AWS Machine Learning Nano Degree
Capstone Proposal

-Sancharika Debnath

Furniture classification

1. Domain background

Nowadays, object classification/recognition from photos is an appropriate job for Machine Learning techniques, particularly Deep Learning approaches [1].
The results obtained using Deep Learning for the problem of object categorization are amazing. These strategies are used in a variety of consumer products (e.g., Google's products) [2].
The growth of GPUs (Graphical Processing Units) has aided academics and business in the advancement of Deep Learning methods, allowing them to explore deeper and more sophisticated Neural Networks [1]. To accelerate the growth of Deep Learning models, even GPU suppliers are developing dedicated and powerful hardware [3].
The task at hand is to categorize various product categories based on photographs, in this case furniture. With the introduction of online marketplaces (such as Amazon, Walmart, and others), product classification has become a difficult task because several sellers selling the same or similar products provide the marketplaces with poor or limited information about the products they are selling, making it difficult for the marketplace to group products together and/or duplicate some records [4].
Using photos of the items might aid in this difficult work of classifying and possibly removing duplicate product information from those databases. Improving and smartening the consumer experience.

2. Problem statement

We'll use Deep Learning to categorize the furnishings in an image in this project. This project will be based on the dataset supplied for the iMaterialist Challenge (Furniture) at FGVC5 [5], a Kaggle Competition.
The task may be stated simply as follows: categorize the types of furniture included in a photograph. When the challenge is to categorize photographs of these types of items, which may possibly be utilized to classify thousands of images automatically, it's reasonable to conclude that we should employ a computer to accomplish this operation.
Even if we, as humans, are competent at picture categorization, doing so for thousands of photographs in a matter of minutes using people is almost impossible, or at the very least will require a large number of people to work in parallel.
As a result, automating this type of activity is critical for certain firms and may save a significant amount of money.
The method I suggest is to build a classification model that uses a pre-trained Convolutional Network to extract features from photos, as well as a final classification model that has been trained specifically for the task.

3. Datasets and inputs

Originally, the published dataset provided by the competition has more than 180000 images for training. Images are classified into 128 possible labels.
For the purpose of this project just part of the labels and images may be used, this is due to budget and computing power restrictions. Datasets of the *top-5* classes and images were prepared and will be used to build the classification

model. The datasets are balanced since we are selecting a limited amount of images and selecting the almost the same number of each class, reducing issues

when dealing with imbalanced datasets.
Three datasets were prepared (they just vary in amount of images), one small with 1000 images (200 images for each class) [~200MB], other with 2500images (500 each class) [~500MB] and the last one with 5000 images (1000 each class) [~900MB]. The images vary in dimensions, they are not standardized, and they have color layers. The images are 8-big sRGB.
Example of images of the dataset:

The original dataset (from the competition) is available as *json-format* file and inside it there are URLs to download the images, some of these images are not available anymore. The images were already downloaded of my part, so it will be provided (as part of this project) a zipped version of each dataset part as well as any other resource to run this project.
The dataset will be split into trained set, validation set and test set. The proportion of each will be defined later.

4. Solution statement

A Convolutional Neural Network is the best Deep Learning model for this assignment. Unfortunately, training such models from scratch is time and computationally expensive. Instead, this challenge will be solved with the help of a pre-trained model. In industry, using a pre-trained model to speed up the process is popular.
A pre-trained CNN model will be used to extract features from the photos, as well as a fine-tuned classification model (a Fully-Connected Neural Network).

5. Benchmark Model

One alternative benchmark is to compare the answer against vanilla models, which are solutions to the same issue taught from scratch (without any pre-trained models). The vanilla model will be used as a comparison to see how much better we can do with transfer learning from a pre-trained model.
The vanilla CNN design will be described in further depth in the Project Design section, but it will essentially consist of numerous Convolutional Layers and a Fully-Connected at the end.

6. Evaluation Metrics

Because the dataset is balanced, accuracy will be used as the final criterion to assess the outcomes. The pre-trained model used will be determined by resource constraints and the quality of outcomes obtained using this metric. Candidates for pre-trained models are given in the Project Design.

7. Project Design

This section describe a bit about how the vanilla model will be build and trained, as well as the classification model with a pre-trained CNN. At the end is discussed about fine-tuning.

7.1. Vanilla model

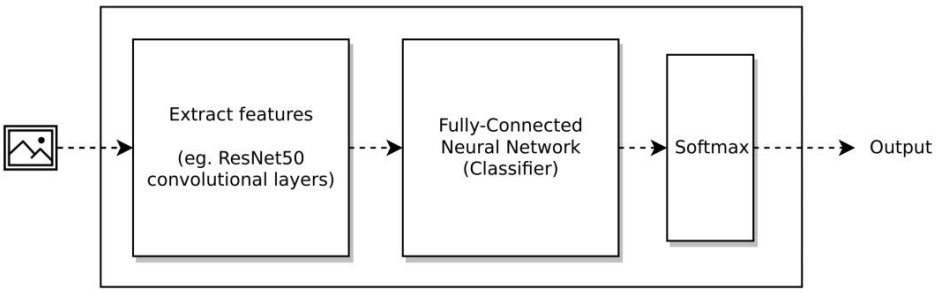
For the vanilla model, the images will re-sized to 244x244 and pixels, normalized. To fair comparison between the models, both of them will use the same dataset, without any modification.
The planned vanilla model to classify images will be designed with a sequence of **Convolutional Layers** with **Max Pooling** and **Dropout** (if need, as regularizer) in between. The final classification layer can be a typical **Fully-connected**. This kind of architecture I've worked with in the past and usually performs reasonably well on this kind of task.
The following Keras model summary shows an example of this architecture and, of course, the version that will be used might differ of this shown here.

_____
Layer (type) Output Shape Param #
==========
conv2d_21 (Conv2D) (None, 222, 222, 32) 896
_____
max_pooling2d_21 (MaxPooling (None, 222, 222, 32) 0
_____
dropout_5 (Dropout) (None, 222, 222, 32)
0_____
conv2d_22 (Conv2D) (None, 220, 220, 64) 18496
_____
max_pooling2d_22 (MaxPooling (None, 110, 110, 64) 0
_____
dropout_6 (Dropout) (None, 110, 110, 64) 0
_____
conv2d_23 (Conv2D) (None, 108, 108, 128) 73856
_____
max_pooling2d_23 (MaxPooling (None, 108, 108, 128) 0
_____
dropout_7 (Dropout) (None, 108, 108, 128) 0
_____
global_average_pooling2d_8 ((None, 128) 0
_____
dense_9 (Dense) (None, 5) 645
==========
Total params: 93,893.0
Trainable params: 93,893.0
Non-trainable params: 0.0
In order to improve
the
quality of the model, data augmentation can be
considered. Although

it is almost sure that a vanilla model trained with few resources is certain that will perform worse that a model that uses a pre-trained ConvNet, exploring its potential can be rewarding. Summing up, the vanilla model will receive a tensor of shape (batch_size, 244, 244, 3) and will output a vector of shape 5 which the probabilities of each class based on an given image. As an activation of output layer, the Softmax will be used.

7.2. Transfer learning

In this scenario, I am going to use a pre-trained model to extract the bottleneck features from the images. The images will be resized to 244x244 and bottleneck features extracted and saved to disk (caching purposes). At this moment there is no plan to, for example, crop or apply any transformation in images. But, during the workflow, we can include this step to improve the results.
The candidates as pre-trained model would be *Resnet50* and *InceptionV3*, they are available in Keras and ready to use. To speed up the process the bottleneck features extracted will be saved to the disk and will be available for download later to the reviewers.
The classification model will be created and prepared to consume as an input the features extracted using the pre-trained CNN. A **fully-connected** layer with Softmax as activation function of the output layer. A **Dropout** layer can be used to help in generalization as well as to reduce overfitting problems.
Once having a model trained, a typical and simplified final structure of a algorithm to perform the classification is shown below:



7.3. Fine-tuning

For both cases, where applicable, a fine-tuning on training and in the models to improve the quality of the results will be made. For example, switching the optimizer and their parameters, like switching from **Adam** to **SGD** or **RMSprop**.
Adjusting parameters like **learning rate**, **batch size** and **momentum.** Or the parameters available for each optimizer that might lead to optimization. An alternative to fine tune the parameters is by applying a grid search of something similar to explore the parameters automatically.
One useful tool to explore the training and results is the **TensorBoard**. This tool can be used to help to understand the training procedure and potential issues and optimization opportunities.

References

[1] Krizhevsky, Alex, Ilya Sutskever, and Geoffffrey E. Hinton. "Imagenet classifification with deep convolutional neural networks." *A dvances in neural information processing systems*. 2012.

[2] Abadi, Martín, et al. "Tensorflflow: a system for large-scale machine learning." *OSDI*. Vol. 16. 2016.
Jianmin C., Zhifeng C., Andy D. et al.
[3] https://developer.nvidia.com/deep-learning
[4] Fensel, Dieter, et al. "Product data integration in B2B e-commerce." IEEE Intelligent Systems 16.4 (2001): 54-59.
[5] https://www.kaggle.com/c/imaterialist-challenge-furniture-2018