



INHA UNIVERSITY TASHKENT
DEPARTMENT OF CSE & ICE
ARTIFICIAL INTELLIGENCE

Group: Dream Team

Project title: “Virtual Designer”

Team members:

Sevara Abdullaeva	U1710059 - 001
Madina Akhmadjanova	U1710182 - 001
Bekhzod Aliev	U1710270 - 002
Javlonkhujja Eshonkhujjaev	U1710206 - 001
Yuliya Durova (Team Leader)	U1710104 - 001

Abstract

These days, due to lockdown we faced a lot of difficulties regarding offline interaction with people, things etc. So, looking at all this picture, we came up with the idea to solve this real-life problem. The real-life problem is shopping and designer job during lockdown. So, our decision which is based on Artificial Intelligence is to help people to become designers and buy furniture through online store.

The 4th industrial revolution which was boosted by pandemic, so this would help our system and society to get used for our application. The concept of our idea and project is based on O2O service. We decided to name our project as “Virtual Designer”, and we are going to solve such problems as time-consuming (instead of going to offline shops), interaction with people due to isolation needs and money expenditure on offline services.

As soon as we came up with this idea, we analyze it through convergence and consilience in order to be sure that we are the first in this field. We gathered together several times offline and online to decide whether this idea is a real consilience or not, and came up with the decision to do this project. So, we are going to discuss about our architecture more deeply in our report.

During the Artificial Intelligence course we learned how to identify datasets, prepare it; what is Machine Learning, what learning algorithms are better to use in classification and what are better for prediction. Also, now we are able to build a deep learning model with several neuron layers, using different activation functions. In this report we would like to show the result we achieved through the course.

Table of Contents

1.	Introduction.....	4
1.1	Objective	4
1.2	Motivation	4
1.3	Roles of team members.....	4
2.	Feasibility study	4
2.1	Value chain optimization for cost reduction	4
2.1.1	Reduction in the manual process.....	4
2.1.2	Reduction in human effort per process	5
2.1.3	Reduction in error rate from human decision making	5
2.2	Insight generation by Artificial Intelligence	5
2.2.1	Increase in number/impact of positive decisions.	5
2.2.2	Decrease in number/impact of negative decisions.	5
2.3	End-to-end picture.....	6
3.	Data for Machine Learning.....	7
3.1	Define dataset.....	7
3.2	Identifying free datasets available.....	8
3.3	Loading & reading the datasets with Python Jupyter.....	9
3.4	Data processing and discovery technologies for the collected and stored datasets.....	12
3.5	Wrangling of datasets with Python Pandas	13
3.6	Visualization of results.....	17
4.	Machine Learning	20
4.1	Machine Learning Life Cycle Architecture	20
4.2	Choosing features to use/extract from data	21
4.3	Classify machine learning types for the selected features.....	21
4.4	Getting started with TensorFlow Installation.....	21
4.5	Logistic regression	23
5.	Deep Learning.....	27
5.1	Deep Learning algorithms.....	27
5.2	Experimental end-to-end DL design and implementation of a simple scenario for our application with TensorFlow.....	28
5.3	Demo	33
6.	Summary	38
6.1	Conclusion.....	38
6.2	Future plans	38

1. Introduction

1.1 Objective

[X] – Online Furniture Shopping

[Y] – Analysis of Artificial arrangement of furniture in house

[X: Online Furniture Shopping] + IICT -> [Y: Analysis of Artificial arrangement of furniture in house]

1.2 Motivation

The difficulty of furniture arrangement was always one of the main obstacles in people's life. Moreover, at the height of lockdown, people are obliged to stay at their home without the opportunity to go out for a shopping and face some problems with parameters and sizes in online shopping. So, we decided to tackle this problem and provide the solution by developing online shopping and integrating AI to automate the process of matching the suitable appointment.

Our aim is to optimize the shopping process by developing room recognition in the house, in order to make rearrangements of the current furniture and creating new designs for empty rooms by recommending appointments based on technologies of 3D-modelling and Machine Learning. Our approach is a consilience supported by convergence in the field of online shopping, which will reduce the cost of expenditure and increase the efficiency of traditional ways of this industry.

1.3 Roles of team members

Yuliya Durova – Technical Leader, Researcher.

Sevara Abdullaeva – Researcher, Developer.

Bekhzod Aliev – Project Manager, Researcher.

Javlonkhujja Eshonkhujjaev – Researcher, Design developer.

Madina Akhmadjanova – Researcher, Designer.

2. Feasibility study

2.1 Value chain optimization for cost reduction

2.1.1 Reduction in the manual process

As far as our proposal is aimed at designing the view of a given space, it reduces the human involvement. It means that there is no longer a need to hire an expensive designer for ordinary and basic apartments. In addition to this, with the usage of augmented reality (as feasible part of our idea) for color matching, room recognition and furniture substitution no sketches and 3D models will be required, for everything will be analyzed and proposed immediately.

2.1.2 Reduction in human effort per process

Because the main task will be performed by Artificial Intelligence, human involvement is decreased to minimum. Person only should point his smartphone's camera to the required area of the room and the application layer will display options of furniture. No need for designers or visits to physical shops.

2.1.3 Reduction in error rate from human decision making

Sometimes when people think about designing there is a possibility of mistakes. For example, in choosing the color of the furniture or measuring the size of the place where it will be placed. In order to avoid this kind of mistake our machine will correctly measure size, choose colors and suggest more suitable choices for people. By this way we are reducing human error.

2.2 Insight generation by Artificial Intelligence

2.2.1 Increase in number/impact of positive decisions.

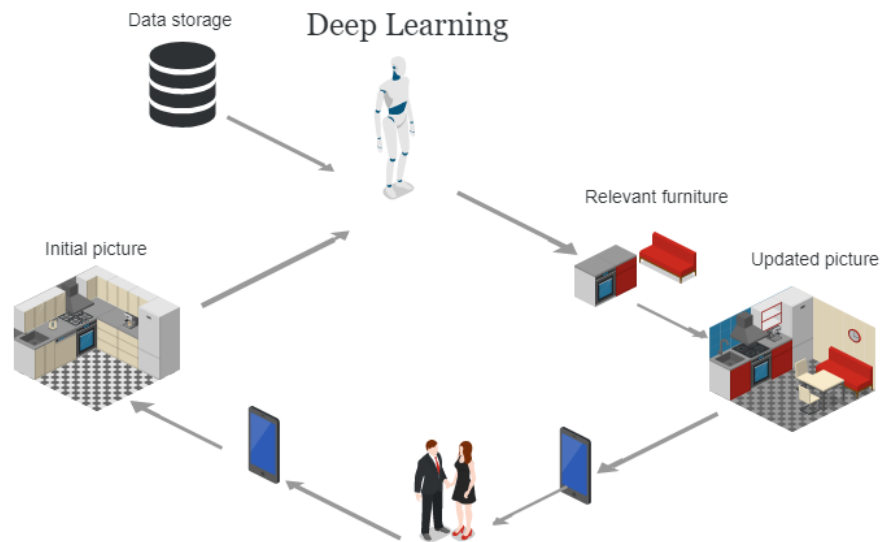
People are trying to find some ways to make their life easier, which means that by using our program they will gain some advantages. Our approach helps to elaborate decisions which were made by a machine. People first open the camera, then our system will grab the data and analyze it. So, using AI and MA algorithms, we can create useful insights for smart placement of the objects in the room. Moreover, developing this kind of O2O application with integrated AI, will have a great influence and provide convergence and consilience to the development of value creation.

2.2.2 Decrease in number/impact of negative decisions.

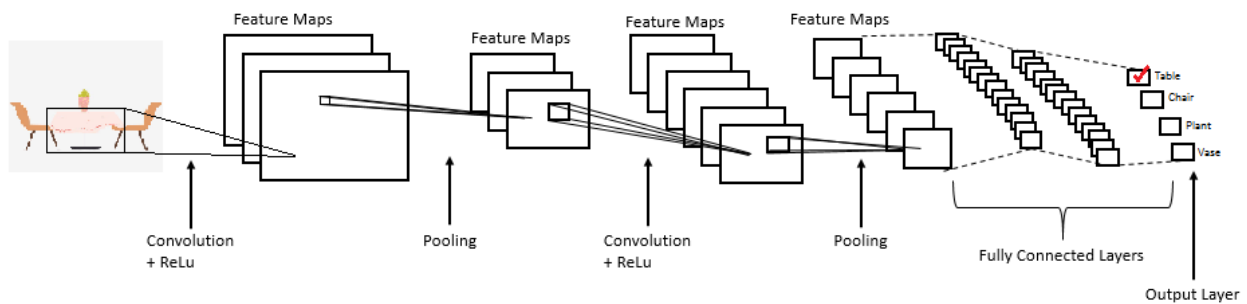
Nowadays, people think more about reducing the value of negative impact. So, the application of placing the furniture based on the AI, helps to decrease the negative decisions made by humans which are based on slow, random searching. Therefore, if our program will be implemented into the chain of furniture market and sells, it might reduce the human error, mistake involvement, information loss to name a few.

2.3 End-to-end picture

Our aim is to help humanity to solve the common problem, which is confused in choosing and placing the furniture. So, our app will solve this problem by gathering the data and analyzing it using database and internet. Afterwards, which will be occurred in a manner of showing the example of suggested models which people see on their smartphone.



Where for Deep Learning we use Convolutional Neural Network for object detection.



3. Data for Machine Learning

3.1 Define dataset

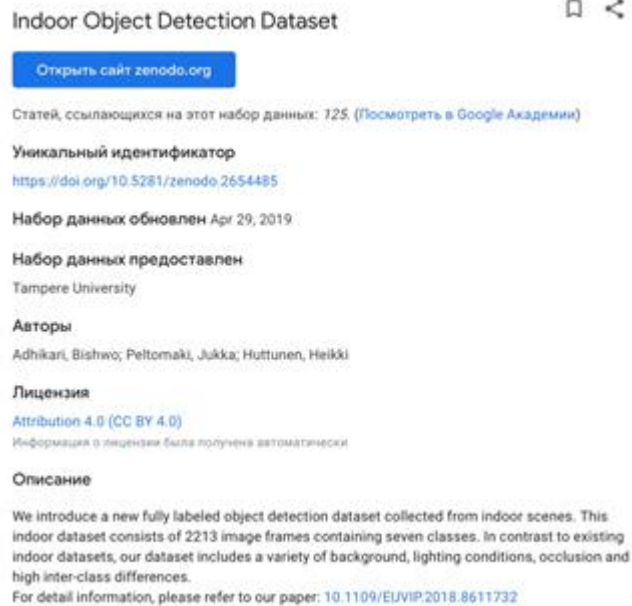
We introduce the Datasets, Items and Data Collections in order to provide some information about our project. The Datasets that we would like to find and use are the minimum requirements to start our project of Virtual Designer.

So, first we prepare all the needed data by selecting, processing and transforming it. Then, we construct a dataset. Finally, we convert the data to vectors for machine learning understanding.

Data Set	Item	Data Collection
Objects	Furniture	Name, Size, Color, Type, Image, Description
Area	Room	Size, Items, Image, Borders, type
Color	Color	Name, Shadow, Image

3.2 Identifying free datasets available

Our application is intended to act like a virtual designer shop, so we need to find a dataset that will have products – in our case furniture, and for usage of this furniture – rooms. We have run through various datasets in Kaggle and Dataset Search from google, and found that most of them lack essential information for us. For example, we found the Indoor Object Detection Dataset, which only detects existing indoor objects.



Indoor Object Detection Dataset

[Открыть сайт zenodo.org](#)

Статей, ссылающихся на этот набор данных: 125. (Посмотреть в Google Академии)

Уникальный идентификатор
<https://doi.org/10.5281/zenodo.2654485>

Набор данных обновлен Apr 29, 2019

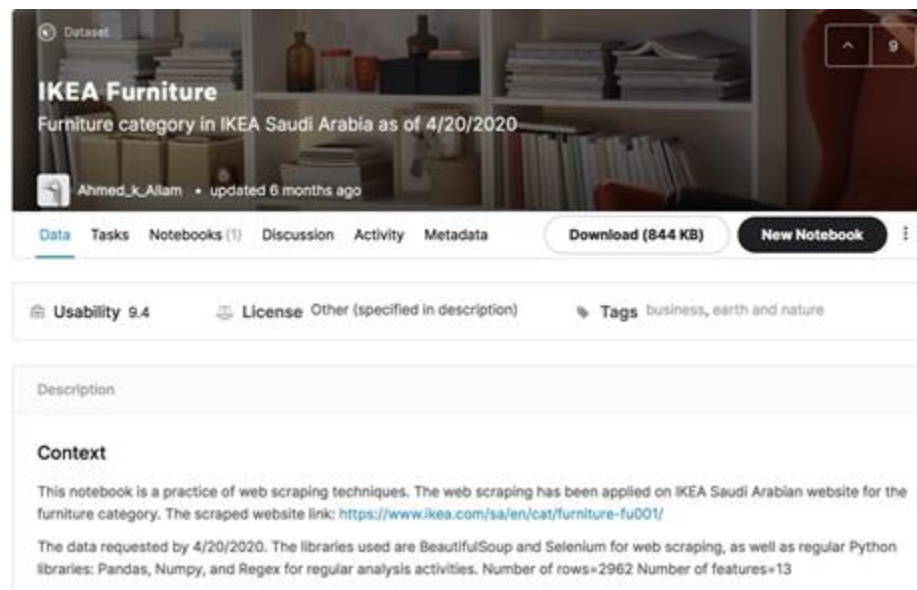
Набор данных предоставлен
Tampere University

Авторы
Adhikari, Bishwo; Peltomaki, Jukka; Huttunen, Heikki

Лицензия
[Attribution 4.0 \(CC BY 4.0\)](#)
Информация о лицензии была получена автоматически

Описание
We introduce a new fully labeled object detection dataset collected from indoor scenes. This indoor dataset consists of 2213 image frames containing seven classes. In contrast to existing indoor datasets, our dataset includes a variety of background, lighting conditions, occlusion and high inter-class differences.
For detail information, please refer to our paper: [10.1109/EUWIP.2018.8611732](https://arxiv.org/abs/10.1109/EUWIP.2018.8611732)

Also, for products, we searched IKEA Furniture datasets (1), while it possesses some valuable information and we do try to use it, still is not enough for us.



Dataset

IKEA Furniture
Furniture category in IKEA Saudi Arabia as of 4/20/2020

Ahmed_k_Alam • updated 6 months ago

[Data](#) [Tasks](#) [Notebooks \(1\)](#) [Discussion](#) [Activity](#) [Metadata](#) [Download \(844 KB\)](#) [New Notebook](#)

Usability 9.4 **License** Other (specified in description) **Tags** business, earth and nature

Description

Context
This notebook is a practice of web scraping techniques. The web scraping has been applied on IKEA Saudi Arabian website for the furniture category. The scraped website link: <https://www.ikea.com/sa/en/cat/furniture-fu001/>
The data requested by 4/20/2020. The libraries used are BeautifulSoup and Selenium for web scraping, as well as regular Python libraries: Pandas, Numpy, and Regex for regular analysis activities. Number of rows=2962 Number of features=13

3.3 Loading & reading the datasets with Python Jupyter

To read it we need to open Jupyter Lab and create an “.ipynb” file. Then, import Python Pandas. As our dataset has “. csv” extension, we need to use Pandas function read_csv(). Here is screenshot of reading Dataset:

```
[1]: import pandas as pd
mydataset=pd.read_csv('Desktop/4 year/ai datasets/IKEA_SA_Furniture_Web_Scrapings_sss.csv')
mydataset
```

```
[1]:
```

	Unnamed: 0	item_id	name	category	price	old_price	sellable_online	link	other_colors	short_description	designer	depth	height	width
0	0	90420332	FREKVENS	Bar furniture	265.0	No old price	True	https://www.ikea.com/sa/en/p/frekvens-bar-tabl...	No	Bar table, in/outdoor, 51x51 cm	Nicholai Wiig Hansen	NaN	99.0	51.0
1	1	368814	NORDVIKEN	Bar furniture	995.0	No old price	False	https://www.ikea.com/sa/en/p/nordviken-bar-tab...	No	Bar table, 140x80 cm	Francis Cayouette	NaN	105.0	80.0
2	2	9333523	NORDVIKEN / NORDVIKEN	Bar furniture	2095.0	No old price	False	https://www.ikea.com/sa/en/p/nordviken-nordvik...	No	Bar table and 4 bar stools	Francis Cayouette	NaN	NaN	NaN
3	3	80155205	STIG	Bar furniture	69.0	No old price	True	https://www.ikea.com/sa/en/p/stig-bar-stool-wk...	Yes	Bar stool with backrest, 74 cm	Henrik Preutz	90.0	100.0	60.0
4	4	30180504	NORBERG	Bar furniture	225.0	No old price	True	https://www.ikea.com/sa/en/p/norberg-wall-moun...	No	Wall-mounted drop-leaf table, ...	Marcus Aronson	60.0	43.0	74.0
...
3689	3689	99157902	ELVARLI	Wardrobes	750.0	SR 820	True	https://www.ikea.com/sa/en/p/elvarli-1-section...	No	1 section, 92x51x222-350 cm	Ehén Johansson	50.0	NaN	91.0
3690	3690	9158152	ELVARLI	Wardrobes	1572.0	SR 1,765	True	https://www.ikea.com/sa/en/p/elvarli-2-section...	No	2 sections, 135x51x222-350 cm	Ehén Johansson	50.0	NaN	135.0
3691	3691	99157541	ELVARLI	Wardrobes	924.0	SR 1,050	True	https://www.ikea.com/sa/en/p/elvarli-2-section...	No	2 sections, 175x51x222-350 cm	Ehén Johansson	50.0	NaN	175.0
3692	3692	69157573	ELVARLI	Wardrobes	2745.0	SR 3,130	True	https://www.ikea.com/sa/en/p/elvarli-3-section...	No	3 sections, 178x51x222-350 cm	Ehén Johansson	50.0	NaN	178.0
3693	3693	69157376	ELVARLI	Wardrobes	1231.0	SR 1,535	True	https://www.ikea.com/sa/en/p/elvarli-2-section...	No	2 sections, 175x51x222-350 cm	Ehén Johansson	50.0	NaN	175.0

3694 rows x 14 columns

Finally, we found the dataset “Interior Design Dataset – IKEA”(2) based on IKEA products and rooms (with IKEA products), it was created for Style search – what is partially one of our purposes. Loaded it using browser from GitHub.



Interior Design Dataset - IKEA

Dataset was collected from IKEA.com website for the purpose of building Style Search engine. Please note that all images are property of IKEA.COM and are only allowed for non-commercial use.

It consists of :

- 2193 object (product) photos.
- 298 context (room scene) photos in which those objects appear.
- Text descriptions for products.
- Ground truth information on which items appear in which rooms.

We group together objects of the same category (chair, table, sofa, etc).

Room images:	Object images:	Description:
		You sit comfortably thanks to the armrests. There's a natural and living feeling of wood, as knots and other marks remain on the surface.

First of all, we need to open Jupyter Lab and create an “.ipynb” file. Then, import Python Pandas. As our dataset has “.p” – pickle extension, we need to use Pandas function read_pickle(). Here is screenshot of reading Dataset:

```
[1]: import pandas as pd
mydataset=pd.read_pickle('Downloads/Ikea-master/text_data/products_dict.p')
mydataset

[5]: {'500.210.76': {'name': 'MLA',
                  'type': 'Easel',
                  'color': 'Softwood,white',
                  'size': '',
                  'img': 'images/objects/500.210.76.jpg',
                  'desc': 'View more product information Your child can use it for different purposes. The easel has a whiteboard on one side and a blackboard on the other.'},
      '802.538.09': {'name': 'KURA',
                  'type': 'Reversible bed',
                  'color': 'White,pine',
                  'size': '90x200 cm',
                  'img': 'images/objects/802.538.09.jpg',
                  'desc': 'View more product information Turned upside down the bed quickly converts from a low to a high bed.'},
      '802.962.48': {'name': 'STICKAT',
                  'type': 'Quilt cover and pillowcase',
                  'color': 'Turquoise,lilac',
                  'size': '150x200,50x80 cm',
                  'img': 'images/objects/802.962.48.jpg',
                  'desc': 'Made from 100% cotton, a natural material thats soft against your childs skin and gets softer with every wash.'},
      '502.962.97': {'name': 'STICKAT',
                  'type': 'Bed pocket',
                  'color': 'Orange',
                  'size': '39x30 cm',
                  'img': 'images/objects/502.962.97.jpg',
                  'desc': 'Clever storage solution that you can hang on our childrens beds.'},
      '203.086.97': {'name': 'TROFAST',
                  'type': 'Frame',
                  'color': 'Light white stained pine',
                  'size': '94x52 cm',
                  'img': 'images/objects/203.086.97.jpg',
                  'desc': 'A playful and sturdy storage series for storing and organising toys, sitting, playing and relaxing.'},
      '302.980.23': {'name': 'TROFAST',
                  'type': 'Storage box',
                  'color': 'Orange',
                  'size': '42x38x23 cm',
                  'img': 'images/objects/302.980.23.jpg',
                  'desc': 'Fits in TROFAST frames.'},
      '501.150.62': {'name': 'TROFAST',
                  'type': 'Storage box',
```

To see it in more convenient way lets convert it to DataFrame and transpose:

```
df=pd.DataFrame(data=mydataset)
df_tr=df.T # transposed to look better
df_tr
```

	name	type	color	size	img	desc
500.210.76	MLA	Easel	Softwood,white		images/objects/500.210.76.jpg	View more product information Your child can u...
802.538.09	KURA	Reversible bed	White,pine	90x200 cm	images/objects/802.538.09.jpg	View more product information Turned upside do...
802.962.48	STICKAT	Quilt cover and pillowcase	Turquoise,lilac	150x200,50x80 cm	images/objects/802.962.48.jpg	Made from 100% cotton, a natural material that...
502.962.97	STICKAT	Bed pocket	Orange	39x30 cm	images/objects/502.962.97.jpg	Clever storage solution that you can hang on o...
203.086.97	TROFAST	Frame	Light white stained pine	94x52 cm	images/objects/203.086.97.jpg	A playful and sturdy storage series for storin...
...
302.673.14	TSINGE	Chair cushion, outdoor	Blue	50x49 cm	images/objects/302.673.14.jpg	You can vary the look of your outdoor area sim...
902.818.59	BEHNDIG	Glass	Clear glass	30 cl	images/objects/902.818.59.jpg	Can be stacked inside one another to save spac...
101.929.56	KALAS	Mug	Assorted colours		images/objects/101.929.56.jpg	Great for parties and everyday meals. Made of ...
690.203.31	KUNGSHOLMEN,KUNGS	2-seat sofa, outdoor	Black-brown	160x80x73 cm	images/couch/690.203.31.jpg	
102.670.51	KUNGSHOLMEN	Stool, outdoor	Black-brown		images/objects/102.670.51.jpg	By combining different seating sections you ca...

2191 rows x 6 columns

Basically, we can use the Python Pandas function to read datasets in various formats, like we did.

Cite for “Interior Design Dataset – IKEA”:

```
@inproceedings{FedCSIS201756,  
author={Ivona Tautkute and Aleksandra Możejko and Wojciech Stokowiec and Tomasz  
Trzcíński and Łukasz Brocki and Krzysztof Marasek},  
pages={1275--1282},  
title={What Looks Good with my Sofa: Multimodal Search Engine for Interior  
Design},  
booktitle={Proceedings of the 2017 Federated Conference on Computer Science and  
Information Systems},  
year={2017},  
editor={M. Ganzha and L. Maciaszek and M. Paprzycki},  
publisher={IEEE},  
doi={10.15439/2017F56},  
url={http://dx.doi.org/10.15439/2017F56},  
volume={11},  
series={Annals of Computer Science and Information Systems}  
}
```

When data is read and loaded, it is still not in machine readable format. Therefore, we need to transform the data to a more accurate and readable format. Here Data Processing and Discovery comes to help.

There are two types of data processing: real-time and distributed processing.

- Distributed Processing is a model where software system components are shared among many computers in order to increase efficiency. Distributed Processing is classified into Cloud Computing and Hadoop.
- Cloud Computing is a system where information being accessed is found remotely in the cloud or virtual space. The service allows users to store files on remote servers and takes all the heavy lifting in processing data.

- Hadoop is a set of open-source programs and procedures that anyone can use as a basis for their big data operations. It provides a huge storage for any kind of data, processing power and ability to handle virtually limitless concurrent tasks or jobs.
- A real-time processing takes an input and executes it in very short time, providing quick output. Examples of such systems are traffic control, heart rate monitor. A real-time processing is classified into: In-memory computing and Data Stream Processing.

3.4 Data processing and discovery technologies for the collected and stored datasets

In-memory computing is a software where data stored in RAM across a cluster of computers can be processed in parallel. RAM is much faster than usual spinning disk that allows you to get immediate actions and responses.

Data Stream Processing is technology that queries continuous data streams and as soon as it is received, immediately takes action over it.

For better analyzing and processing of data, data discovery was introduced. It is a process of collecting data from different datasets, and then cleaning, filtering, integrating and separating it in order to create one relevant and accurate dataset. There are four categories of Data Discovery Tools: Self-service Data Preparation, Smart Data Discovery, Search-based Data Discovery, Visual Data Discovery.

Visual Data Discovery is a structure that blends data from different sources into an in-memory store, that is linked to an interactive visualization.

Search-based Data Discovery maps structured and unstructured data into categorized architecture of dimensions that users can explore through a searching interface.

Smart Data Discovery the system automatically analyzes and generates explanations to any attribute, generating facts about your data, including the drivers of the results, key segments that influence behavior, and anomalies where the data is not aligned with expected patterns.

Self-service Data Preparation is a tool that is often combined with visual data capabilities to support (through profiling, structuring and enriching) the data discovery process with the ability to integrate data.

In our project the best choice of data processing system is real-time processing. The data will be stored in IMDG (In-Memory Data Grid) with continuous event streams. IMDG is a structure that allows storing data in distributed memory on the main server and provides fast parallel processing and analysis.

3.5 Wrangling of datasets with Python Pandas

Data Wrangling (described based on two downloaded datasets): First of all, let's discover our columns in dataset IKEA SA Furniture.

```
[2]: mydataset.dtypes
```

```
[2]: Unnamed: 0          int64
     item_id          int64
     name            object
     category        object
     price          float64
     old_price       object
     sellable_online    bool
     link            object
     other_colors      object
     short_description object
     designer         object
     depth          float64
     height         float64
     width          float64
     dtype: object
```

Now we studied how to group information together to look easier, as an example we distributed designers over the categories of products:

```
: mydataset.groupby(by=['category', 'designer']).size()
```

```
: category    designer
Bar furniture Carina Bengs          2
              Ehlén Johansson       9
              Francis Cayouette      3
              Henrik Preutz          1
              IKEA of Sweden         2
              ..
Wardrobes     L Hilland/J Karlsson   2
              Ola Wihlborg           4
              Ola Wihlborg/Ehlén Johansson/IKEA of Sweden 1
              Ola Wihlborg/IKEA of Sweden 11
              T Winkel/T Jacobsen    2
Length: 844, dtype: int64
```

```
:
```

By the following code, we can check do we have any missing values, and according to output we miss depth, height and width in some cases

```
mydataset.isna().sum()

Unnamed: 0      0
item_id        0
name           0
category       0
price          0
old_price      0
sellable_online 0
link           0
other_colors   0
short_description 0
designer        0
depth         1463
height        988
width         589
dtype: int64
```

As our project is intended on recognizing available space, it is crucial to know the height and width of the product, so let eliminate rows with missing values of height and width

```
mydataset_shrunked=mydataset.dropna(axis=0, how='any')
```

```
mydataset_shrunked.isna().sum()
```

```
Unnamed: 0      0
item_id        0
name           0
category       0
price          0
old_price      0
sellable_online 0
link           0
other_colors   0
short_description 0
designer        0
depth          0
height         0
width          0
dtype: int64
```

Next, we are going to check uniqueness of values:

```
mydataset_shrunked.nunique()
```

```
Unnamed: 0      1899
item_id        1602
name           289
category        17
price          729
old_price      248
sellable_online  2
link          1602
other_colors     2
short_description 992
designer         191
depth          108
height         178
width          241
dtype: int64
```

We observed that item_id, which actually should be unique is repeated, so let's check which items appear more than one time and how many times:

```
: repeated_items=mydataset_shrunked.groupby(by='item_id').size().sort_values(ascending=False)
```

```
: repeated_items
```

```
: item_id
89302971    3
90269697    3
50468953    3
80362974    3
29252912    3
..
60344759    1
60351900    1
60364295    1
60365153    1
116595      1
Length: 1602, dtype: int64
```


Then, what if we don't need an item to be twice in our Dataset, with the help of next code we can remove repetitions:

mydataset_shrunked[mydataset_shrunked.duplicated(subset='item_id', keep=False)].sort_values('item_id')

Unnamed: 0	item_id	name	category	price	old_price	sellable_online	link	other_colors	short_description	designer	depth	height	width	
1365	1365	121766	INGOLF	Chairs	395.0	No old price	True	https://www.ikea.com/sa/en/p/ingolf-bar-stool-...	No	Bar stool with backrest, 74 cm	Carina Bengs	45.0	102.0	4
11	11	121766	INGOLF	Bar furniture	395.0	No old price	True	https://www.ikea.com/sa/en/p/ingolf-bar-stool-...	No	Bar stool with backrest, 74 cm	Carina Bengs	45.0	102.0	4
1845	1845	165213	URBAN	Children's furniture	225.0	No old price	True	https://www.ikea.com/sa/en/p/urban-junior-chair-...	No	Junior chair	Carl Öjerstam	48.0	79.0	4
1514	1514	165213	URBAN	Chairs	225.0	No old price	True	https://www.ikea.com/sa/en/p/urban-junior-chair-...	No	Junior chair	Carl Öjerstam	48.0	79.0	4
141	141	228705	BRIMNES	Beds	1095.0	No old price	True	https://www.ikea.com/sa/en/p/brimnes-day-bed-f-...	No	Day-bed frame with 2 drawers, ...	K Hagberg/M Hagberg	53.0	21.0	8
470	470	99297144	JONAXEL	Bookcases & shelving units	170.0	No old price	True	https://www.ikea.com/sa/en/p/jonaxel-frame-wit-...	No	Frame with mesh baskets, 50x5...	IKEA of Sweden	51.0	70.0	5
378	378	99300839	BESTÅ	Bookcases & shelving units	1620.0	No old price	True	https://www.ikea.com/sa/en/p/besta-storage-com-...	Yes	Storage combination w glass doors, ...	IKEA of Sweden/Marcus Arvonen	42.0	192.0	12
3344	3344	99300839	BESTÅ	TV & media furniture	1620.0	No old price	True	https://www.ikea.com/sa/en/p/besta-storage-com-...	Yes	Storage combination w glass doors, ...	IKEA of Sweden/Marcus Arvonen	42.0	192.0	12
1853	1853	99323614	SMÅÖÖRA	Nursery furniture	370.0	No old price	True	https://www.ikea.com/sa/en/p/smagoera-changing-...	No	Changing tbl/bookshelf w 1 shlf ut	IKEA of Sweden	40.0	91.0	6
2860	2860	99323614	SMÅÖÖRA	Tables & desks	370.0	No old price	True	https://www.ikea.com/sa/en/p/smagoera-changing-...	No	Changing tbl/bookshelf w 1 shlf ut	IKEA of Sweden	40.0	91.0	6

562 rows x 14 columns

Using head() we can observe 5 top rows from the dataset:

```
[34]: NoRep_mydataset.head()
```

```
[35]:
```

	Unnamed: 0	item_id	name	category	price	old_price	sellable_online	link	other_colors	short_description	designer	depth	height	width
3	3	80155205	STIG	Bar furniture	69.0	No old price	True	https://www.ikea.com/sa/en/p/stig-bar-stool-w-...	Yes	Bar stool with backrest, 74 cm	Henrik Preutz	50.0	100.0	60.0
4	4	30180504	MORBERG	Bar furniture	225.0	No old price	True	https://www.ikea.com/sa/en/p/morberg-wall-moun-...	No	Wall-mounted drop-leaf table, ...	Marcus Arvonen	60.0	43.0	74.0
5	5	10122847	INGOLF	Bar furniture	345.0	No old price	True	https://www.ikea.com/sa/en/p/ingolf-bar-stool-...	No	Bar stool with backrest, 63 cm	Carina Bengs	45.0	91.0	40.0
6	6	70404875	FRANKLIN	Bar furniture	129.0	No old price	True	https://www.ikea.com/sa/en/p/franklin-bar-stoo-...	No	Bar stool with backrest, foldable, ...	K Hagberg/M Hagberg	44.0	95.0	50.0
8	8	50406465	FRANKLIN	Bar furniture	129.0	No old price	True	https://www.ikea.com/sa/en/p/franklin-bar-stoo-...	No	Bar stool with backrest, foldable, ...	K Hagberg/M Hagberg	44.0	95.0	50.0

With another IKEA Dataset we also tried to do wrangling. Below is how we grouped by specific columns, so we can see which style contains which items:

```
1: df_tr.groupby(by=['name', 'type']).size()
```

```
1: name      type      size
ADDE      Chair      2
AGAM      Junior chair  2
AGEN      Chair      1
AINA      Curtains, 1 pair  3
          Fabric      2
          ..
VYSSA SNOSA Mattress for extendable bed  1
YDDINGEN   High cabinet      1
          Wash-stand with 2 drawers      1
          Wash-stand with 2 drawers,1 door  1
ZAMIOCULCAS Potted plant      1
Length: 1595, dtype: int64
```

3.6 Visualization of results

In this part of our homework we are familiarized with Python Matplotlib, for this purpose we need to add “%matplotlib inline”. It has a function plot where we can choose the plot type, for example, bar, area barh, box, hexbin, hist, scatter, line, pie.

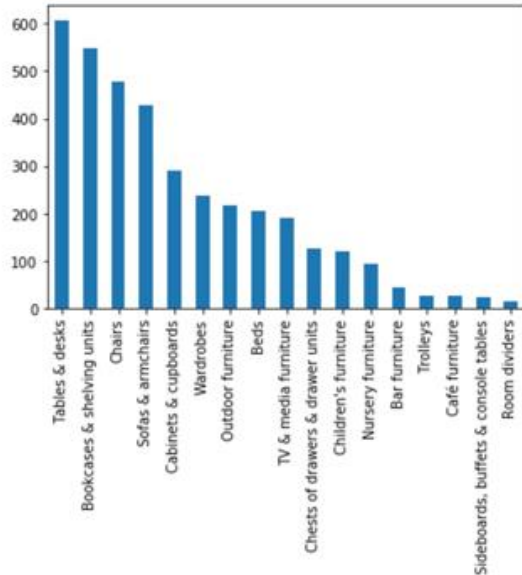
```
import pandas as pd
import numpy as np
%matplotlib inline
```

```
import matplotlib.pyplot as plt
```

We are going to work on IKEA SA Furniture dataset first. Let's try to illustrate the bar graph, that will show the quantity of furniture, that is available online:

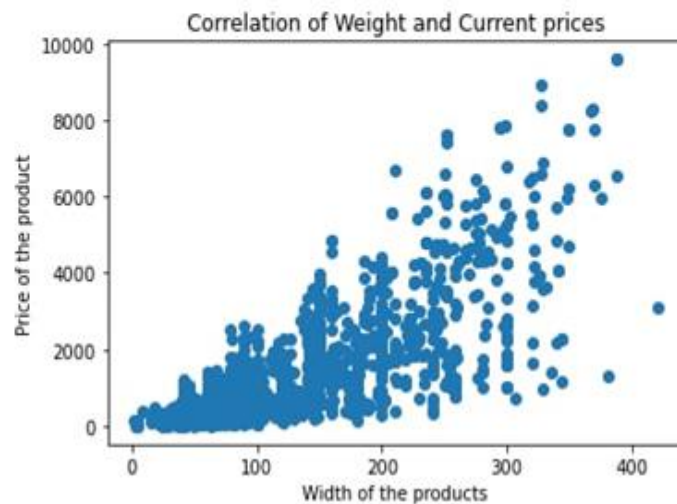
```
[40]: mydataset[mydataset["sellable_online"]]["category"].value_counts().plot(kind="bar")
```

```
[40]: <matplotlib.axes._subplots.AxesSubplot at 0x11934dd10>
```



From this example of correlation of prices and weight we can see that furniture with size from near 0 to 100 has the biggest number of products. Also, the wider the object the bigger the price.

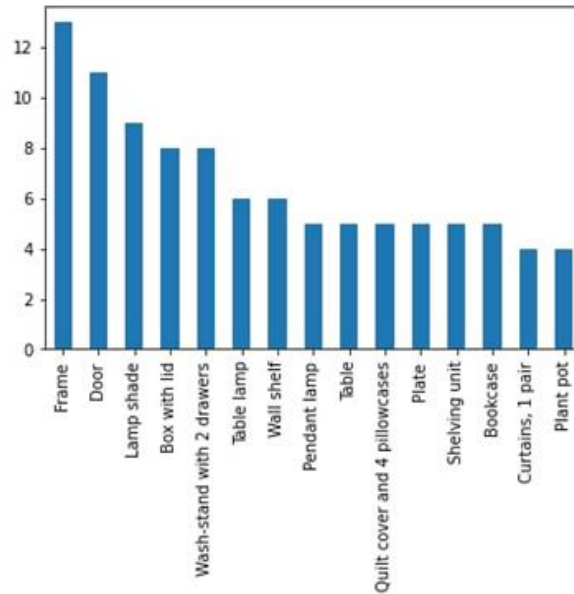
```
[31]: plt.scatter(NoRep_mydataset["width"],NoRep_mydataset["price"])
plt.xlabel("Width of the products")
plt.ylabel("Price of the product")
plt.title("Correlation of Weight and Current prices")
plt.show()
```



In this graph we filter the types of furniture by its color in this case it is white. We can see the correlation of numbers of furniture types with white colors.

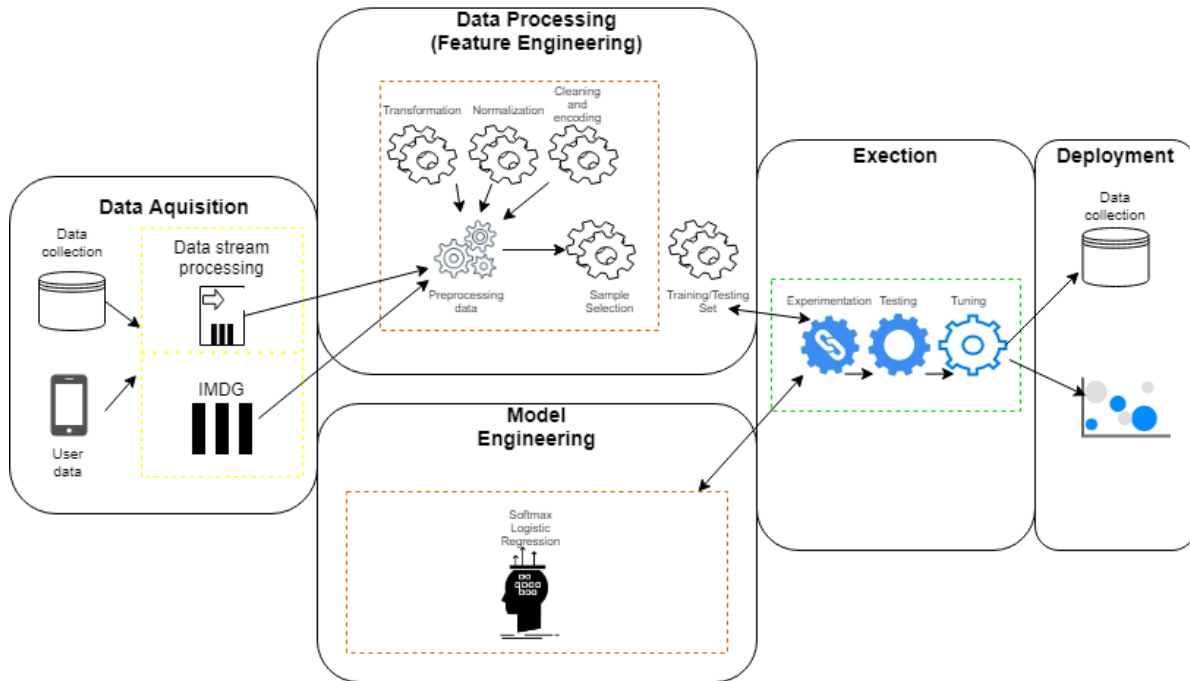
```
[11]: %matplotlib inline
df_tr[df_tr["color"]=="White"]["type"].value_counts().head(15).plot(kind="bar")
```

```
[11]: <matplotlib.axes._subplots.AxesSubplot at 0x23aba158160>
```



4. Machine Learning

4.1 Machine Learning Life Cycle Architecture



In our project we are developing application that can detect present furniture in room and based on its analysis, searches for suitable to design items in our database. Also, detected items will contribute to our datasets that will improve the quality of identification.

The model's concept is pattern recognition (object recognition). In our Machine Learning model, we are going, firstly, transform the data (pictures of furniture) into grayscale in order to get vectors of pixels and use their arrays in training the model. For better model analyzing we use Gradient Descent Method for optimization and finding minimal cost function.

When we start a Session we train the model and then feed the test dataset and get our output as processed. In last step, we would like to evaluate the results of our model and feed it test data to identify what the test object is. In future, we would like to add color recognition algorithm and size calculation based of existing algorithms as Apple Measure for better design offers.

4.2 Choosing features to use/extract from data

The main feature we extract from data is array that is created by transformation of image pixels.

Then for understanding the space measurement and overall design we also would like to extract colors and sizes.

Also, in future we would like to add category feature in order to create and use categorical dataset for furniture and if, for example, our AI identifies that item refers to one existing category in database, it will offer others items from the same class.

4.3 Classify machine learning types for the selected features.

Basically, our Machine Learning's type is classification that predicts categorical outcome. In order to train our model, we input the X array of pixels and Y of labels. This type of Machine Learning is called Supervised, where the feature vector X and the objective value Y are given. In supervised learning we are developing a finely-tuned predictor function, $h(x)$, hypothesis. Hypothesis is a certain function that we believe (or hope) is similar to the true function, the target function that we want to model.

The primary goal of any supervised learning algorithm is to minimize the predictor error while defining a hypothesis based on training data. There are two techniques of supervised learning: Regression and Classification.

For our project Classification is curtail idea. Classification defines boundary that there is a clear separation of the output variables. For example, in our project we distinguish chair and table, clock and mirror, etc.

4.4 Getting started with TensorFlow Installation

First, we need to install TensorFlow on our notebook, before we install it, we should sure that we have Python environment variables, next step is to check if we have, we need to install pip virtual environment as we see in the command below.

As installation is done we should run command: `pip install TensorFlow` because I had it before I just used the command `–upgrade` to check new versions and download it to computer.

```

C:\Users\User>pip install --user virtualenv
Collecting virtualenv
  Downloading virtualenv-20.1.0-py2.py3-none-any.whl (4.9 MB)
    |#####| 4.9 MB 62 kB/s
Collecting distlib<1,>=0.3.1
  Downloading distlib-0.3.1-py2.py3-none-any.whl (335 kB)
    |#####| 335 kB 100 kB/s
Requirement already satisfied: six<2,>=1.9.0 in x:\python\anaconda3\lib\site-packages (from virtualenv) (1.15.0)
Requirement already satisfied: filelock<4,>=3.0.0 in x:\python\anaconda3\lib\site-packages (from virtualenv) (3.0.12)
Collecting appdirs<2,>=1.4.3
  Downloading appdirs-1.4.4-py2.py3-none-any.whl (9.6 kB)
Installing collected packages: distlib, appdirs, virtualenv
WARNING: The script virtualenv.exe is installed in 'C:\Users\User\AppData\Roaming\Python\Python38\Scripts' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed appdirs-1.4.4 distlib-0.3.1 virtualenv-20.1.0

C:\Users\User>pip install --upgrade tensorflow
Collecting tensorflow
  Downloading tensorflow-2.3.1-cp38-cp38-win_amd64.whl (342.5 MB)
    |#####| 342.5 MB 7.8 kB/s
Collecting grpcio<=1.8.6
  Downloading grpcio-1.33.2-cp38-cp38-win_amd64.whl (2.7 MB)
    |#####| 2.7 MB 1.7 MB/s
Collecting opt-einsum<=2.3.2
  Downloading opt_einsum-3.3.0-py3-none-any.whl (65 kB)
    |#####| 65 kB 1.0 MB/s
Requirement already satisfied, skipping upgrade: wrapt<=1.11.1 in x:\python\anaconda3\lib\site-packages (from tensorflow) (1.11.2)
Collecting absl-py<=0.7.0
  Downloading absl_py-0.11.0-py3-none-any.whl (127 kB)
    |#####| 127 kB 1.6 MB/s
Collecting termcolor<=1.1.0
  Downloading termcolor-1.1.0.tar.gz (3.9 kB)
Collecting tensorboard<3,>=2.3.0
  Downloading tensorboard-2.4.0-py3-none-any.whl (10.6 MB)
    |#####| 10.6 MB 491 kB/s
Collecting keras-preprocessing<1.2,>=1.1.1
  Downloading Keras Preprocessing-1.1.2-py2.py3-none-any.whl (42 kB)
    |#####| 42 kB 441 kB/s
Collecting google-pasta<=0.1.8
  Downloading google_pasta-0.2.0-py3-none-any.whl (57 kB)
    |#####| 57 kB 1.5 MB/s
Requirement already satisfied, skipping upgrade: h5py<2.11.0,>=2.10.0 in x:\python\anaconda3\lib\site-packages (from tensorflow) (2.10.0)
Requirement already satisfied, skipping upgrade: six<=1.12.0 in x:\python\anaconda3\lib\site-packages (from tensorflow) (1.15.0)
Requirement already satisfied, skipping upgrade: wheel<=0.26 in x:\python\anaconda3\lib\site-packages (from tensorflow) (0.34.2)
Collecting astunparse<=1.6.3
  Downloading astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Requirement already satisfied, skipping upgrade: numpy<1.19.0,>=1.16.0 in x:\python\anaconda3\lib\site-packages (from tensorflow) (1.18.5)
Collecting tensorflow-estimator<2.4.0,>=2.3.0
  Downloading tensorflow_estimator-2.3.0-py2.py3-none-any.whl (459 kB)
    |#####| 459 kB 1.3 MB/s
Collecting protobuf<=3.9.2
  Downloading protobuf-3.13.0-py2.py3-none-any.whl (438 kB)

```

Here to check if our installation is ok, we need to run simple command like Hello World or Hi, TensorFlow, as we see in the snapshot it works.

The screenshot shows a Jupyter Notebook window titled 'Untitled4'. The notebook contains a code cell with the following Python code:

```

In [4]: import tensorflow.compat.v1 as tf
        tf.disable_v2_behavior()
        hi = tf.constant("Hi, TensorFlow")
        sess = tf.Session()
        print(sess.run(hi))
        a = tf.constant(30)
        b = tf.constant(10)
        print(sess.run(a+b))

```

The output of the code cell is:

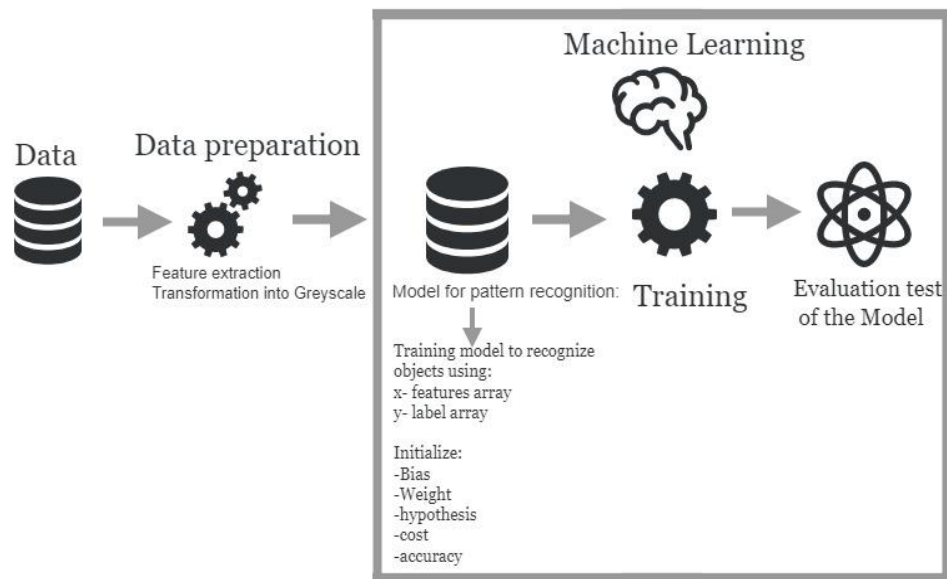
```

b'Hi, TensorFlow'
40

```

The Jupyter Notebook interface includes a top bar with the Jupyter logo and the text 'jupyter Untitled4 Last Checkpoint: 5 minutes ago (unsaved changes)'. Below the top bar is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, Help. A toolbar with various icons for file operations and execution is also visible. The bottom of the notebook shows the input prompt 'In []:' for the next cell.

4.5 Logistic regression



First, we prepared our data samples for logistic regression classification `train_images`; `test_images`:

```
import os, cv2, itertools
import numpy as np
import pandas as pd
from tqdm import tqdm

import tensorflow as tf
import tensorflow.compat.v1 as tf
tf.compat.v1.disable_eager_execution()
import matplotlib.pyplot as plt
%matplotlib inline

TRAIN_DIR="Desktop/train/"
TEST_DIR="Desktop/test/"

IMG_SIZE=50

CHANNELS = 3
sample_size=100

train_bed=[TRAIN_DIR+i for i in os.listdir(TRAIN_DIR) if 'bed' in i]
train_clock=[TRAIN_DIR+i for i in os.listdir(TRAIN_DIR) if 'clock' in i]
train_images=np.hstack([np.random.choice(train_bed, sample_size, replace=False),np.random.choice(train_clock, sample_size, replace=False)])

test_images = [TEST_DIR+i for i in os.listdir(TEST_DIR)]
```

These two functions read our image and transforms to array/vector form:

```
def read_image(file_path):
    img=cv2.imread(file_path,cv2.IMREAD_COLOR)
    img_res=cv2.resize(img,(IMG_SIZE,IMG_SIZE),interpolation=cv2.INTER_CUBIC)
    #plt.imshow(img_res, cmap='gray')
    #plt.show()
    return img_res

def prep_data(images):
    m = len(images)
    n_x = IMG_SIZE * IMG_SIZE*CHANNELS
    X = np.zeros((m,IMG_SIZE ,IMG_SIZE, CHANNELS), dtype=np.uint8)
    y = np.zeros((1, m))
    print ("X shape is {}".format(X.shape))

    for i, image_file in enumerate(images):
        image = read_image(image_file)
        X[i,:] = read_image(image_file)
        if 'bed' in image_file.lower():
            y[0, i] = 1
        elif 'clock' in image_file.lower():
            y[0, i] = 0
        else:
            y[0, i] = image_file.split('/')[1].split('.')[0]
        #if i%10 == 0: print('Processed {} of {}'.format(i, m))
    return X, y
```


Here we can see how logistic regression is used, exactly when we define w, b, hypothesis (using sigmoid), cost. Next, we use GradientDescentOptimizer with learn and start feeding the model.

```
X_training, y_training = prep_data(train_images)
X_testing, y_testing = prep_data(test_images)

X_training = X_training.reshape(X_training.shape[0], IMG_SIZE*IMG_SIZE*CHANNELS)
X_testing = X_testing.reshape(X_testing.shape[0], -1)

X_training = X_training/255
X_testing = X_testing/255

y_training = y_training.T
y_testing = y_testing.T

var_x = tf.placeholder(tf.float32, [None, IMG_SIZE * IMG_SIZE * CHANNELS])
var_y = tf.placeholder(tf.float32, [None, 1])

w = tf.Variable(tf.zeros([IMG_SIZE * IMG_SIZE * CHANNELS, 1]), name='weight')
b = tf.Variable(tf.zeros([1]), name='bias')

hypothesis = tf.math.sigmoid(tf.matmul(var_x, w) + b)

cost = -tf.reduce_mean(var_y * tf.math.log(hypothesis) + (1 - var_y) * tf.math.log(1 - hypothesis))
train = tf.train.GradientDescentOptimizer(0.001).minimize(cost)

predicted = tf.cast(hypothesis > 0.5, dtype=tf.float32)
accuracy = tf.reduce_mean(tf.cast(tf.equal(predicted, var_y), dtype=tf.float32))

sess = tf.InteractiveSession()
sess.run(tf.global_variables_initializer())
feed = {var_x: X_training, var_y: y_training}
for step in range(2000):
    sess.run(train, feed_dict=feed)
    if step % 100 == 0:
        print("step: ", step, "cost: ", sess.run(cost, feed_dict=feed))

h, c, a = sess.run([hypothesis, predicted, accuracy], feed_dict=feed)
print("\nHypothesis: ", h, "\nCorrect: ", c, "\nAccuracy: ", a)
```

Below is the output of our program with accuracy=0.96. For every 100 epochs the step and cost is printed :

```
print("\nHypothesis: ",h,"\nCorrect: ",c,"\nAccuracy: ",a)
```

```
X shape is (200, 50, 50, 3)
X shape is (70, 50, 50, 3)
step: 0 cost: 0.68948084
step: 100 cost: 0.5161035
step: 200 cost: 0.43859524
step: 300 cost: 0.3923957
step: 400 cost: 0.36060593
step: 500 cost: 0.33672863
step: 600 cost: 0.3177179
step: 700 cost: 0.30195427
step: 800 cost: 0.28849488
step: 900 cost: 0.27675217
step: 1000 cost: 0.26633888
step: 1100 cost: 0.25698733
step: 1200 cost: 0.2485051
step: 1300 cost: 0.24074951
step: 1400 cost: 0.23361106
step: 1500 cost: 0.22700393
step: 1600 cost: 0.22085965
step: 1700 cost: 0.215122
step: 1800 cost: 0.20974511
step: 1900 cost: 0.20469007
```

Hypothesis: [[9.98811603e-01]
[8.46427679e-01]
[7.10335374e-01]
[9.80300605e-01]
[8.88004899e-01]
[9.53049302e-01]
[9.29223418e-01]
[9.33976412e-01]
[9.56128120e-01]
[7.10335374e-01]
[9.42811847e-01]
[9.29283857e-01]
[8.44928622e-01]
[9.71595287e-01]
[3.38160038e-01]
[9.04659271e-01]
[9.78561997e-01]

[6.53443038e-02]
[6.93136454e-02]
[1.54300064e-01]
[1.41949832e-01]
[3.65658373e-01]
[7.86840916e-04]
[7.88354874e-03]
[7.90174603e-02]
[2.16781676e-01]
[4.27668154e-01]
[2.61822343e-02]
[1.78570032e-01]
[3.82486284e-02]
[4.31318432e-01]
[1.68433487e-02]
[8.36291909e-03]]

Correct: `[[1.]]`

[1.]
[1.]
[1.]
[1.]
[1.]
[1.]
[1.]
[1.]
[1.]
[1.]
[1.]
[0.]
[1.]
[1.]
[1.]
[0.]
[1.]
[1.]
[0.]
[1.]
[1.]
[1.]

```
[0.]
[0.]
[0.]
[0.]
[0.]
[0.]
[0.]
```

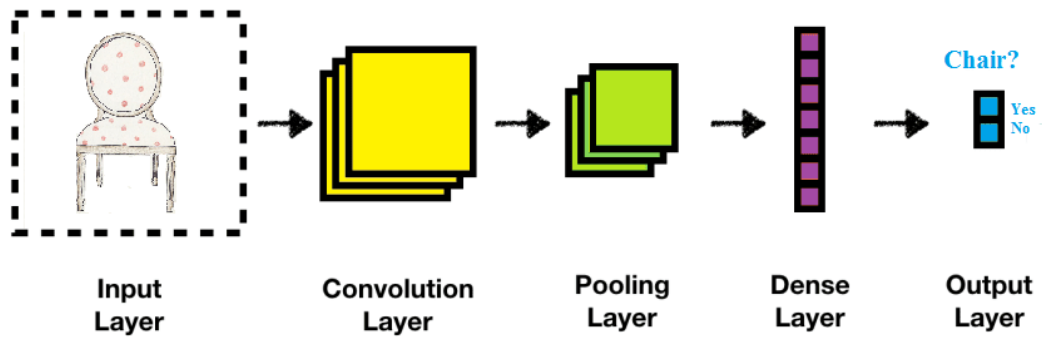
Accuracy: 0.96

5. Deep Learning

5.1 Deep Learning algorithms

The aim of our project to create an application that is able to analyze the objects in the room, their placement and color. The primary and basic algorithm, that is required, is Convolutional Neural Networks (CNN). This is most common algorithm that is used for image classification in computer vision.

The design of CNN is combination of different layers. It may include convolutional, RELU(nonlinear filter), pooling, and fully connected layers(dense layer).



Firstly, we input the image. Then in convolution layer it transformed into feature map, with shape (# of images) x (feature map height) x (feature map width) x (feature map channels). In this layer, the relationship between pixels is preserved using small squares of input data for learning.

RELU layer applies to a nonlinear function, because not all data may be in linear fashion. Also, it has advantage in solving vanishing gradient problem.

Next, in pooling the number of parameters may be reduced if the image is very large. The process also called subsampling or downsampling, in a result it retains important information.

The dense layer or fully connected layer, connects all output features from combination of all previous layers.

Finally, in the output we are given the probability of belonging to a particular class.

5.2 Experimental end-to-end DL design and implementation of a simple scenario for our application with TensorFlow.

For our project we use IKEA Furniture dataset. Firstly, we load the training and test data, identify categories and image size. Then using `create_training_data()` and `create_test_data` functions classify data by categories, transform it into grayscale format and append to arrays X for features and Y for labels. That is our convolutional layer.

```
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
import cv2
from tqdm import tqdm
from tensorflow import keras

DATADIR = "Desktop/ai_datasets/ikea-master /images"
TEST_DATADIR="Desktop/ai_datasets/ikea-master /testt"
CATEGORIES = ["Clock", "Bed","Chair","Dining table","Plant_pot","Couch"]
CATEGORIES_T = ["Clock", "Bed","Chair","Dining table","Plant_pot","Couch"]
IMG_SIZE =32

training_data = []

def create_training_data():
    for category in CATEGORIES:

        path = os.path.join(DATADIR,category)
        class_num = CATEGORIES.index(category)
        for img in tqdm(os.listdir(path)):
            try:
                img_array = cv2.imread(os.path.join(path,img) ,cv2.IMREAD_GRAYSCALE)
                new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
                training_data.append([new_array, class_num])
            except Exception as e:
                pass

create_training_data()
X = []
Y = []
for features,label in training_data:
    X.append(features)
    Y.append(label)

print(X[0].reshape(-1, IMG_SIZE, IMG_SIZE, 1))

X = np.array(X).reshape(-1, IMG_SIZE, IMG_SIZE, 1)
Y=np.array(Y)
X = X/255.0

model=keras.Sequential([keras.layers.Flatten(input_shape=X.shape[1:]),
                        keras.layers.Dense(64,activation=tf.nn.relu),
```

Next, we use a Simple Framework to build CNN model.

We add Flatten layer for flattening, connecting convolution layer and dense layer.

Dense connects all output features from combination of all previous layers.

Activation functions:

- Relu is linear function that identifies all positive values, and gives zero for all negative values of input X.
- softmax function that is used for labels classification.

```
model=keras.Sequential([keras.layers.Flatten(input_shape=X.shape[1:]),
                        keras.layers.Dense(64,activation=tf.nn.relu),
                        keras.layers.Dense(7,activation=tf.nn.softmax)
                        ])
model.compile(optimizer=tf.train.AdamOptimizer(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(X,Y,epochs=1200)

testing_data=[]
def create_test_data():
    for category in CATEGORIES_T:

        path = os.path.join(TEST_DATADIR,category)
        class_num = CATEGORIES_T.index(category)
        for img in tqdm(os.listdir(path)):
            try:
                img_array = cv2.imread(os.path.join(path,img) ,cv2.IMREAD_GRAYSCALE)
                new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
                testing_data.append([new_array, class_num])
            except Exception as e:
                pass
create_test_data()

Xt = []
Yt = []
for features,label in testing_data:
    Xt.append(features)
    Yt.append(label)

print(Xt[0].reshape(-1, IMG_SIZE, IMG_SIZE, 1))

Xt = np.array(Xt).reshape(-1, IMG_SIZE, IMG_SIZE, 1)
Yt=np.array(Yt)
Xt = Xt/255.0

test_loss,test_acc=model.evaluate(Xt,Yt)
print("test accuracy:", test_acc)
predictions=model.predict(Xt)

fig=plt.figure(figsize=(25, 12))
```


In this part, we would like to check our test data and show output of pictures and its labels.

```
fig=plt.figure(figsize=(25, 12))
for num,data in enumerate(testing_data[:40]):

    img_num = data[1]
    img_data = data[0]

    y = fig.add_subplot(4,10,num+1)
    orig = img_data
    data = img_data.reshape(-1,IMG_SIZE,IMG_SIZE,1)
    #model_out = model.predict([data])[0]
    model_out = model.predict([data])[0]

    if np.argmax(model_out) == 0:
        str_label='Clock'
    if np.argmax(model_out) == 1:
        str_label='Bed'
    if np.argmax(model_out) == 2:
        str_label='Chair'
    if np.argmax(model_out) == 3:
        str_label='Dining table'
    if np.argmax(model_out) == 4 :
        str_label='plant_pot'
    if np.argmax(model_out) == 5 :
        str_label='couch'
    y.imshow(orig,cmap='gray')
    plt.title(str_label)
    y.axes.get_xaxis().set_visible(False)
    y.axes.get_yaxis().set_visible(False)
plt.show()
```

Run 1200 epochs:

```
Epoch 1/1200
411/411 [=====] - 0s 188us/sample - loss: 1.8848 - acc: 0.2555
Epoch 2/1200
411/411 [=====] - 0s 39us/sample - loss: 1.6698 - acc: 0.3017
Epoch 3/1200
411/411 [=====] - 0s 39us/sample - loss: 1.5677 - acc: 0.3917
Epoch 4/1200
411/411 [=====] - 0s 41us/sample - loss: 1.4582 - acc: 0.4526
Epoch 5/1200
411/411 [=====] - 0s 39us/sample - loss: 1.4114 - acc: 0.4866
Epoch 6/1200
411/411 [=====] - 0s 39us/sample - loss: 1.3822 - acc: 0.4769
Epoch 7/1200
411/411 [=====] - 0s 41us/sample - loss: 1.3274 - acc: 0.5304
Epoch 8/1200
411/411 [=====] - 0s 40us/sample - loss: 1.2769 - acc: 0.5061
Epoch 9/1200
411/411 [=====] - 0s 39us/sample - loss: 1.2202 - acc: 0.5718
Epoch 10/1200
411/411 [=====] - 0s 41us/sample - loss: 1.1834 - acc: 0.5985
Epoch 11/1200
411/411 [=====] - 0s 39us/sample - loss: 1.1579 - acc: 0.5523
Epoch 12/1200
411/411 [=====] - 0s 41us/sample - loss: 1.1279 - acc: 0.5839
Epoch 13/1200
411/411 [=====] - 0s 39us/sample - loss: 1.0862 - acc: 0.6302
Epoch 14/1200
411/411 [=====] - 0s 40us/sample - loss: 1.0505 - acc: 0.6618
Epoch 15/1200
411/411 [=====] - 0s 41us/sample - loss: 0.9919 - acc: 0.7178
Epoch 16/1200
411/411 [=====] - 0s 41us/sample - loss: 1.0001 - acc: 0.6861
Epoch 17/1200
411/411 [=====] - 0s 41us/sample - loss: 0.9745 - acc: 0.7056
Epoch 18/1200
411/411 [=====] - 0s 39us/sample - loss: 0.9555 - acc: 0.7372
Epoch 19/1200
411/411 [=====] - 0s 39us/sample - loss: 0.9232 - acc: 0.7178
Epoch 20/1200
411/411 [=====] - 0s 40us/sample - loss: 0.9185 - acc: 0.6959
Epoch 21/1200
411/411 [=====] - 0s 41us/sample - loss: 0.8918 - acc: 0.7348
Epoch 1180/1200
411/411 [=====] - 0s 37us/sample - loss: 0.0278 - acc: 0.9878
Epoch 1181/1200
411/411 [=====] - 0s 35us/sample - loss: 0.0250 - acc: 0.9903
Epoch 1182/1200
411/411 [=====] - 0s 35us/sample - loss: 0.0462 - acc: 0.9830
Epoch 1183/1200
411/411 [=====] - 0s 35us/sample - loss: 0.1290 - acc: 0.9465
Epoch 1184/1200
411/411 [=====] - 0s 34us/sample - loss: 0.1243 - acc: 0.9562
Epoch 1185/1200
411/411 [=====] - 0s 35us/sample - loss: 0.0639 - acc: 0.9781
Epoch 1186/1200
411/411 [=====] - 0s 35us/sample - loss: 0.0386 - acc: 0.9878
Epoch 1187/1200
411/411 [=====] - 0s 36us/sample - loss: 0.0613 - acc: 0.9781
Epoch 1188/1200
411/411 [=====] - 0s 36us/sample - loss: 0.1220 - acc: 0.9586
Epoch 1189/1200
411/411 [=====] - 0s 37us/sample - loss: 0.4238 - acc: 0.9075
Epoch 1190/1200
411/411 [=====] - 0s 35us/sample - loss: 0.1115 - acc: 0.9586
Epoch 1191/1200
411/411 [=====] - 0s 33us/sample - loss: 0.0582 - acc: 0.9878
Epoch 1192/1200
411/411 [=====] - 0s 34us/sample - loss: 0.0598 - acc: 0.9805
Epoch 1193/1200
411/411 [=====] - 0s 35us/sample - loss: 0.0572 - acc: 0.9781
Epoch 1194/1200
411/411 [=====] - 0s 34us/sample - loss: 0.0398 - acc: 0.9854
Epoch 1195/1200
411/411 [=====] - 0s 35us/sample - loss: 0.0530 - acc: 0.9830
Epoch 1196/1200
411/411 [=====] - 0s 33us/sample - loss: 0.0488 - acc: 0.9830
Epoch 1197/1200
411/411 [=====] - 0s 34us/sample - loss: 0.0262 - acc: 0.9903
Epoch 1198/1200
411/411 [=====] - 0s 34us/sample - loss: 0.0407 - acc: 0.9854
Epoch 1199/1200
411/411 [=====] - 0s 35us/sample - loss: 0.0287 - acc: 0.9878
Epoch 1200/1200
411/411 [=====] - 0s 36us/sample - loss: 0.0356 - acc: 0.9878
```

```

100%|██████████| 9/9 [00:00<00:00, 836.50it/s]
100%|██████████| 10/10 [00:00<00:00, 967.66it/s]
100%|██████████| 7/7 [00:00<00:00, 1352.19it/s]
100%|██████████| 9/9 [00:00<00:00, 1210.21it/s]
100%|██████████| 6/6 [00:00<00:00, 1271.96it/s]
100%|██████████| 9/9 [00:00<00:00, 1194.96it/s]
[[[255]
  [255]
  [255]
  ...
  [255]
  [255]
  [255]]

 [255]
  [255]
  [255]
  ...
  [255]
  [255]
  [255]]

 [255]
  [255]
  [255]
  ...
  [255]
  [255]
  [255]]

 ...

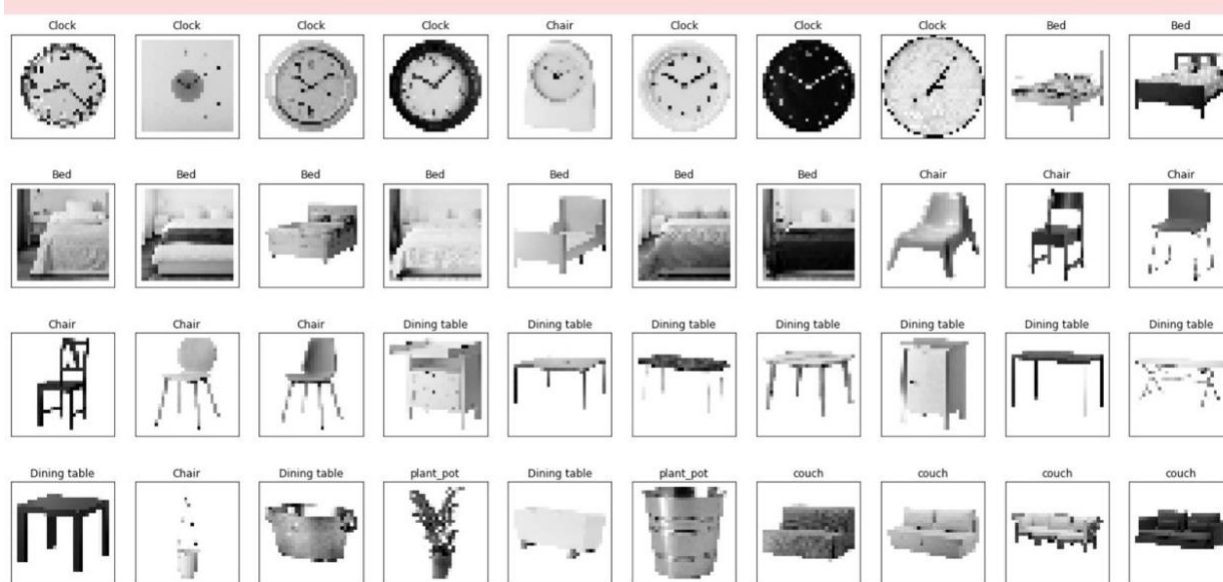
```

See the results:

```

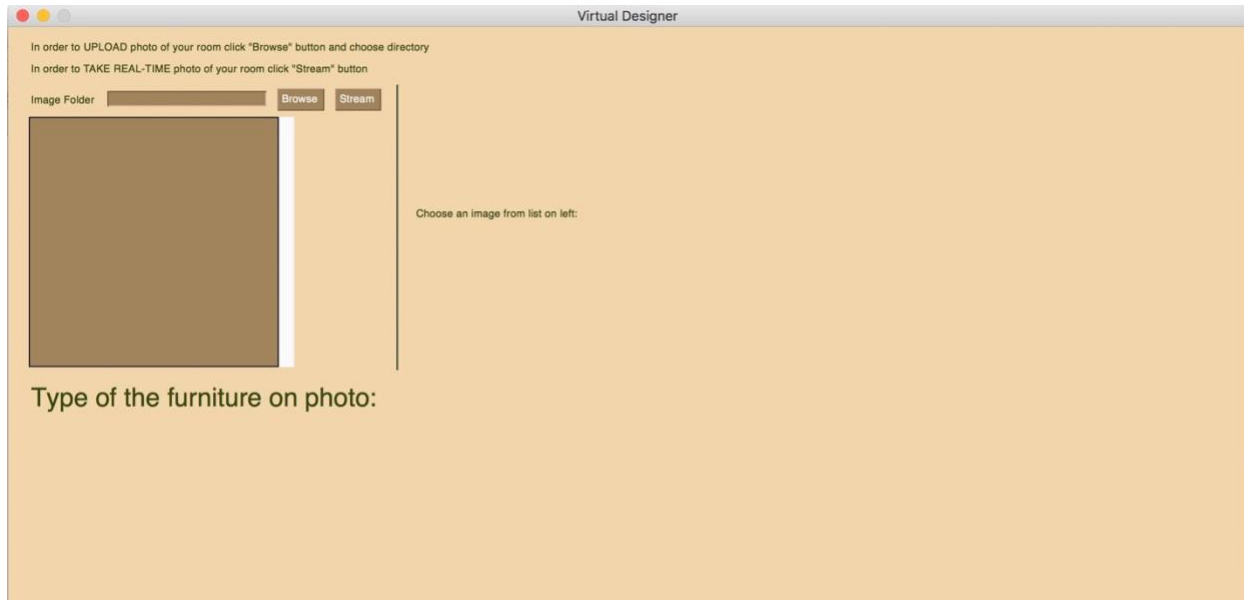
...
[255]
[255]
[255]]]]
45/45 [=====] - 0s 892us/sample - loss: 0.3976 - acc: 0.9333
test accuracy: 0.9333334

```

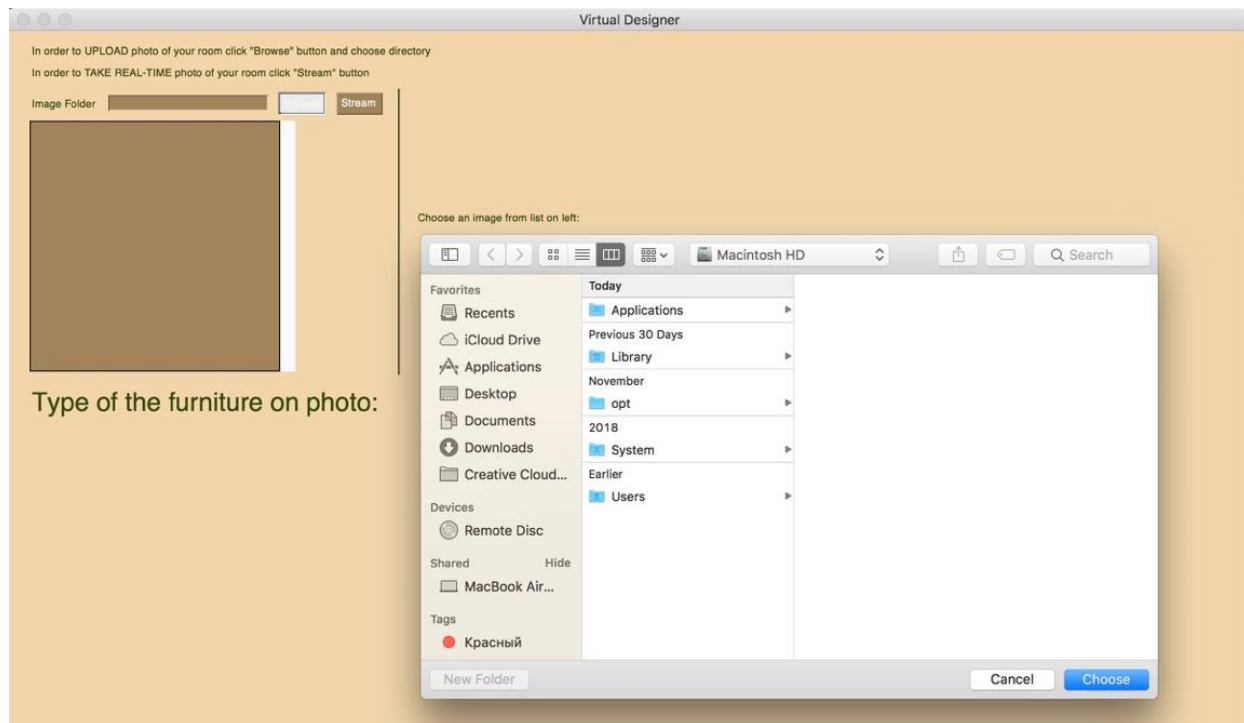


5.3 Demo

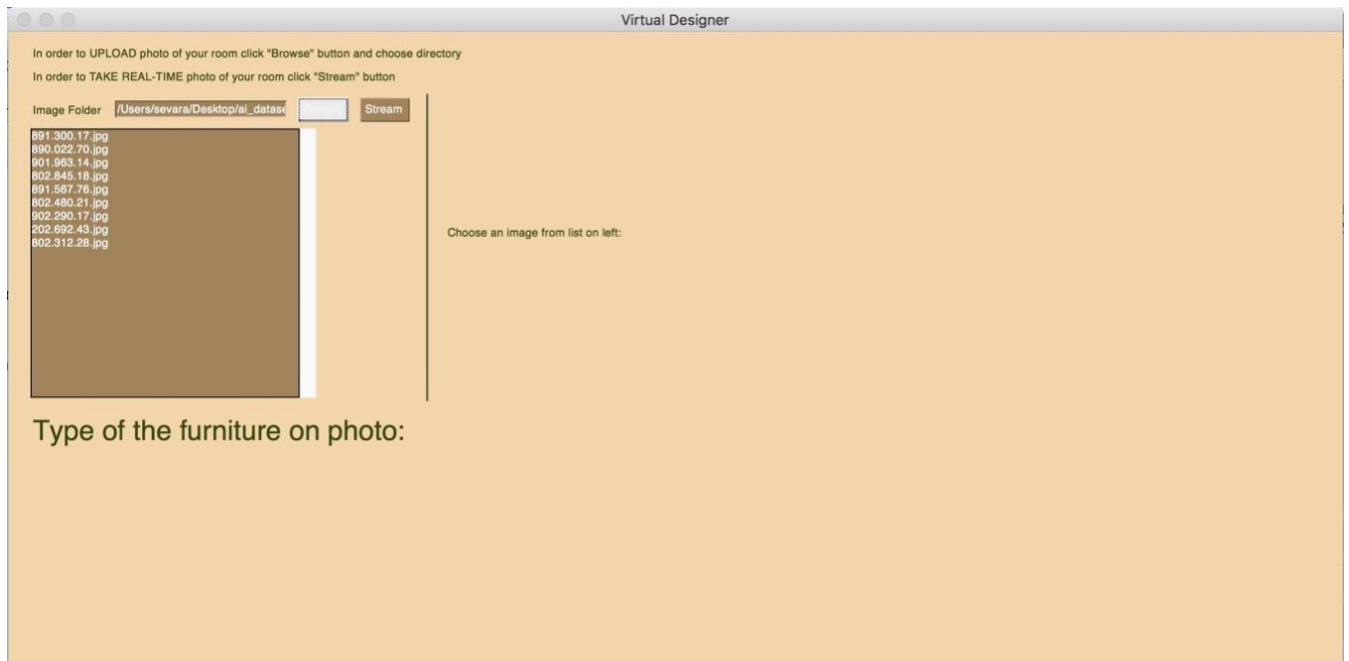
We have introduced basic demo to show how our model can be integrated to be used within application. With the help of PySimpleGUI, we created simple interface that looks like this:



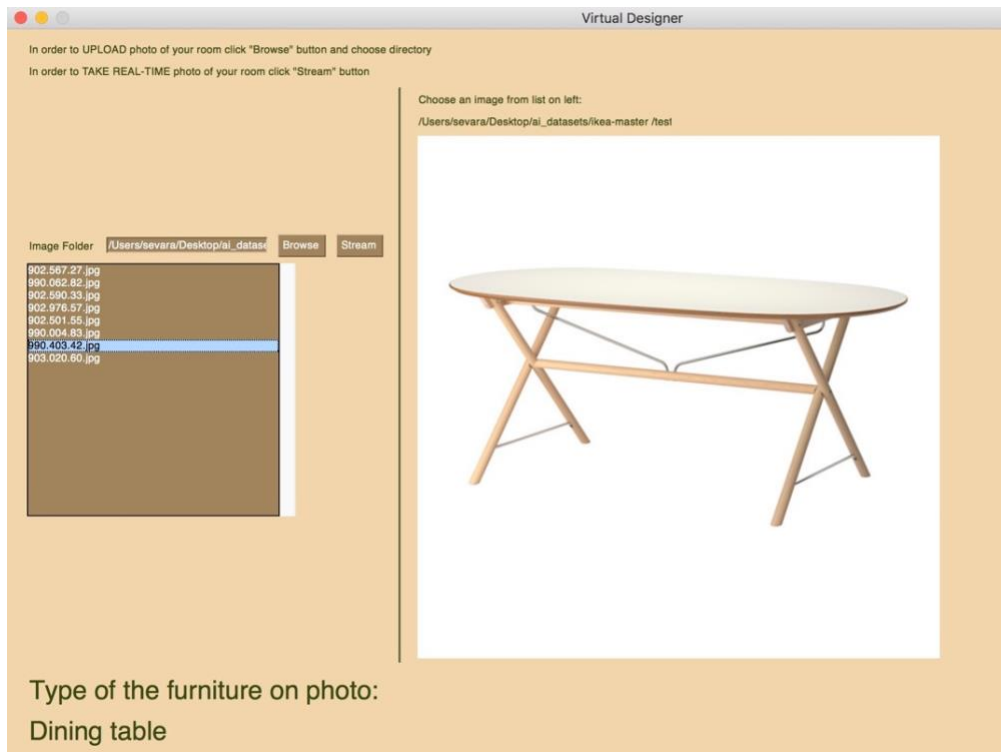
In order to upload existing photo of you room you can push Browse button and it will open your directories:



Next you will have a list of all applicable files in current directory on left:



Choose one and our model will predict its type:



Virtual Designer

In order to UPLOAD photo of your room click "Browse" button and choose directory

In order to TAKE REAL-TIME photo of your room click "Stream" button

Choose an image from list on left:

/Users/sevara/Desktop/ai_datasets/ikea-master /test

Image Folder /Users/sevara/Desktop/ai_dataase Browse Stream

891.300.17.jpg

890.022.70.jpg

901.963.14.jpg

802.845.18.jpg


891.567.78.jpg

802.480.21.jpg

902.290.17.jpg

202.692.43.jpg

802.312.28.jpg



Type of the furniture on photo:
Bed

Virtual Designer

In order to UPLOAD photo of your room click "Browse" button and choose directory

In order to TAKE REAL-TIME photo of your room click "Stream" button

Choose an image from list on left:

/Users/sevara/Desktop/ai_datasets/ikea-master /test

Image Folder /Users/sevara/Desktop/ai_dataase Browse Stream

902.874.13.jpg


003.015.28.jpg

903.347.30.jpg

002.110.88.jpg

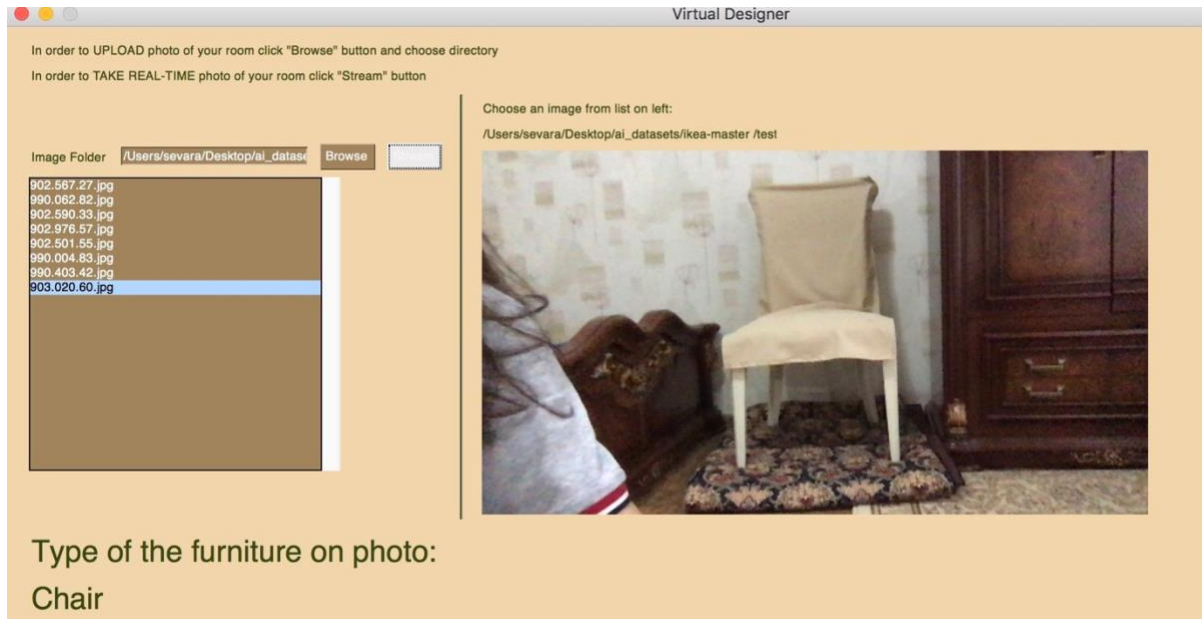
991.278.11.jpg

991.278.06.jpg

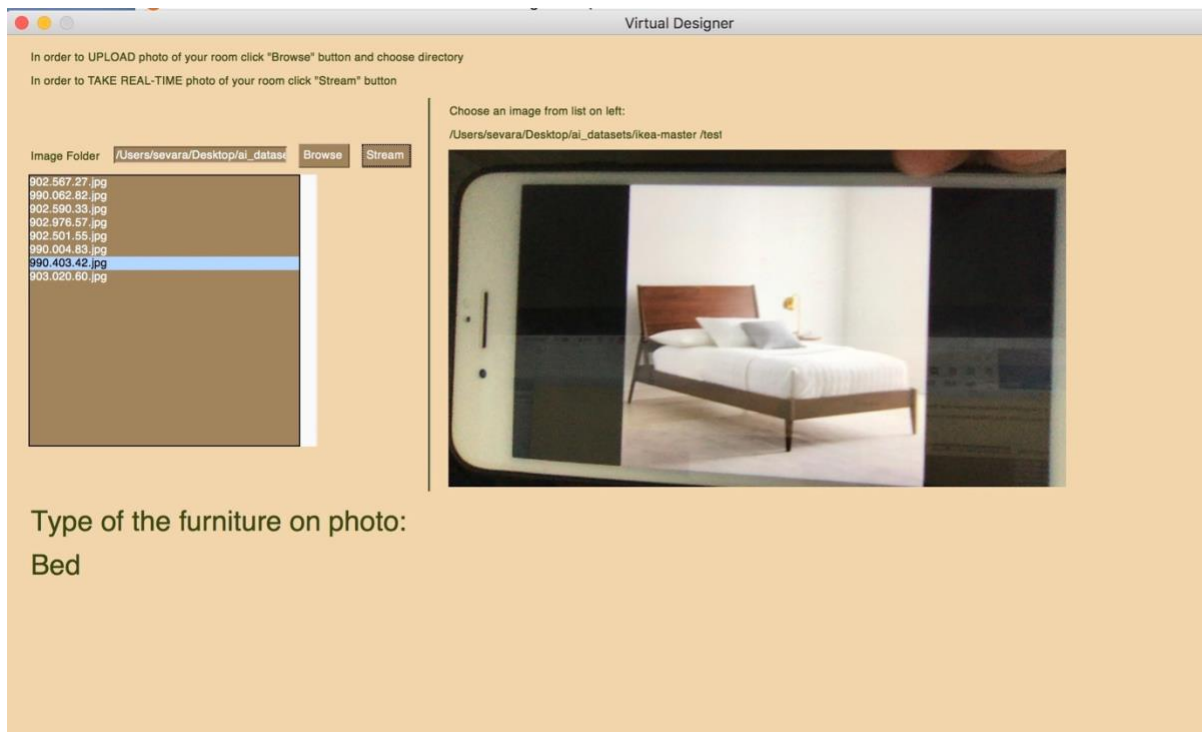


Type of the furniture on photo:
Chair

As we mentioned, our platform should also analyze room in real-time, so we included real-time frame capturing and analyzing. It can be used by pressing Stream button, it will take a capture from Camera:



As you can see it predicted furniture to be a Chair. Next, because of lack of furniture in my room, I tried to show photo from my phone to the camera. Here is a result:



Virtual Designer

In order to UPLOAD photo of your room click "Browse" button and choose directory

In order to TAKE REAL-TIME photo of your room click "Stream" button

Image Folder

/Users/sevara/Desktop/ai_dataset

Browse

902.567.27.jpg

990.062.82.jpg

902.590.33.jpg

902.976.57.jpg

902.501.55.jpg


990.004.83.jpg

990.403.42.jpg

903.020.60.jpg

Choose an image from list on left:

/Users/sevara/Desktop/ai_datasets/ikea-master/test



Type of the furniture on photo:

Dining table

6. Summary

6.1 Conclusion

The proposed project aim was to provide O2O (Online to Offline), not only for existing product/commodity market, but also to cover furniture-oriented shops, which especially suffer during the period of lockdown in their sales. Our platform will influence purchase decision by enticing customers with some features. Generating best-fit furniture according to the style of room, empty space available and preferences of user, placing it in room in the real-time with the help of AR (Augmented Reality), to name a few – all will save lots of money and time of customer, by cutting a need for designer. They were constructed using AI (Artificial Intelligence) components integration. Following the structure and guidelines of this course (AI) and of MA (Multimedia Application), we were able to implement an MVP for our product.

What we taught our model on this stage is classification of furniture. We trained it on the free available ikea-dataset of six types of furniture.

Through the scope of this course, we have applied different loss functions, initializations, optimizers and etc. Finally, with the help of our lecture materials we implemented CNN (Convolutional Neural Network) that helped us to achieve the best accuracy at 93.3%.

It was not an easy task, because unlike numbers and letters, furniture can come in different colors, shapes and styles. Additionally, vast of them are very similar in their edges. Chair and bed, for example, both have a backrest and legs, but they differ in length. So better algorithm was critical to get desired accuracy.

6.2 Future plans

Furniture classification will help in the future implementation, when we will classify not only type but also a style of particular furniture/object in room. Next upgrade will include merging of our furniture classification with object detection API given by TensorFlow itself, in order to detect multiple objects in the room.

Open issues that we faced are how to collect required number of data to train and test our model. Also, we should integrate real-time interaction, namely real-time object detection and suggestion, that is not an easy task in terms of time, money, resources and power. Possibly, on the next stage the number of problems can expand, but till this stage lots of them were clearly explained in the scope of this course.