	0.65	0.66	117
12	0.55	0.58	0.57	116
13	0.82	0.88	0.85	112
14	0.99	0.96	0.98	136
15	0.72	0.82	0.77	131
16	0.87	0.86	0.87	136
17	0.74	0.88	0.80	95
18	0.82	0.82	0.82	120
19	0.70	0.94	0.80	93
20	0.82	0.73	0.77	142
21	0.56	0.81	0.66	94
22	0.83	0.81	0.82	106
...				
accuracy			0.77	3600
macro avg	0.77	0.78	0.77	3600
weighted avg	0.79	0.77	0.77	3600

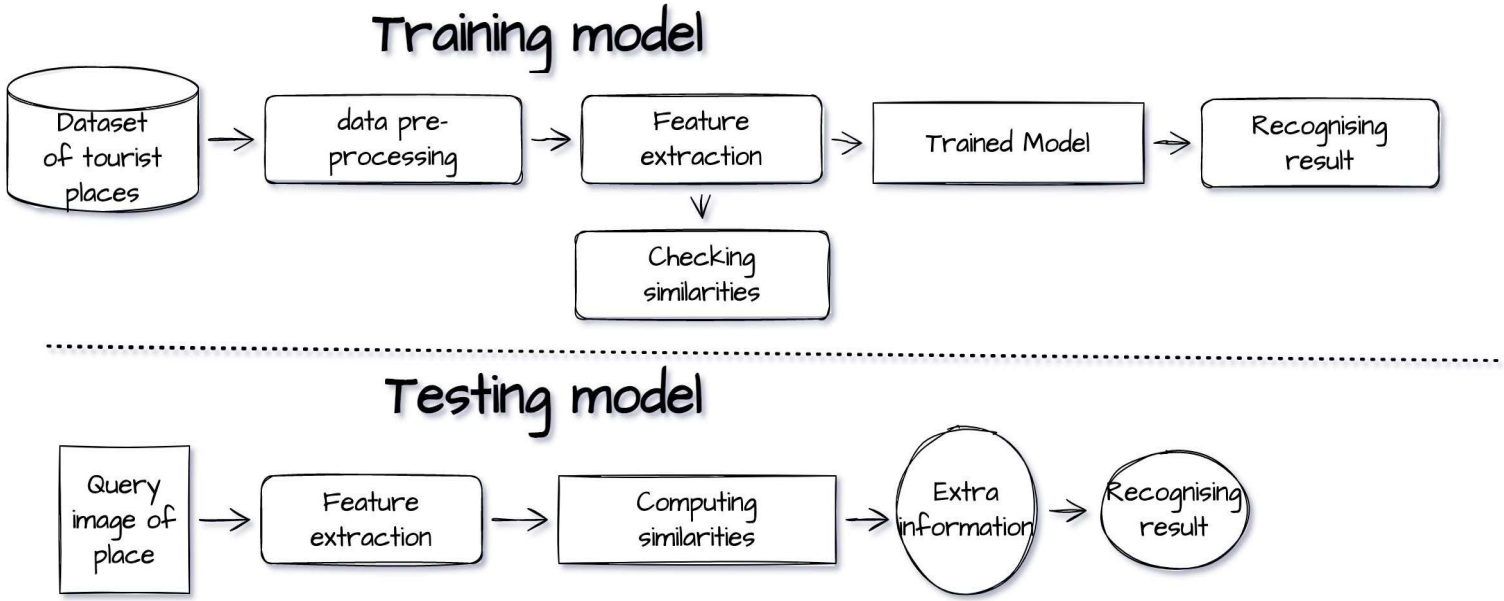
VISHNU
UNIVERSAL LEARNING

Confusion Matrix

```
print(confusion_matrix(y_pred,y_test))

resnet_101_v2_model.save("/content/drive/MyDrive/resnet_101_v2_model.h5", save_format="h5")
```

```
Output exceeds the size limit. Open the full output data in a text editor
[[ 60   3   0   0   3   1   0   0   0   1   0   0  16   3   0   1   0   1
    0   0   1   0   0   0   1   1   0   0   0   0]
 [  7  72   5   2   2   0   0   0   0   3   0   0  10   2   0   0   1   0
    0   0   2   3   3   0   3   2   0   0   1   0]
 [  0   2 108   0   0   0   0   0   1   0   0   0   0   1   0   0   0   0
    3   0   1   1   1   1   4   0   0   0   0   0]
 [  2   4   0  72   1   1   0   0   0   5   3   0   2   3   0   0   2   1
    0   7   3   4   1   0   0   1   1  11   1   1]
 [  2   1   0   0 102   0   0   0   1   0   0   1   0   0   0   0   0   1
    0   0   0   0   2   0   0   1   0   0   0   0]
 [  8   2   0   0   1  95   0   1   0   0   1   0   5   0   0   0   3   1
    0   1   0   5   0   0   2   0   0   0   1   1]
 [  0   0   0   0   0   0  97   0   1   1   0   0   0   0   0   6   0   7
    0   0   0   1   0   0   0   0   0   2   0   1]
 [  0   0   0   0   0   0   0 109   0   0   0   0   0   0   0   0   0   0
    0   0   0   1   0   0   0   0   0   0   0   0]
 [  0   1   0   0   1   1   9   0 108   2   2   8   1   2   0  11   0   5
    2   0   0   4   2   1   1   0   2   9   1   1]
 [  7   4   2   3   4   0   0   0   0  72   3   2   4   0   0   1   2   1
    0   5   0   1   1   0   0   0   0   1   0   0]
 [  0   0   0   0   0   0   0   0   0   2 100   1   0   0   0   0   0   0
    3   4   0   2   0   0   0   0   0   5   9   3]
 [  0   1   1   2   1   0   0   0   0   5   1  76   1   1   0   2   1   1
    0   4   0   2   0   0   0   2   0   8   0   8]
 [ 17   9   1   1   4   2   0   0   0   1   0   2  67   1   0   2   2   0
...
 [  0   1   1   4   0   0   0   0   0   6  10   6   1   2   0   0   0   1
    3   2   0   2   0   4   1   5   0  13 113   8]
 [  0   3   0   2   0   0   0   0   0   2   0   2   0   0   0   0   0   0
    3   2   0   3   0   0   1   3   0   5   0  79]]
```

PROPOSED SYSTEM ARCHITECTURE



Modules

- Tensorflow
- Keras
- Openc CV
- Sklearn
- CUDA

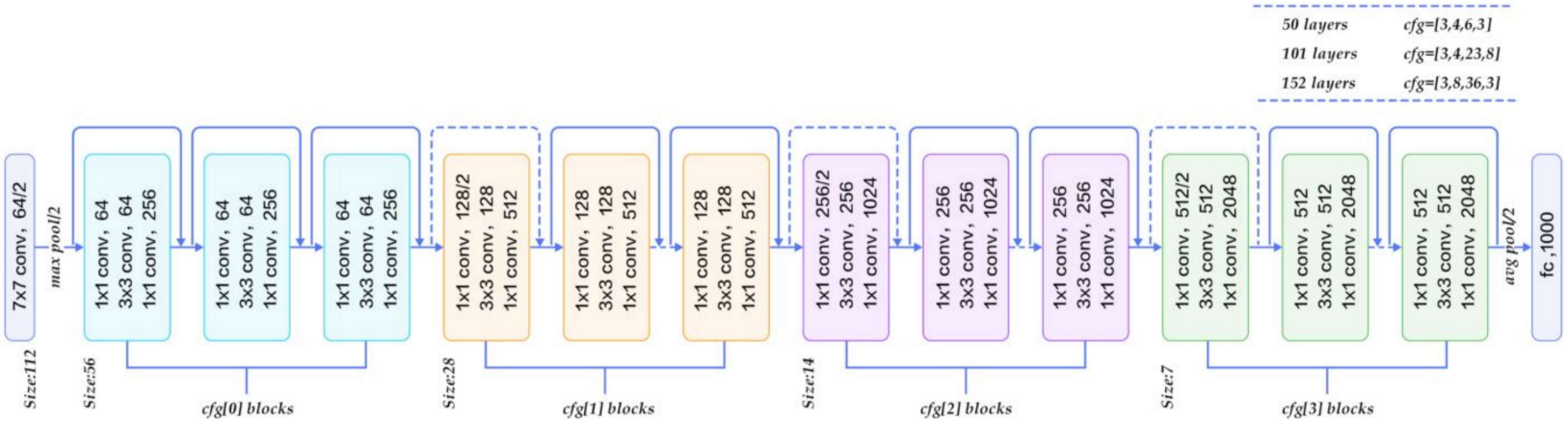
```
Python

113/113 [==============================] - 11s 94ms/step - loss: 1.2401 - acc: 0.7719
Resnet_101_V2 Loss: 1.2400946617126465
Resnet_101_V2 Accuracy: 77.19444632530212 %
```

Result

Proposed Model Accuracy:

Model 1:
Resent 50-v2 – 71.07%
Model 2:
Resnet 101-v2-77.1 %

METHODOLOGY



Deep residual networks like the popular ResNet-50 model is a convolutional neural network (CNN) that is 50 layers deep. A Residual Neural Network (ResNet) is an ANN of a kind that stacks residual blocks on top of each other to form a network.

DATA SET

Source: Dataset contains torist spot images

Number of Images: 30000

Number of Classes: 30

CSV File: .csv file contains information about the classes

TECHNOLOGY STACK

HARDWARE REQUIREMENTS

Processor: i5 10th Gen or better

RAM: 8GB or better

Storage: 120GB or more

SOFTWARE REQUIREMENTS

Windows 10

Python 3.6 or newer

Necessary Python Modules

SOCIETAL IMPACT

- Now a days more and more information about tourist attractions is being portrayed visually rather than textually. As a result, tourists who are interested in a specific attraction represented in photographs may not know how to conduct a text search to learn more about the intriguing tourist places.
- In light of this issue, and in order to improve the tourism industry's competitiveness, this study proposes an innovative tourist spot identification mechanism based on deep learning-based object detection technology for real-time detection and identification of tourist spots by taking pictures on location or retrieving images from the Internet.

Feasibility Study

TECHNICAL STUDY:

- Dataset is trained using Resnet50v2 model and Resnet101v2 which need to be supported by system which has > i3 processor and graphics card.
- Flask frame work is used for deploying model and showing prediction

OPERATIONAL STUDY:

- User friendly interface to interact with user to predict result.
- Prediction time will be reduced.

PROJECT TIMELINE

Date	Duration	Task
25 - 09 - 2022	1 week	• Introduction/ Abstract • Specifications • Literature Survey • References
19 - 11 - 2022	6 weeks	• Finding out suitable algorithm. • Data Set Selection. • Literature Survey
22 – 12 - 2022	4 weeks	• Literature Survey • Dataset Collection • Segregating dataset into respective classes • Pre-processing the dataset • Splitting the dataset into testing and training modules

Project Timeline Contd..

16-03-2023	5 weeks	• Training the model • Modifying different input parameters for improving accuracy • Tried different epochs number and batch size
		• Severity prediction • Paper publishing

REFERENCES

[1] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z., 2016. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).

[2] Szeliski, R., 2010. Computer vision: algorithms and applications. Springer Science & Business Media.

[3] Umbaugh, S.E., 2010. Digital image processing and analysis: human and computer vision applications with CVIPtools. CRC press.

[4] Alsing, O., 2018. Mobile object detection using tensorflow lite and transfer learning.

[5] Gada, S., Mehta, V., Kanchan, K., Jain, C. and Raut, P., 2017, December. Monument recognition using deep neural networks. In 2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)

VISHNU
UNIVERSAL LEARNING

THANK YOU