Shopping Assistant AI

J COMPONENT PROJECT REPORT

Submitted by

Priyanshi Singh (19BIT0111)
Kunj Patel (19BIT0256)
Sejal Bharti (19BIT0286)

in fulfillment for the award of the degree of

B. Tech

Under the guidance of **Dr. Durai Raj Vincent**

Computer Science and Engineering

Vellore-632014, Tamil Nadu, India

School of Information Technology

Contents

1. Title: Shopping assistant AI

2. Abstract

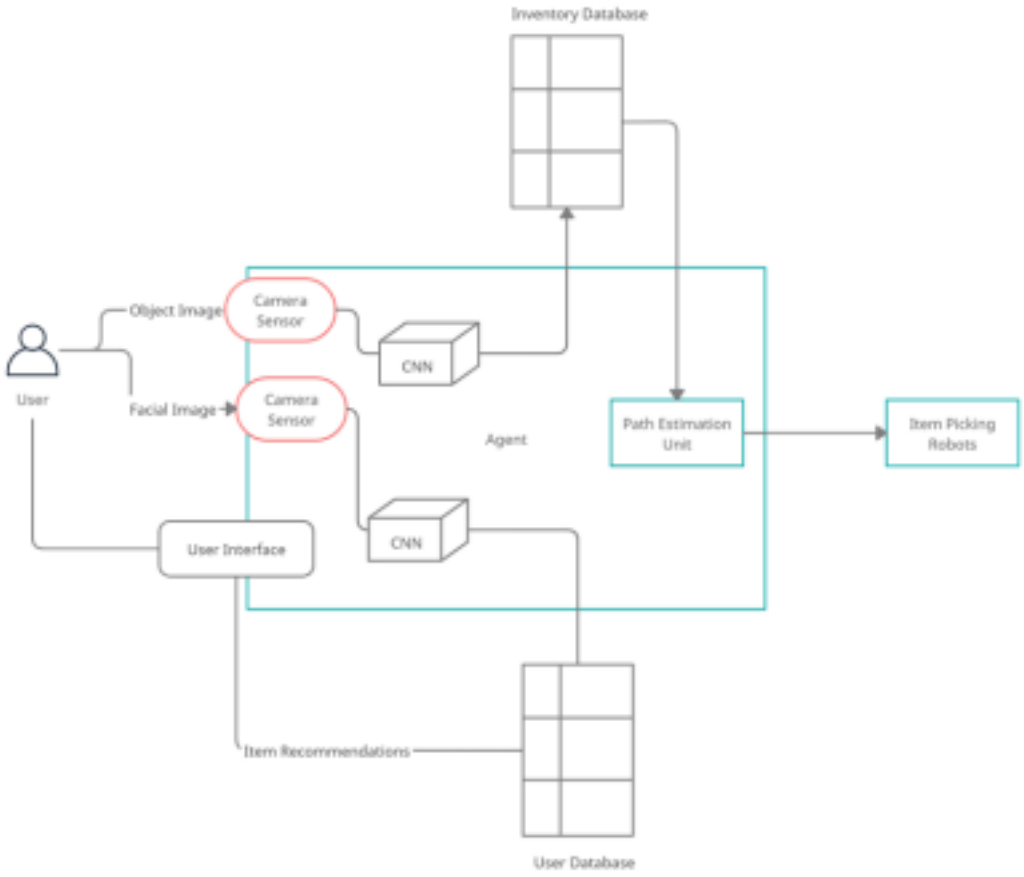
Our Project Shopping assistant AI is focused on application of artificial intelligence in a supermarket or a grocery store. We plan to understand the problems being faced by the current retailers and existing solutions that are being proposed by the authors of the various research papers and articles; we also plan to develop a shop automation system of our own with various modules that aim to improve the user experience in the grocery stores and supermarkets.

According to a survey excessive time consumption is one of the biggest problems a customer faces in a shop.Our automation system consists of explanation and implementation of three modules that aim to eliminate this problem. Let's consider an example- A customer X goes to a regular grocery store to buy some ingredients, In order to complete the task X should know all the ingredients required to make the dish ; X should then wander the store searching for each and every ingredient and X won't be sure if he would find all the ingredients. Our automation system solves all these problems ;When X scans the image of a dish using his camera or typing the name of the dish , the system shows him the details of all the ingredients and the shortest path that he needs to travel in order to collect all the desired ingredients.

3. Introduction

In the past few years we have observed a shift from offline to online in every sphere of life, whether it be shopping, banking or even education as we are facing right now in the COVID-19 scenario. This shift has added more convenience to most of our lives but has harmed some. The Retailers and supermarket owners have had loss of revenue because online shopping is much more convenient and hassle free. In the project we aim to identify what makes users prefer online shopping, how can we make supermarkets more convenient and hassle-free and how to implement Artificial Intelligence for better functioning of these supermarkets.

4. Architecture diagram



5. Background study (Related papers and study)

In one of the papers[1] a design framework of a shopping assistant system is presented which is to be used in supermarkets mainly by elderly or disabled people. We think that this design framework can also be implemented for the general public and not only the elder because of its ease of use thus we have implemented some of it in our modules.

In the paper[4] the author talks about an Intelligent shopping assistant system which

some-what matches with my proposed topic. It lays down in detail the technical aspects of the model i.e what hardware is to be used, system overview, product detection, Path Planning and RFID System, Q-Learning Obstacle Avoidance etc.

In the paper[5] the author talks about why it is important for the retail industry to accept and implement AI. The Retail Industry has been collecting customer data for years without putting it to any kind of use. This is where AI jumps in and enables Retailers to learn to be data-savvy. Having an idea about the customer needs beforehand leads to better management of the stock of products.

AmazonGo are the stores owned by the Tech giants Amazon Co. These are currently operating in the US. These stores were built on the basic idea of being easily accessible and smart by implementation of techniques like machine learning and AI. With this paper we take a deeper look into how these stores operate and what inspiration can we draw from them to our Shop Assistant AI.

Using all the information gained through the survey of the research papers; we can come up with a shop assistant AI which makes the usage of Retail stores / supermarkets much more convenient and hassle free . We've discussed the major competition being faced by online shopping and ways to overcome it, reduction of errors inside the retail store using AI, Removal of the waiting queue for checkout , prevention of time wastage using smart assistants, time saving by providing the shortest path and many more applications.

6. Methodology

This project involves a pipeline of several modules, out of which we have implemented two of them namely Object Recognition System and Shortest Optimal Path Finding module. The flow of data is as follows: The user gives the Object Recognition System an image of a grocery/food item and the system classifies and detects the object present in the image using a series of neural networks. It then fetches the information related to the item from the shop's database such as groceries involved (if the image was of a food item like pizza, it'll get items like flour, yeast, tomatoes, cheese etc), grocery item IDs, inventory of the item (for availability) and other specifics which depend on the organization implementing the system. Using this information fetched, it creates a list of available grocery items in the shop and passes it to the Optimal Path Finding module. The job of the Optimal Path Finding module is to find the shortest path and order to pick up the items in the shop depending on the layout of the shop encoded in the knowledge base. This shortest path can then be sent to item picking robots in the shop (in case of a completely automated shop) or be displayed to the user which they can follow to manually pick

up the items.

6.1 Object Recognition System

The Object Recognition System consists of a combination of CNNs[11] (convolutional neural networks) which classify the image independently and the most common class is selected as the final class.

6.1.1 BaseLine CNN

The first CNN is a baseline Neural Network which was constructed, trained and tested by us. The dataset used for the training of the neural network was engineered manually by us which was created by web scraping and filtering. The dataset has 4 classes: “cheeseburger”, “hotdog”, “ice_cream” and “pizza”. The reason for choosing these classes is that we have also involved the use of some of the state of the art neural networks like Xception, InceptionV3 and ResNet50 which were trained on ImageNet dataset. So we had a constraint of using only those classes present in the ImageNet dataset because we have used a pretrained model. The training set contains a total of 851 images and the test set contains a total of 58 images. We kept this size of dataset since the training of a deep neural network on a dataset of this size itself took around 2 hours and a bigger dataset would have more engineering issues and would not train a very good classifier if it’s skewed. The present dataset which was used was also a bit skewed in terms of distribution of image with different orientations in the test dataset, i.e the test dataset does not reflect the actual distribution of the training dataset.

In terms of preprocessing of the images, a standard preprocessing of scaling was done.

The model architecture is as follows:

Block 1: Conv2D layer with 32 convolutions and relu activation, BatchNormalization and MaxPooling (2,2)

Block 2: Conv2D layer with 64 convolutions and relu activation, BatchNormalization

Block 3: Conv2D layer with 64 convolutions and relu activation, BatchNormalization

Block 4: Conv2D layer with 64 convolutions and relu activation,

BatchNormalization

Block 5: Conv2D layer with 64 convolutions and relu activation,
BatchNormalization and MaxPooling (2,2)

Block 6: Conv2D layer with 128 convolutions and relu activation,
BatchNormalization and MaxPooling (2,2)

Block 7: Flatten layer

Block 8: Densely connected layer of 256 neurons and relu activation

Block 9: Densely connected layer of 128 neurons and relu activation

Block 10: Densely connected layer of 64 neurons and relu activation

Block 11: Output layer of 4 neurons for 4 classes with softmax activation

The optimizer used for training was Adam with a learning rate of 0.01 and the loss function used was categorical cross entropy. The training was done for 40 epochs with Early Stopping and ModelCheckpoint callback functions. The best epoch model with the validation accuracy of 89.65% was stored.

6.1.2 Pretrained State of the art CNNs

The other 3 models used were Xception[12], InceptionV3[13] and ResNet50[14] with pretrained weights on the ImageNet dataset. These models are known to have exceptional accuracy over the dataset and involving their use made our system more reliable in terms of accuracy. The decision algorithm was straightforward which involved declaring the class predicted by at least 50% of the models as the final class.

6.2 Shortest Optimal Path Finding module

After identifying the ingredients and their respective product ID this module creates a 2D array in which each element of the array represents the path cost of going from product1 (depicted by row number) to product2 (depicted by column number) . After finding the cost of the paths we apply the following algorithms in order to find the

ideal path i.e the one which takes the least amount of time. The module flow is as follows: First the cost of each pair of items in the item list is found out. Using this information a fully connected graph is formed. On this graph prim's algorithm is applied to get the order forming the shortest path. Now the path between each subsequent item pair is found out.

6.2.1 Floyd Warshall Algorithm

The Floyd–Warshall algorithm[15] is an algorithm for finding shortest paths in a directed weighted graph with positive or negative edge weights.

We initialize the solution matrix as the input graph matrix as a first step. Then we update the solution matrix by considering all vertices as an intermediate vertex. The idea is to one by one pick all vertices and update all shortest paths which include the picked vertex as an intermediate vertex in the shortest path. When we pick vertex number k as an intermediate vertex, we already have considered vertices {0, 1, 2, .. k-1} as intermediate vertices.

6.2.2 Prim's Algorithm

The idea behind Prim's algorithm is simple, a spanning tree means all vertices must be connected. So the two disjoint subsets (discussed above) of vertices must be connected to make a Spanning Tree. And they must be connected with the minimum weight edge to make it a Minimum Spanning Tree.

7. Results and Discussion

BaseLine CNN architecture:

```
In [8]: classifier = Sequential()

classifier.add(Conv2D(32,(3,3), activation='relu', input_shape=(180, 180, 3), padding="same"))
classifier.add(BatchNormalization())
classifier.add(MaxPooling2D(pool_size=(2, 2)))

classifier.add(Conv2D(64,(3,3), activation='relu', padding="same"))
classifier.add(BatchNormalization())

classifier.add(Conv2D(64,(3,3), activation='relu', padding="same"))
classifier.add(BatchNormalization())

classifier.add(Conv2D(64,(3,3), activation='relu', padding="same"))
classifier.add(BatchNormalization())

classifier.add(Conv2D(64,(3,3), activation='relu', padding="same"))
classifier.add(BatchNormalization())
classifier.add(MaxPooling2D(pool_size=(2, 2)))

classifier.add(Conv2D(128,(3,3), activation='relu', padding="same"))
classifier.add(BatchNormalization())
classifier.add(MaxPooling2D(pool_size=(2, 2)))

classifier.add(Flatten())

classifier.add(Dense(256, activation='relu'))
classifier.add(BatchNormalization())

classifier.add(Dense(128, activation='relu'))
classifier.add(BatchNormalization())

classifier.add(Dense(64, activation='relu'))
classifier.add(BatchNormalization())

classifier.add(Dense(4, activation = 'softmax'))

opt = Adam(learning_rate=0.001)
classifier.compile(optimizer = opt, loss = 'categorical_crossentropy', metrics = ['accuracy'])

print(classifier.summary())
```

```
dense_4 (Dense)                 (No
=================================
Total params: 16,108,868
Trainable params: 16,107,140
Non-trainable params: 1,728
_________________________________
None
```

Training:

Training Model

```
In [31]: es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=20)
mc = ModelCheckpoint(model_dir, monitor='val_accuracy', mode='max', verbose=1, save_best_only=True)
history=classifier.fit_generator(
        training_set,
        steps_per_epoch=851/5,
        epochs=40,
        validation_data=test_set,
        validation_steps=50/5,
        callbacks=[es, mc]
)
```

```
171/170 [==============================] - 236s 1s/step - loss: 0.8587 - accuracy: 0.6555 - val_loss: 0.5672 - val_accuracy:
0.8793

Epoch 00037: val_accuracy did not improve from 0.89655
Epoch 38/40
171/170 [==============================] - 245s 1s/step - loss: 0.8505 - accuracy: 0.6510 - val_loss: 0.5225 - val_accuracy:
0.7931

Epoch 00038: val_accuracy did not improve from 0.89655
Epoch 39/40

171/170 [==============================] - 233s 1s/step - loss: 0.8698 - accuracy: 0.6510 - val_loss: 2.2827 - val_accuracy:
0.3621

Epoch 00039: val_accuracy did not improve from 0.89655
Epoch 40/40
171/170 [==============================] - 196s 1s/step - loss: 0.8560 - accuracy: 0.6710 - val_loss: 3885.2014 - val_accurac
y: 0.6552

Epoch 00040: val_accuracy did not improve from 0.89655
```

Model predictions:

Test Image

```
In [9]: test_img_dir = os.path.join(test_dir, 'burger', '3.jpg')
```

BaseLine Prediction

```
In [10]: classifier = load_model(model_dir)
test_img = image.load_img(test_img_dir, target_size = (180, 180))
test_img = image.img_to_array(test_img)
test_img = np.expand_dims(test_img, axis=0)

result = classifier.predict(test_img)
BaseLinePred = getLabel(result)
print('Predicted:', BaseLinePred)
```

```
Predicted: cheeseburger
```

Resnet50 Prediction

In [11]:
```
from keras.applications.resnet50 import preprocess_input, decode_predictions

resnet50 = ResNet50(weights='imagenet')
img = image.load_img(test_img_dir, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

ResNetPred = resnet50.predict(x)
ResNetPred = decode_predictions(ResNetPred, top=1)[0][0][1]
print('Predicted:', ResNetPred)
```

```
Predicted: cheeseburger
```

InceptionV3 Prediction

In [12]:
```
from keras.applications.inception_v3 import preprocess_input, decode_predictions

inceptionV3 = InceptionV3(weights='imagenet')
img = image.load_img(test_img_dir, target_size=(299, 299))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

InceptionPred = inceptionV3.predict(x)
InceptionPred = decode_predictions(InceptionPred, top=1)[0][0][1]
print('Predicted:', InceptionPred)
```

```
Predicted: cheeseburger
```

Xception Prediction

In [13]:
```
from keras.applications.xception import preprocess_input, decode_predictions

xception = InceptionV3(weights='imagenet')
img = image.load_img(test_img_dir, target_size=(299, 299))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

XceptionPred = xception.predict(x)
XceptionPred = decode_predictions(XceptionPred, top=1)[0][0][1]
print('Predicted:', XceptionPred)
```

```
Predicted: cheeseburger
```

Floyd Warshall algorithm:

In [17]:
```
# Standard Floyd Warshall Algorithm
# with little modification Now if we find
# that dis[i][j] > dis[i][k] + dis[k][j]
# then we modify next[i][j] = next[i][k]
def floydWarshall(V):
    global dist, Next
    for k in range(V):
        for i in range(V):
            for j in range(V):
                # We cannot travel through
                # edge that doesn't exist
                if (dis[i][k] == INF or dis[k][j] == INF):
                    continue
                if (dis[i][j] > dis[i][k] + dis[k][j]):
                    dis[i][j] = dis[i][k] + dis[k][j]
                    Next[i][j] = Next[i][k]
```

Prim's algorithm:

In [20]:

```
def primMST(cost):
    inMST = [False] * V1

    # Include first vertex in MST
    inMST[0] = True

    # Keep adding edges while number of included
    # edges does not become V-1.
    edge_count = 0
    mincost = 0
    while edge_count < V1 - 1:

        # Find minimum weight valid edge.
        minn = INT_MAX
        a = -1
        b = -1
        for i in range(V1):
            for j in range(V1):
                if cost[i][j] < minn:
                    if isValidEdge(i, j, inMST):
                        minn = cost[i][j]
                        a = i
                        b = j

        if a != -1 and b != -1:
            print("Edge %d: (%d, %d) cost: %d" %
                (edge_count, required[a], required[b], minn))
            printPath(constructPath(required[a],required[b]))
            edge_count += 1
            mincost += minn
            inMST[b] = inMST[a] = True
    print("Minimum cost = %d" % mincost)
```

Class selection:

Getting best prediction, groceries and their ids

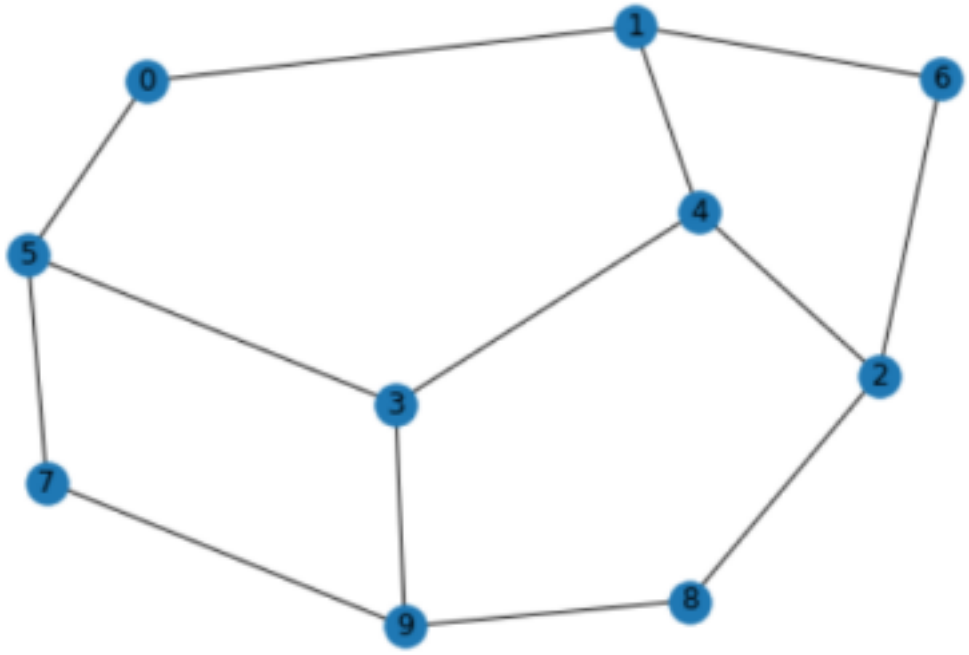
```
In [14]: results = [BaseLinePred, ResNetPred, InceptionPred, XceptionPred]
         finalPrediction = getFinalLabel(results)
         groceries = getGroceries(finalPrediction)
         groceriesLabels = groceries[0]
         groceriesIds = groceries[1]
         print('Food Item Detected: ', finalPrediction)
         print('Groceries required: ')
         for i in range(4):
             print(i+1,' ',groceriesLabels[i])
         print('Groceries ids: ', groceriesIds)
```

```
Food Item Detected:  cheeseburger
Groceries required:
1   bread
2   tomato
3   potato
4   cheese
Groceries ids:  [0, 5, 6, 1]
```

Shortest Path:

```
graph = [
    [ 0, 2, INF, INF, INF, 9, INF, INF, INF, INF],
    [ 2, 0, INF, INF, 6, INF, 4, INF, INF, INF],
    [ INF, INF, 0, INF, 4, INF, 3, INF, 5, INF],
    [ INF, INF, INF, 0, 2, 1, INF, INF, INF, 3],
    [ INF, 6, 4, 2, 0, INF, INF, INF, INF, INF],
    [ 9, INF, INF, 1, INF, 0, INF, 6, INF, INF],
    [ INF, 4, 3, INF, INF, INF, 0, INF, INF, INF],
    [ INF, INF, INF, INF, INF, 6, INF, 0, INF, 3],
    [ INF, INF, 5, INF, INF, INF, INF, INF, 0, 7],
    [ INF, INF, INF, 3, INF, INF, INF, 3, 7, 0]
]

# Function to initialize the
```




```
[[0, 9, 6, 2], [9, 0, 10, 9], [6, 10, 0, 4], [2, 9, 4, 0]]
Edge 0: (0, 1) cost: 2
0 -> 1
Edge 1: (6, 1) cost: 4
6 -> 1
Edge 2: (0, 5) cost: 9
0 -> 5
Minimum cost = 15
```

8. Conclusion

AI is undoubtedly revolutionizing shopping experiences. As an effort to understand the impact of AI on the field, the results of this project will help us explore how AI helps bridge the drawbacks of virtual and physical shopping that exist. This project deals with the problem of automated recognition of photographed food items and the subsequent output of the appropriate ingredients. We perform object recognition using Convolutional Neural Networks (CNN). We then use path-finding algorithms such as Floyd Warshall algorithm, Prim's algorithm for Minimum spanning tree for finding the optimal path the customer (or part picking robot) can take to avoid aimlessly wandering the isles for the required products. This research completely revolutionizes our everyday grocery shopping. The research will also enable us to

also understand the impact of AI on shopping personnel as their jobs are displaced and replaced by AI. For future scope, we intend to add other features as well like face-recognition for logging into the customer's account where the customer will have his history of purchases. We also intend to add features like recommendation system, supply forecasting and automatic restocking.

References

[1] Marin-Hernandez, A., de Jesús Hoyos-Rivera, G., Garcia-Arroyo, M. and Marin-Urias, L.F., 2012, October. Conception and implementation of a supermarket shopping assistant system. In *2012 11th Mexican International Conference on Artificial Intelligence* (pp. 26-31). IEEE.

[2] Shyna, K. and Vishal, M., 2017. A Study On Artificial Intelligence E-Commerce. *International Journal of Advances in Engineering & Scientific Research*, *4*(4), pp.62-68.

[3] Zhang, W. and Hu, D., 2009, November. Research on customer-oriented shopping guide system of suit-dress with intelligent characteristics. In *2009 IEEE 10th International Conference on Computer-Aided Industrial Design & Conceptual Design* (pp. 1979-1982). IEEE.

[4] Wu, B.F., Yao, S.J., Hou, L.W., Chang, P.J., Tseng, W.J., Huang, C.W., Chen, Y.S. and Yang, P.Y., 2016, November. Intelligent shopping assistant system. In *2016 International Automatic Control Conference (CACS)* (pp. 236-241). IEEE.

[5] Oosthuizen, K., Botha, E., Robertson, J. and Montecchi, M., 2020. Artificial intelligence in retail: The AI-enabled value chain. *Australasian Marketing Journal*, pp.j-ausmj.

[6] De Bellis, E. and Johar, G.V., 2020. Autonomous shopping systems: Identifying and overcoming barriers to consumer adoption. *Journal of Retailing*, *96*(1), pp.74-87.

[7] Polacco, A. and Backes, K., 2018. The amazon go concept: Implications, applications, and sustainability. *Journal of Business and Management*, *24*(1), pp.79-92.

[8] Chiu, Y.P., Lo, S.K., Hsieh, A.Y. and Hwang, Y., 2019. Exploring why people spend more time shopping online than in offline stores. *Computers in Human Behavior*, *95*, pp.24-30.

[9]Dalta, I.M., Nur, F.P.D., Amaral, S.T. and Adiono, T., 2019, October. Artificial intelligence based in-store traffic monitoring system for evaluating retail

performance.
In *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)* (pp. 430-431). IEEE.

[10] Buza, K., Buza, A. and Kis, P.B., 2010, May. Towards better modeling of supermarkets. In *2010 International Joint Conference on Computational Cybernetics and Technical Informatics* (pp. 499-503). IEEE.

[11] Albawi, S., Mohammed, T.A. and Al-Zawi, S., 2017, August. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)* (pp. 1-6). Ieee.

[12] Chollet, F., 2017. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1251-1258).

[13] Xia, X., Xu, C. and Nan, B., 2017, June. Inception-v3 for flower classification. In *2017 2nd International Conference on Image, Vision and Computing (ICIVC)* (pp. 783-787). IEEE.

[14] Ray, S., 2018. Disease classification within dermascopic images using features extracted by resnet50 and classification through deep forest. *arXiv preprint arXiv:1807.05711*.

[15] Papadimitriou, C. and Sideri, M., 1999. On the Floyd–Warshall algorithm for logic programs. *The journal of logic programming*, *41*(1), pp.129-137.