STAIRCASE IMAGE RECOGNITION WITH CONVOLUTIONAL NEURAL NETWORKS

BRANDON TANG, REIDEN ONG
Temasek Junior College

ABSTRACT

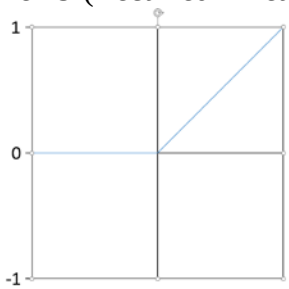
Staircase image recognition has applications in robotics, enabling robots to detect and classify staircases allows robots to plan their gait to climb up the stairs effectively. A popular way to perform object image recognition is to utilise convolutional neural networks. In this paper, we compare the accuracy and runtimes of 4 different pre-trained convolutional neural networks to investigate which would be best to use for real world application in a staircase cleaning robot. We found that for the 4 models we tried, their ranking for accuracy was the inverse of their ranking for prediction speed. Ranked by accuracy, the models are: Inception-Resnet v2, Inception v3, ResNet 50, MobileNet v2. For real world use, we recommend Inception-v3 with its relatively high accuracy and short prediction times.

INTRODUCTION

A neural network (NN) is a data structure which is used to determine relationships in a set of data by training the network with a established dataset, then allowing it to make predictions with new sources of data. It is a popular form of supervised machine learning.

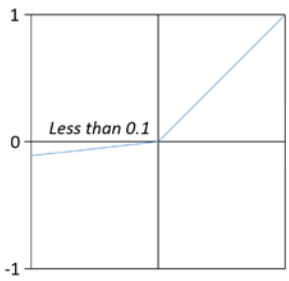
The basic structure of a NN is a directed, layered graph where each node (i) takes input from a set of neurons from the previous layer (J). Each neuron then calculates its activation value by applying a function on the summation of the product of the output values (a_j) of the nodes in J the weight of the edge W_{ij} where j is a member of set J. This activation function can be a number of different functions, including:

1. ReLU (Rectified Linear Unit)



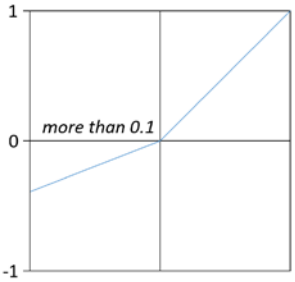
$y = \max(0, x)$

2. Leaky ReLU



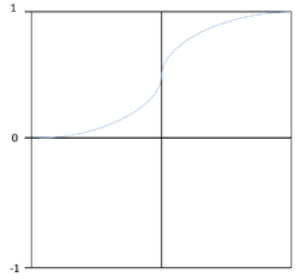
$y = \max(\sim 0.1x, x)$

3. Random ReLU



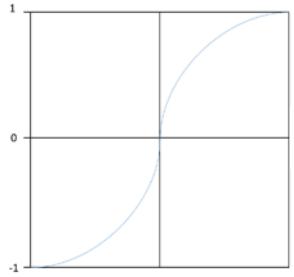
$y = \max(\sim 0.1x, x)$

4. Sigmoid



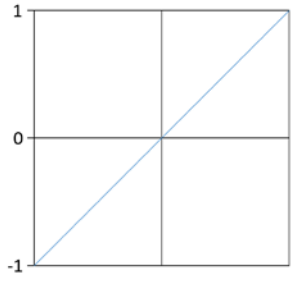
$y = \dfrac{1}{1 + e^{-x}}$

5. Hyperbolic Tangent



$y = \tanh x$

6. Linear



$y = x$

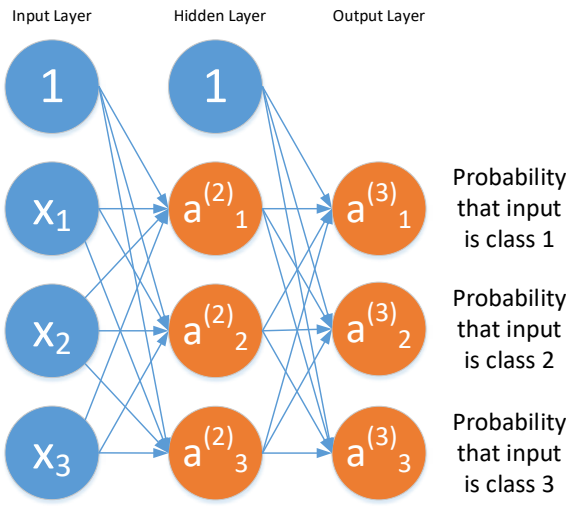


Figure 1.0 Example of a multiclass neural network

A image prediction NN is made up of layers of nodes, which are similar to neurons in a human brain, which the input (an image converted into an array) is routed through, and as an output the probabilities of the image being a certain class is received.

Convolutional Neural Networks (CNNs) are a form of neural networks specifically adapted for image recognition purposes, utilising specific features such as convolutions and pooling.

The key layers of the CNN are the input layer, convolution layer, ReLU layer, pooling layer, and the output (dense, softmax activated) layer. The input layer receives the raw pixel values of the image with 4, 3 or 1 channels depending on colour mode (RGBA, RGB, grayscale).

The convolution layer works by sliding a filter across the width and height of the input volume and computes the dot product between the filter and the kernel at any position. This convolves the original image into a feature map, which is a mapping of where specific features of the image are found. The convolutional layer can also act as feature identifiers, such as curve detector filters etc.

The Rectified Linear unit (ReLU) activation operation is used after every convolution operation to remove negative values from the neurons by running the function $x = \max(0, x)$, where x is the activation value. This helps to mitigate the vanishing gradient problem that can occur when training extremely deep models. The pooling layer is a method used to downsample the resolution and size of each feature map but still retain the most important information. Various methods of pooling exist, such as Max pooling, Average pooling, Sum pooling.

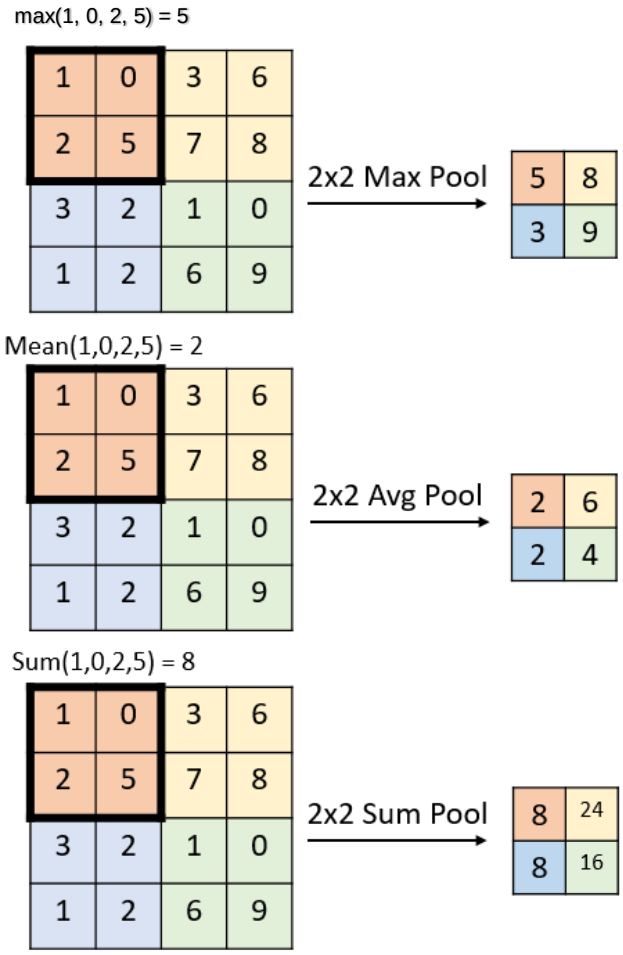


Figure 1.1: various pooling methods with 2x2 filters and a stride of 2

Pooling layers help to reduce the amount of parameters, thus reducing the computational complexity. As the resultant resolution is also decreased, this also helps to prevent overfitting of the model.

Apart from just being convolutional neural networks, the models we tested were also residual neural networks, with the exception of Inception v3.

Residual neural networks (ResNets) are inspired by pyramidal cells in the cerebral cortex. In residual neural networks, some activations can jump over layers and be added to the activation of a deeper layer.

ResNets are generally structured into various residual blocks, with a residual connection between the first and last layer of the block.

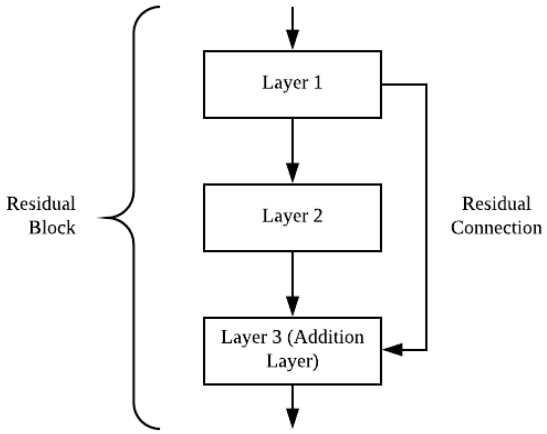


Figure 1.2 Residual blocks

Large ResNets can have multiple different types of residual blocks. For example, Google's Inception-ResNet v2 has 3 different residual blocks, named inceptions blocks a,b and c respectively.

Residual neural networks also to increase model accuracy by dealing with the vanishing gradient problem. When the model is first starting to learn, it can start by training a "shallower" version of itself by utilising the residual connections. This ensures a high enough gradient to perform large changes in its weights. As it learns more, it can make use of the layers within residual blocks to extract features more finely, enabling it to achieve a higher accuracy.

MATERIALS

The convolutional neural networks were developed in Ubuntu 18.04. Training and testing was performed on the following hardware.

1. CPU: Ryzen 1600 3.2GHz
2. RAM: 16GB DDR4 2666Hz
3. GPU: Nvidia GTX 1060 6GB

All training and prediction was done with GPU acceleration (Nvidia 390 driver, CUDA 9.0) in a Jupyter notebook.

The neural networks were developed on Python 3.6.7 mainly using the tensorflow-backend library, keras.

Additional Python Modules :

- Neural Networks: Tensorflow, Keras
- Calculations: NumPy, SciKit
- Image Processing: MatPlotLib, Open-CV, skimage
- File Processing: os, time
- Misc: Progress Bar: tqdm Confusion Matrix: sklearn

METHODOLOGY

Preparing the Dataset

For our testing, we decided to find staircase image data corresponding to 4 different classes

1. Right curving staircases
2. Left curving staircases
3. Straight staircases
4. Negative samples (images which did not contain staircases but possible scenarios an indoor bot could face)

To develop the dataset, we used 3 methods to obtain images. First is to use a web crawler to run through google images and extract images relevant to the respective classes. We then manually went through each image to verify its legitimacy. The second was to take videos of different angles of a real-life staircase, and periodically extract the frames at fixed intervals. Thirdly, for the curved stairs, we realised that a right curving stairs is a left curving stairs and vice versa, so we laterally inverted some of the curved staircase images to enlarge our dataset. These 3 techniques resulted in approximately 1000 images per class.

The raw images were then formatted to a dimension of 299 x 299 pixels to fit the largest expected input out of our models. We opted to load the images in greyscale as we believed that colour was not relevant for our classification task.

A side-script was run to split the dataset into a training set, validation set and testing set in a ratio of 0.81 : 0.09 : 0.1. The images were imported into the testing programme as a generator with the generator.

To further improve our training dataset, we utilised real-time data augmentation on the training data with the 'ImageDataGenerator' function from the 'keras.preprocessing.image' library. The parameters we used were:

1. *rescale= 1./255 #Except for ResNet 50*
2. *horizontal_flip=False*
3. featurewise_center=True
4. featurewise _std_normalisation=True
5. rotation_range=20
6. width_shift_range=0.2
7. height_shift_range=0.2
8. zoom_range=0.2

This image data generator was fit to a subsample of the training data (200 images x 4 classes) to allow it to calculate an estimate of the standard deviation and mean of the training data.

The rescale option ensured that the data was within the expected input range of the pretrained models. All models we tested expected input in the range of [0,1], with the exception of ResNet 50 which expected input in the range [0,255]. This rescale option is the only option used for the validation and test data generators.

It is important that horizontal_flip be set to false as a left curved staircase is a right curved staircase when laterally inverted.

Model Architecture

We carried out transfer learning of 4 different models, with pretrained weights for the 'imagenet' dataset by fitting them to our staircase datasets. These models acquired from "keras.applications".

1. Inception-Resnet v2
2. Inception v3
3. ResNet 50
4. MobileNet v2

Each model had its top dense layer removed and replaced with the following layers.

1. Global average pooling 2D
2. Dropout - 0.5 deactivation
3. Dense - 1024, ReLU activation
4. Dense - 4, softmax activation

The training was carried out in 2 phases.

First, the base model layers were frozen, and the new layers were trained for 5 epochs with the 'adam' optimizer at learning rate 0.001 (default).

Then, the top/last few layers of the base model were unfrozen and the model was trained again for 20 epochs or until the loss validation stopped decreasing (with a patience of 5 epochs). The optimizer used was 'adam' with a learning rate of 0.0001 as the model was already close to optimal.

The layers to be unfrozen were chosen by trying out several different values to find a closet optimal layer value. Note that due to limitations in GPU memory, we were unable to test out training the entire model for all models except for MobileNetv2.

Model	Section	Layer Number
Inception Resnet v2	Everything after A reduction block	275
Inception v3	Everything after the last A inception block	101
ResNet 50	Everything after conv3_x	80
MobileNet v2	Everything after block 7	73

Table 2.0: Sections of models unfrozen for second phase of training

We then compare the neural network models through the criterion of test accuracy, prediction runtimes, as well as displaying the results for each category in a confusion matrix.

Model Evaluation Methodology

Accuracy = Number of correct classifications/ Total number of Classifications

Runtime is calculated by taking the average the time it takes to import and make predictions for images in the test set (413 images) sequentially. The images were all resized to 299x299 px before importing. The python function `time.time()`is used to measure the time taken. The following operations are performed for each image to predict and thus included in the time.

```
img=image.load_img(os.path.join(data_dir,
img_name), target_size=(299,299),
color_mode="grayscale")
img=image.img_to_array(img)
img/=255.0
prediction=class_names[np.argmax(model.pr
edict([[img]])[0])]
```

Where image is keras.preprocessing.image

The confusion matrix is generated based on predicted and true values of images from the test dataset

DATA AND RESULTS

Table 3.0: Dataset Size Information

Dataset	Number of Images				
	curved_left	curved_right	negative	straight	total
Training	80	64	97	86	3269
Validation	832	832	873	732	327
Test	114	114	93	92	413

Table 3.1 Summary of Models and their Performance

Model Name	Inception-ResNet v2	Inception v3	ResNet 50	MobileNet v2
Number of Layers	785	316	180	160
Number of Parameters	55,914,724	23,905,060	25,689,988	3,573,828
Average Prediction Run-Time (s)	0.0613146642218 4905 (4th)	0.0295974400083 89746 (3rd)	0.0289990076429 9074 (2nd)	0.0251071464640 17634 (1st)
Train Accuracy (%)	98.12	94.53	87.00	79.09
Validation Accuracy (%)	93.12	89.69	84.38	80.0
Test Accuracy (%)	88.29 (1st)	86.59 (2nd)	82.44 (3rd)	77.56 (4th)

Table 3.2.1 Confusion Matrix for Inception-ResNet v2

True Class	Predicted Class			
	curved_left	**curved_right**	**negative**	**straight**
curved_left	92	18	1	3
curved_right	16	92	0	6
negative	0	0	89	1
straight	1	1	1	89

Table 3.2.2 Confusion Matrix for Inception v3

True Class	Predicted Class			
	curved_left	**curved_right**	**negative**	**straight**
curved_left	91	11	7	5
curved_right	15	89	5	5
negative	0	0	90	0
straight	1	1	5	85

Table 3.2.3 Confusion Matrix for ResNet 50

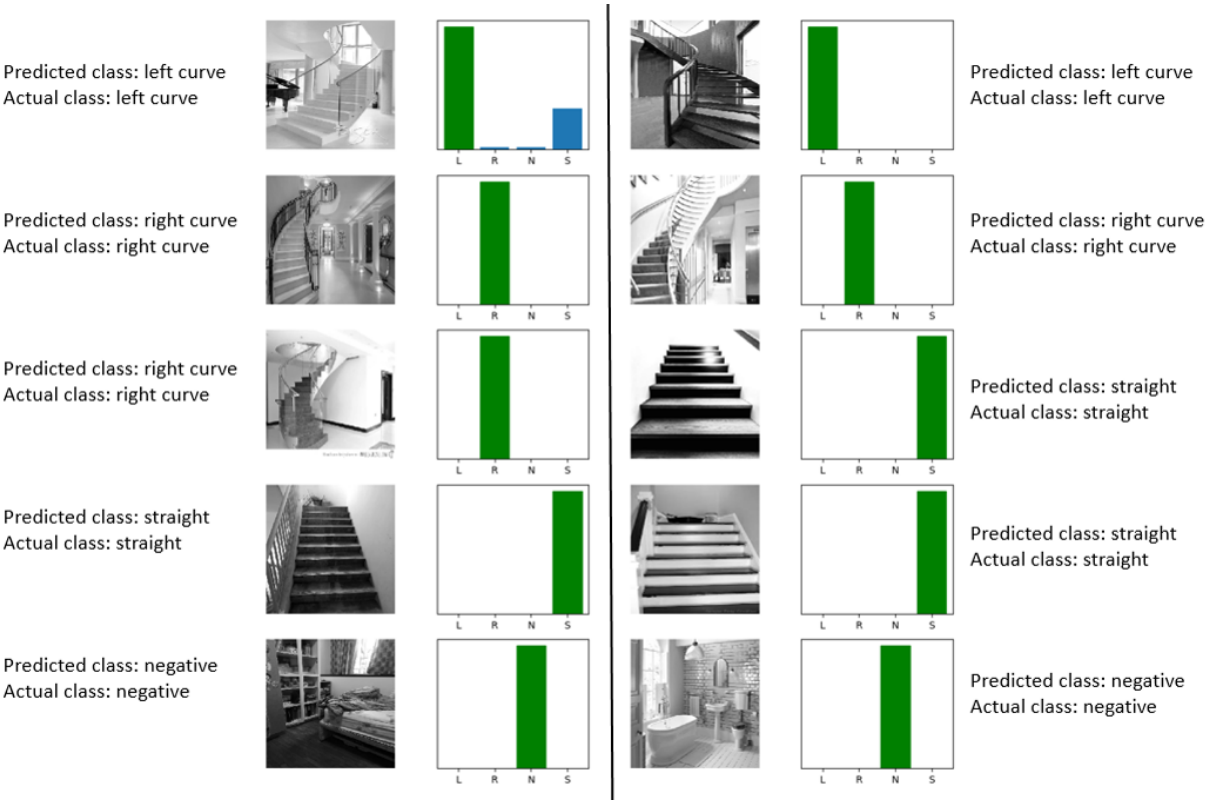
True Class	Predicted Class			
	curved_left	**curved_right**	**negative**	**straight**
curved_left	66	43	1	4
curved_right	7	99	1	7
negative	0	0	89	1
straight	2	2	4	84

Table 3.2.4 Confusion Matrix for MobileNet v2

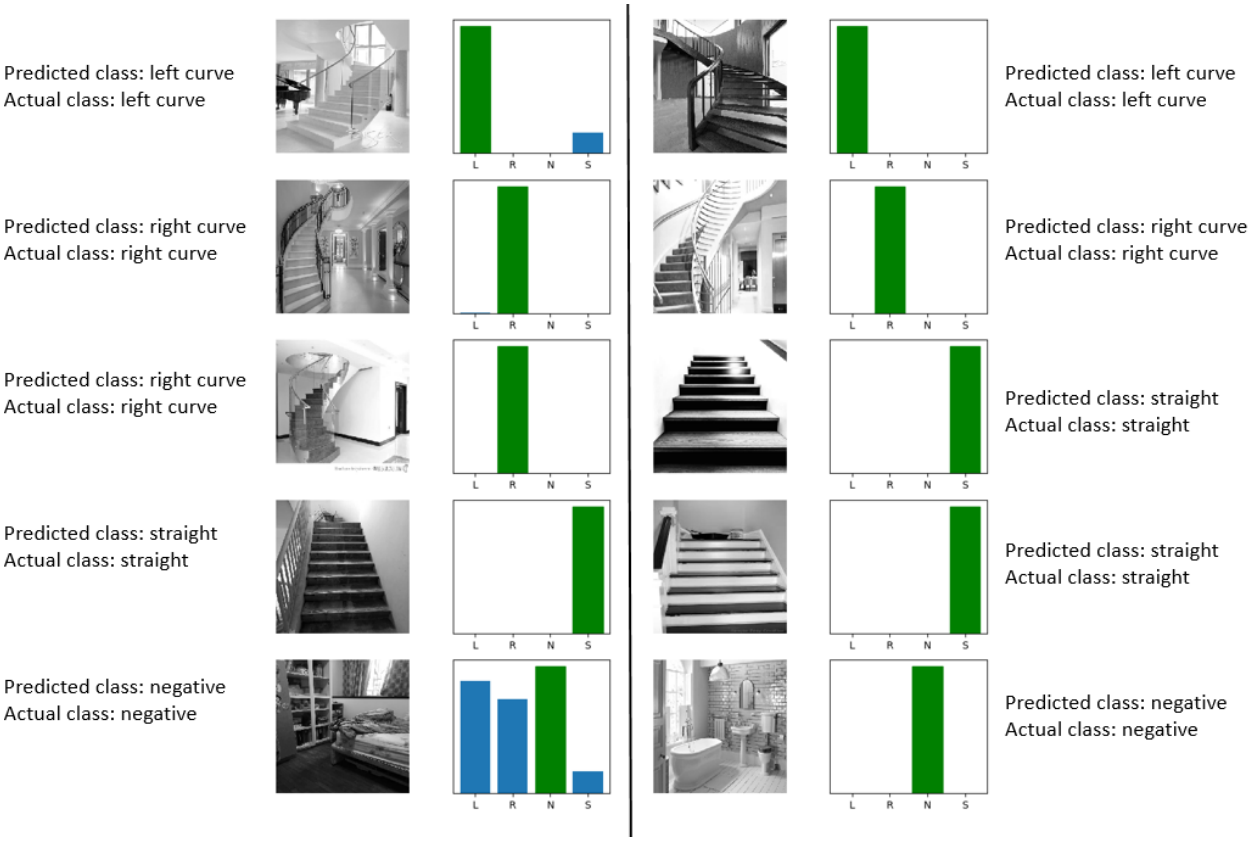
True Class	Predicted Class			
	curved_left	**curved_right**	**negative**	**straight**
curved_left	69	26	5	14
curved_right	23	76	3	12
Negative	0	0	88	2
straight	2	2	3	85

VISUALIZATIONS OF PREDICTIONS

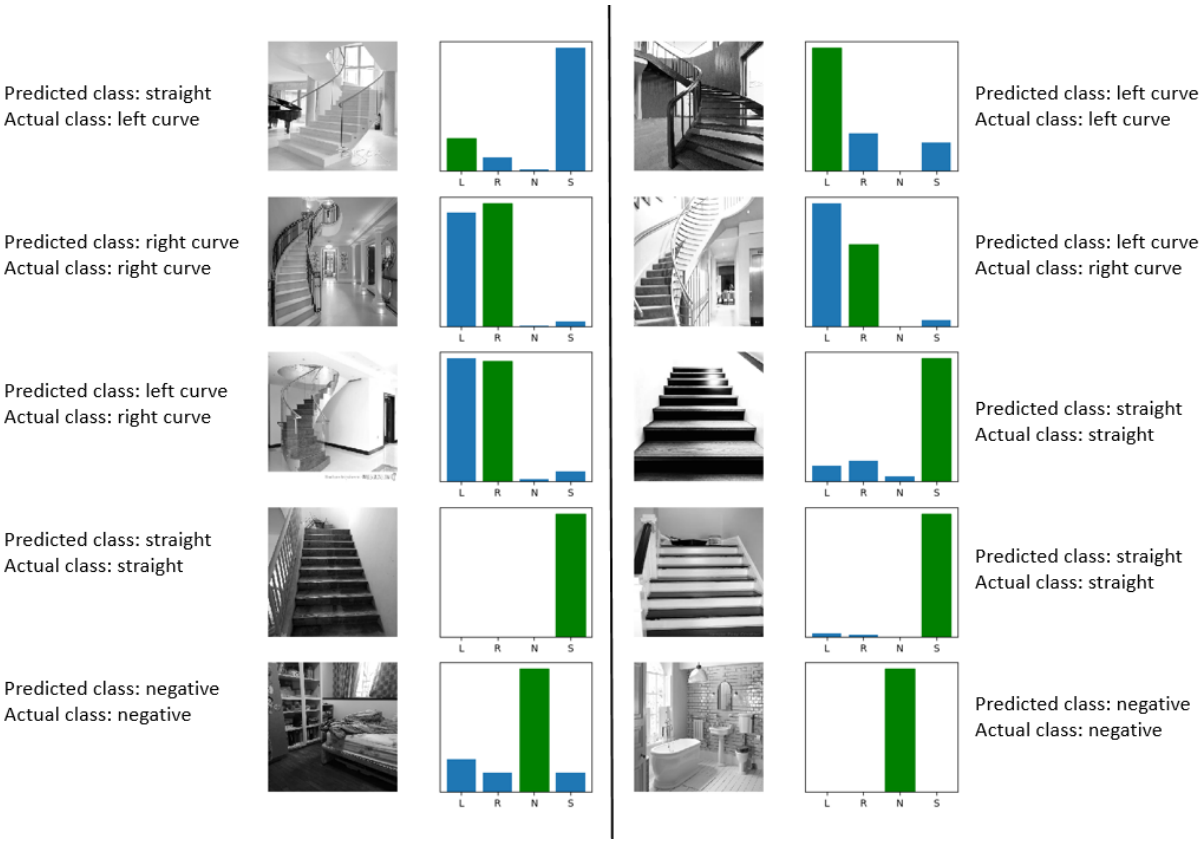
Predictions using Inception 3



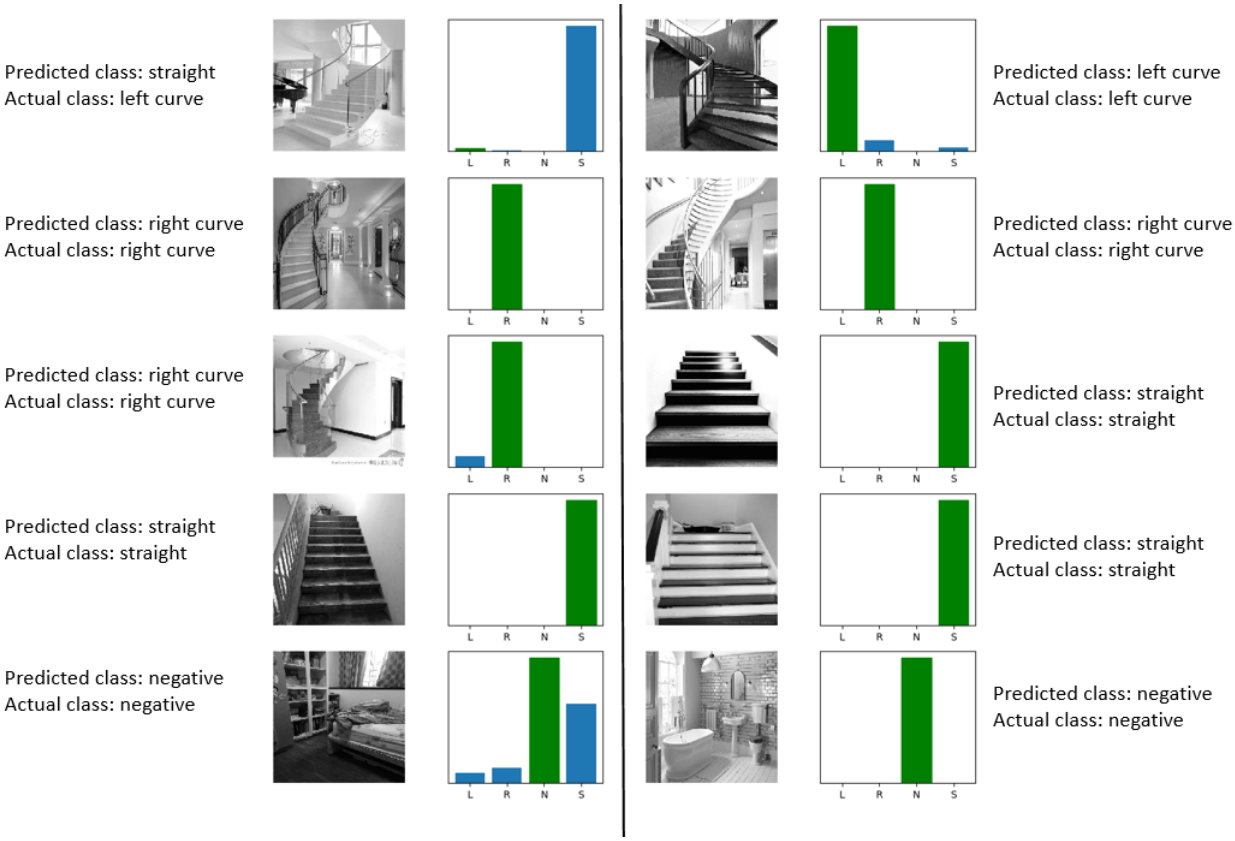
Predictions using Inception-Resnet v2



Predictions using mobilenet v2



Predictions using resnet 50



DISCUSSION OF RESULTS

From the test accuracy and average run time results, the trade-off between model accuracy and model speed is clear. While one would want a model with a high accuracy, in real world use in robotics, a model with a very low number of predictions per second could potentially cause the robot to be slower in processing and thus slower in moving around and performing its cleaning function.

Based on the results above, we believe that the most suitable model to use for the application of staircase cleaning robots would be the Inception v3 model. Its average run-time is only very slighly higher than ResNet 50's average run-time but is less than half of Inception-ResNet v2's run-time. Furthermore, its accuracy is a very acceptable 86.6%, only falling 2% short of Inception-ResNet v2's accuracy while being 4% higher than ResNet 50's accuracy.

CONCLUSION

In conclusion, we have performed transfer learning on 4 pre-trained keras models to enable them to detect and classify staircases in images. Ranked by accuracy, the models are: Inception-Resnet v2, Inception v3, ResNet 50, MobileNet v2. Their ranking for speed is the inverse of their ranking for accuracy. We recommend inception v3 for real world use as it has a relatively high accuracy and low average prediction time.

Acknowledgements

We would like to thank our mentors Dr Ilyas, Dr Mohan and all other researchers at SkunkWorks Laboratory in SUTD for their guidance throughout the project. We would also like to thank our school Temasek Junior College for giving us the opportunity to participate in this project.