CSE 465 PROJECT PRESENTATION

TITLE: Image Recognition using deep learning Convolutional Neural Network (CNN) model.

Kaggle Dataset:
https://www.kaggle.com/pavansanagapati/images-dataset

NAME: AURIK ANJUM NOSHIN
NSU ID: 1712557642
COURSE & SECTION: CSE 465.3

1. What is the project?

The project is about Image Recognition using CNN model by training and evaluating the model.

1.1 Why is it interesting?

The dataset has some unique features that we can work on. There are 7 types of images.

Dataset Size: 752.19 MB; 7 Categories ; 1803 files(.jpg, .png, .bmp) [FIGURE:01]

The project aims to recognize from the dataset's cats, bikes, cars, dogs, humans etc section and increase accuracy of recognizing then accordingly.

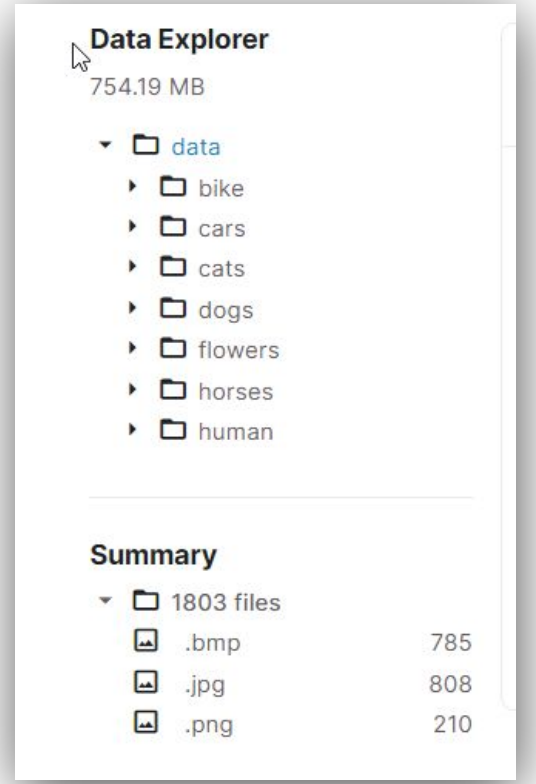

FIGURE: 01 DATASET FEATURES

How we solved the problem

```
X_train shape = (1442, 128, 128, 1)
X_test shape = (361, 128, 128, 1)
```

FIGURE: 02

We loaded the following datasets:

['horses', 'cars', 'dogs', 'flowers', 'bike', 'cats', 'human']

We assigned labels and defined number of classes(7 classes, labels from [0] to [6]). Next, we shuffle and split the dataset using *shuffle(img_data)* and *train_test_split*. [FIGURE:02] Then we print the *x_train* and *x-test* shape and plot image using *plt.imshow(image)* [FIGURE:03] *cat.171.jpg*

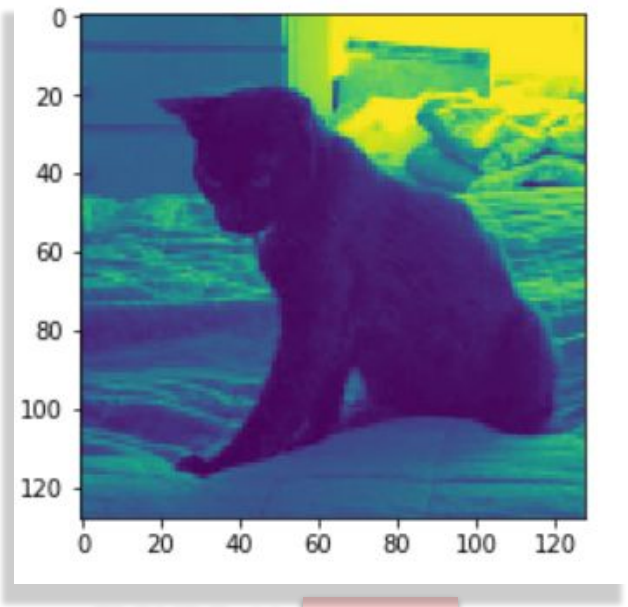

FIGURE: 03 cat.171.jpg

Training CNN model in Keras and graph demonstrations

1. Now we train a *CNN* model in *Keras*.
2. As activation function, we use *RELU* as it is widely implemented.
3. We then compile the model using *cnn_model.compile*. We can take a look at our CNN model summary on [FIGURE:04].
4. Now we start training, we set *num_epoch=100* which takes around 2 minutes and 43 seconds.
5. Next, we plot Train Loss vs Validation Loss in a graph. [FIGURE:05]

```
cnn_model.summary()
```

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 128, 128, 32)	320
conv2d_2 (Conv2D)	(None, 126, 126, 32)	9248
max_pooling2d_1 (MaxPooling2	(None, 63, 63, 32)	0
dropout_1 (Dropout)	(None, 63, 63, 32)	0
flatten_1 (Flatten)	(None, 127008)	0
dense_1 (Dense)	(None, 128)	16257152
dropout_2 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 7)	903

```
Total params: 16,267,623
Trainable params: 16,267,623
Non-trainable params: 0
```

FIGURE: 04 CNN model summary

FIGURE: 05 TRAIN LOSS vs VALIDATION LOSS

Image Testing

1. We also plot Train Accuracy vs Validation Accuracy in a graph [FIGURE:06]
2. Then we test an image and predict the probability of this image belonging to its own class [FIGURE:07] Database file Name: *carsgraz_244.bmp*
3. We plot another image of a different class [FIGURE:08]. Database file Name: *bike_331.bmp*

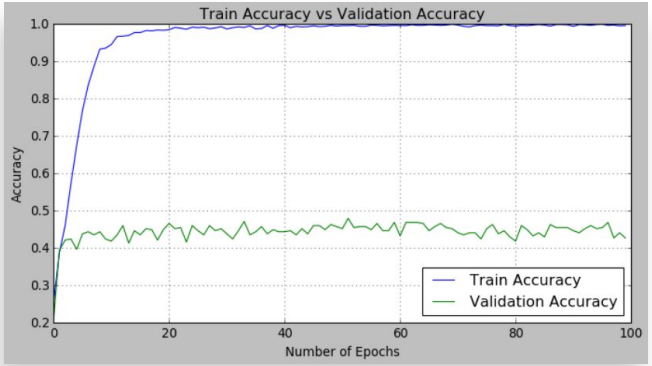

FIGURE: 06 TRAIN ACCURACY vs VALIDATION ACCURACY

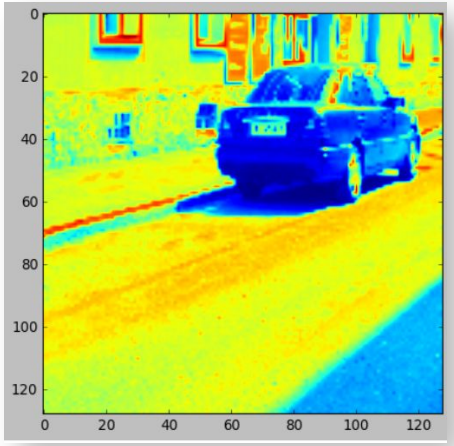

FIGURE: 07:
carsgraz_244.bmp

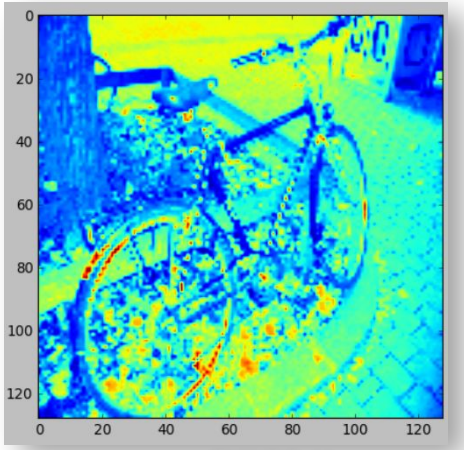

FIGURE: 08:
bike_331.bmp

Visualizing Intermediate layer output of CNN

1. Now we visualize the intermediate layer output of CNN, the image processing is from [FIGURE:08] *bike_331.bmp*
2. This visualization is show on [FIGURE:09] and also [FIGURE :10]

FIGURE: 10: visualizing the intermediate layer output of CNN of the file *bike_331.bmp*

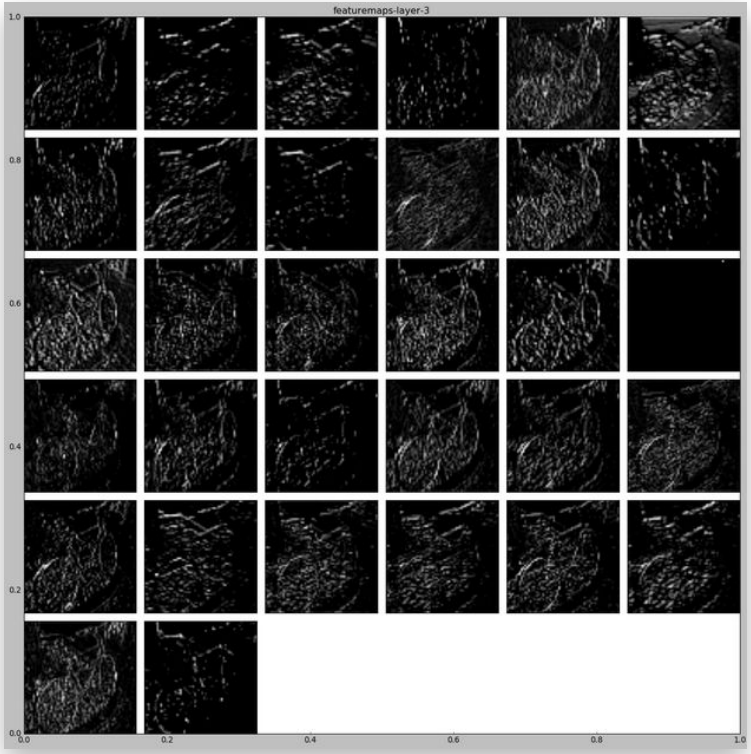

FIGURE: 09: visualizing the intermediate layer output of CNN of the file *bike_331.bmp*