PREDICTING THE PIECES OF FURNITURE IN LIVING ROOMS

MOHAMMED ALI, AYBUKE YALCINER AND HATICE ACAR

{mmomen574@gmail.com, aybukeyalcinerr@gmail.com,aacarhatice97@hotmail.com}

Abstract

This paper introduces some image classification methods, algorithm comparisons and evaluating steps of the accuracy. We have 10 different classes that are Armchair, Carpet, Coffee Table, Lamp, Painting, Sofa, Sofa Table, Television, Television Unit and Vases. We compare 3 different algorithms which are Convolutional Neural Network, Multilayer Perceptron and K - Nearest Neighbor. Finally, the model was trained multiple times in order to get a high accuracy. We end up with 75.28 .The accuracy can be improved and we can also add more objects as future work.

1   Introduction

Nowadays, Smart Homes provide people with more comfortable and relaxing life. Smart Homes require recognizing all pieces of the furnitures in the home to give the homeowners more control and some features in their dream homes.In addition to smart homes; home robots, interactive homes with their owners require furniture predictions. Our main aim is to contribute to smart homes. So we decided to make predictions on top 10 furnitures that is in any living room. We determine 10 furnitures that are Armchair, Carpet, Coffee Table, Lamp, Painting, Sofa, Sofa Table, Television, Television Unit and Vases. We are going to collect more data and predict all objects of the home such as kitchen, bedroom, and bathroom at home in our future work. To start with this project, we collect our own data sets and store them in separate folders according to their names. Many approaches are applied to train and predict objects in living rooms by three different algorithms. Three different algorithms are implemented and compared according the accuracies. After taking the algorithm, which gives the biggest accuracy, we try to optimize and evaluate the accuracy. The rest of the papers as follows, We briefly review the related studies in the Section. 2. In the Section. 3 we describe our approaches. In the Section. 4, we mention about experimental results. And finally, We come up with conclusions and about the future direction. we explained it in the Section 5
.

2   Related Works

Recently, there have been several works about image classification or image recognition based on the convolutional neural networks. To begin with, MIT places [7] uses convolutional neural network but it is about image recognition. They use 205 classes and there are more than 2 million images. Also there is a work that is made using ImageNet [5]. In addition these, there are also well known works such as MNIST handwritten digits [1, 2] and "cats or dogs classifications". [4]
This paper is related to why we choose convolutional neural network and how we improve the accuracy. We have 10 different classes that are Armchair, Carpet, Coffee Table, Lamp,Painting, Sofa, Sofa Table, Television, Television Unit and Vases.

3   Approaches

3.1   Data Collection

Figure 1: Some objects in the train dataset

Dataset is collected from various resources and websites using some Python libraries like BeautifulSoup and manually as well. We collected average 400 - 450 images for all classes.

Collected images are not clean so to make them more clear we use some Python libraries that are PIL and OpenCV. All images are resized to the same size and put the separate folders according to their names. Both train and test sets have the same format. At the beginning of this project, we thought that if all images have the same extension, accuracy comes more so we change all the extensions to the jpg format but we see that this is not the case. There is no change in the accuracy. In conclusion, we have three distinct folders that are training, validation and test dataset. Both these folders have 10 distinct folders which are classes.

3.2 Choosing The Right Framework

In deep learning, there are different frameworks that we can use during this project. We were not familiar with any of them so we firstly made a deep research and at the end of this research, we have decided to use Keras.

Keras is an open source library in Python. It has a lot of features that we can be useful for us.

3.3 Deciding The Right Algorithm

For many years, we had some algorithms that can learn and classify images and analyze texts data such as K-nearest Neighbors Algorithm (K-NN) and Perceptron Algorithm but at that time, we did not have computational power and to process that large amount of data. Currently, we have smart algorithms such as Convolutional Neural Network and we have better computational tools to process more data in a short time. In our project, we have chosen three of the most common algorithms to classify the dataset and do this project. K-nearest Neighbors Algorithm (K-NN) is selected as a machine learning algorithm. In addition, Convolutional Neural Network(CNN) and Multilayer Perceptron are selected to make a multiclass classification as deep learning algorithms.

3.4 Convolutional Neural Network

Convolutional Neural Networks, its other name is ConvNet, is a special form of the general Neural Networks. Convolutional neural networks are used for many application especially for image recognition, image classification, natural language processing and other kind of works.
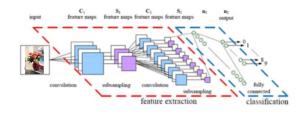

Figure 2: Convolutional Neural Network [7]

In convolutional Neural Network, there are several layers such as convolutional layer, pooling layer and fully connected layer. In addition these layers, there are some other kind of layers such as dropout layer. Classification is made in a fully connected layer and feature extraction is made in the other layers. The general structure of any convolutional neural network is shown in Figure 2 [6].

3.5 Multilayer Perceptron

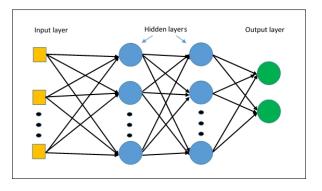Multilayer perceptron, is a deep learning algorithm that we use for classification.

Figure 3: Multilayer Perceptron [6]

It has several layers such as input layer, hidden layer/layers and output layers as shown in Fig-

ure 3. The basic idea is giving the data in the input layer and calculate in the forward direction. In the output layer, the correct outputs are known so weights are calculated using backpropagation algorithm.

3.6 K - Nearest Neighbor

K- Nearest Neighbor is one of the most used and simple machine learning algorithm. To classify new data, it compares the new data with the training examples and decides the distances between data. According to distances and k number, the model decide the class of the new data
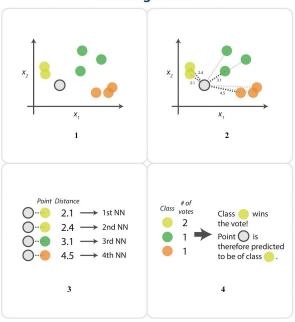
kNN Algorithm

Figure 4: K - Nearest Neighbor(k-NN)

The basic idea of k Nearest Neighbor algorithm is shown in Figure 4 [3]. The steps are like:

1) Learning how many classes do we have and which data we want to classify.

2) Calculate the distances or similarities between new data and the other data which we already know the which class they are belonging.

3) Find k nearest neighbors according the distances or similarities.

4) Predict the class of the new data.

4 Experimental Results

4.1 Comparison of The Algorithms

ALgorithm	Accuracy
Convolutional Neural Network	56.8
Multilayer Perceptron	56.43
K - Nearest Neighbor	No Result

Table 1: Comparison Table of The Accuracies

As we mentioned in Sec 3, we used 3 different algorithms to decide with which algorithm we should continue our way. We cannot get a result from k - nearest neighbor algorithm for the multiclass classification. Because the k - NN algorithms is already a slow algorithm and we had too much dataset for the k - NN algorithm so it takes too much time. Briefly, we cannot take a result according to the complexity problem.

Our multilayer perceptron is created using 4 layers. In the input layer, there are 768 nodes; in hidden layers, there are 522 and 351 nodes respectively; in output layer, there are 10 nodes. In addition these, we used sigmoid (see eq. 2) in the output layer and ReLU (see eq. 1) in other layers as activation functions; we used mean squared error (see eq. 3) as the loss function and we come up with 56.43 accuracy. This result is not good enough because multilayer perceptron does not deal with the dimensional information of the images. In addition this, we didn't make any feature extraction before running the algorithm.

$$R(z) = max(0, z) \qquad (1)$$

$$f(t) = \frac{1}{1 + e^t} \qquad (2)$$

3

$$MSE = \frac{1}{N}\Sigma(actual - predicted)^2 \qquad (3)$$

$$\sigma(z) = \frac{e^z}{\Sigma e^z} \qquad (4)$$

In convolutional Neural Network Algorithm, we had 5 convolutional layers and 5 pooing layers, 5 Batch Normalization layers, 3 dropout layers and 3 dense layers at the beginning. We used ReLU (see eq.1) and softmax (see eq.4) functions as our activation functions and we used mean squared error (see eq. 3) as loss function. In addition these, we used 'adam' as optimizer. We used 50 as batch size and 5 epochs. Although we had many layers in convolutional neural network algorithm, we just got 56.8 accuracy because our dataset was not enough and we didn't use any image processing skills. We only gave the images to the network. And also batch size, epoch size and learning rate may affect the result. Although this accuracy is not good, as we see in table 1 that it is still better than the others because convolutional neural network deals with dimensional information of the images and it makes feature extraction through the layers but multilayer perceptron does not. So we choose the convolutional neural network as our algorithm and the base accuracy is 56.8.

4.2 Limitations Of The Approach

There are some limitations that also affect the results. In k - nearest neighbor algorithm, we first need to determine the k value and the cost is so high because we should calculate the distances between all training data and the new data for all data in the test set so complexity is so high.

In multilayer perceptron, we should decide the optimum learning rate, epochs, and batch size. And to train the model, it uses back-propagation so it also affects the speed of the model. The more back-propagation the model made and the slower training the model. Additionally, multilayer perceptron does not deal with the dimensional information of the images and it does not have the convolutional layers and pooling layers, it cannot make the feature extraction.

Convolutional neural network requires a lot of data. If dataset is not enough for convolutional

neural network we face with over-fitting. Additionally, the more layers we have, the slower running the model and sometimes the computers cannot manage this kind of situations which we will explain the details that we face some problems in improvement part. (see 4.3) And convolutional neural networks are costly. To get a better result, we should decide the optimal parameters and hyper-parameters.

4.3 Improvement

As we said before, (see sec. 4.1) we continue our way using convolutional neural network with 56.8 accuracy as base accuracy. To improve this accuracy, we tried some cases:

Using Some Image Processing Techniques

Our first trial is using some image processing techniques. As we know in the structure of CNN, There are convolutional layers, 2 pooling layers, flatten and dense layers. (There are 128 and 10 nodes in the dense layers respectively in our model.) We use 4 as a set per epoch and we have 100 epochs. We use ReLU (see eq.1) and Sigmoid (see eq. 2) functions as an activation function and mean square error (see eq. 3) as our loss function.

When we run this structure, we get 61.69 accuracy.

	Accuracy
The base structure	56.8
First trial	61.69

Table 2: Comparison Table of The Accuracies

Although the more layer we used in the base convolutional neural network, we get more accuracy from the second structure (see table 2) because image processing affects the accuracy very much and we increase the epoch in the second structure. And because we have fewer layers in the second structure, the complexity is decreased. This new structure gives the result faster

.

Adding Dropout Layers And Changing Some Attributes

As our second trial, we add the dropout layers to the model. Normally, Adding dropout layers make the accuracy higher but our accuracy decrease. Actually to avoid over-fitting, adding dropout layers are important but because we may choose the dropout rate wrongly, the accuracy decreases.

Our other trials are changing some attributes like number of epoch, learning rate, activation and loss function etc. but we cannot get a better accuracy than 61.69. Our choices may be wrong.

Collecting More Images

Our another attempt is collecting more images. The convolutional neural network requires a lot of dataset to give more accuracy so to get more accuracy and also to avoid over-fitting, we collect more images especially complex images.

	Accuracy
The base structure	56.8
First trial	61.69
Collect more images	66.27

Table 3: Comparison Table of The Accuracies

To test how collecting more images affect the accuracy, we use the structure that is in section 4.3. As we see in table 3, the accuracy becomes 66.27 from 61.69.

Adding More Layers and Editing The Dataset

We add 1 more convolutional layer, 3 batch normalization layers and 3 dropout layers, 1 more max pooling layer and 1 more dense layer to the structure and we also change some parameters. In addition these, we edit our training-validation and test dataset. Before editing, we use the %30 of the dataset as test, %30 of the dataset as validation and the rest is train dataset. But after editing, we use the %15 of the dataset as test, %15 of the dataset as validation and the rest is train dataset.

	Accuracy
The base structure	56.8
First trial	61.69
Collect more images	66.27
More layers and editing dataset	69.15

Table 4: Comparison Table of The Accuracies

In this structure, we change the batch size to 128. As we see in table 4, the accuracy becomes 69.15.

Adding More Layers and Changing Attributes

One of our attempts is similar to the others. We add one more convolutional layer, one more pooling layer and one more dropout layer. But this time, we can find better batch sizes, epochs etc. For this model, we specify 25 for steps per epoch, 100 for epochs and 100 for validation steps. In addition these, we also add 'shuffle=false'. It affects because before we add it, the validation accuracy does not increase very well and the test accuracy comes between 60-65. But after we add it, the accuracy comes 75.28. In this model, we use 'adam' as an optimizer and mean squared error (see eq. 3).

	Accuracy
The base structure	56.8
First trial	61.69
Collect more images	66.27
More layers and editing dataset	69.15
More layers and changing attributes	75.28

Table 5: Comparison Table of The Accuracies

MIT Places Model

To get more accuracy, we try to use MIT places pretrained model [7]. Because the dataset of this model is more appropriate for our project. But it breaks our computers. After we try to force our computers to run this model, we cannot open our computers for a while.

Pretrained Models

As we mentioned in section 4.3, our computers cannot run MIT places pretrained model [7] so

we decided to try pretrained models whose weights are image-net. But the highest accuracy we can get with these models are 67.28. Although we changed attributes, parameters and hyper-parameters and also doing fine-tuning we cannot get a good result because the classes of image-net are so different from ours.

Figure 6: The loss graph of the model

5 Conclusions

We aim to make image classification between 10 top furnitures in a living room. After collecting our dataset, to decide which algorithm we should use during this project, we compare 3 popular algorithms that are convolutional neural network, multilayer perceptron and k - nearest neighbor. We just give the original images to select the best algorithm. The accuracy comes 56.8 from convolutional neural network, 56.43 from multilayer perceptron and we cannot get any result from k nearest neighbor because of the complexity. We get the highest accuracy from convolutional neural network so we decide to continue our way with convolutional neural network and take the 56.8 as our base accuracy. Now, we try to improve this base accuracy. When we look at the table 5 in section 4, the highest accuracy, that we can get, is 75.28. Because of we cannot run the MIT places pretrained model [7] and we cannot get a better result from imageNet pretrained models.
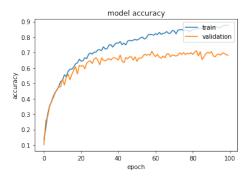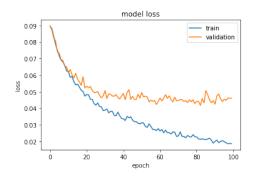
Figure 5 and 6 shows us the accuracy and the loss graphs of the model which gives us 75.28 accuracy.

Image	Prediction	
	Armchair	
	Painting	
	Television	

Figure 7: Test on 3 furnitures

Figure 5: The accuracy graph of the model

After train the model, we also want to see what model say about images. So we just give the 3 furnitures and compare the results. These furnitures are an armchair, a vase and a television. The model predicts armchair and television correctly but it predicts wrongly for the vase (see figure 7).
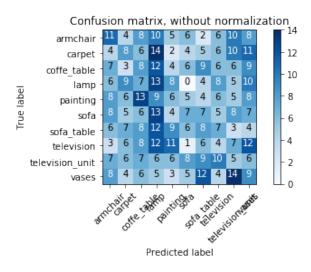
Figure 8: Confusion matrix

In figure 8, we see the confusion matrix. Confusion matrix show the performance of the model on test data. We generate this confusion matrix without normalization. As we see in figure 8, although the model makes prediction on all classes, it usually finds the correct class at the end.

Class: Painting   **Class: Vases**

Figure 9: Examples are taken from two distinct classes

We see that in figure 9, 2 different data are taken from two distinct classes. One of them is from painting class and the other is from vases class. Because of there are too many such kind of similarity between classes, the model makes prediction on all classes.

As we mentioned in section 1, we get inspired by home robots, smart homes and interactive homes with their owner. So we should work on more objects like sleeping room objects, kitchen objects, bathroom objects etc. In addition these, home robots, smart homes and interactive homes

should do any work perfectly that means we should increase the accuracy. To achieve this, we should use convolutional neural network models such as AlexNet, GoogleNet etc. Instead of our own model or we may work with a more powerful computer to run MIT places pretrained model [7] or other pretrained models. After achieve all of these, we can embed the algorithm into an circuit or a robot.

References

[1] Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, Lawrence D Jackel, Yann LeCun, Urs A Muller, Edward Sackinger, Patrice Simard, et al. Comparison of classifier methods: a case study in handwritten digit recognition. In *Pattern Recognition, 1994. Vol. 2-Conference B: Computer Vision & Image Processing., Proceedings of the 12th IAPR International. Conference on*, volume 2, pages 77–82. IEEE, 1994.

[2] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[3] L.V. on February. k-nearest neighbor algorithm using python. 2016.

[4] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3498–3505. IEEE, 2012.

[5] Olga Russakovsky, Jia Deng, Zhiheng Huang, Alexander C. Berg, and Li Fei-Fei. Detecting avocados to zucchinis: what have we done, and where are we going? In *International Conference on Computer Vision (ICCV)*, 2013.

[6] Giancarlo Zaccone. Multi layer perceptron. In *Getting Started with TensorFlow*, 2016.

[7] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.