Capstone Project Report
Furniture classification using images and Deep Learning

SANCHARIKA DEBNATH
sancharikadebnath@gmail.com

1. Definition

1.1 Project Overview

Automated product classification has become a regular and difficult issue for large marketplace sites (Fensel et al., 2001). One of their responsibilities is to aggregate dozens of products based on information provided by multiple vendors, which might be deceptive due to different languages, inadequate data, and so on. Images may be utilized to categorize these items in addition to the verbal meta-data given by the sellers, boosting the quality of product classification and matching.
With the introduction of Deep Learning algorithms and strong hardware (Krizhevsky et al., 2012; Abadi et al., 2016), image recognition has become more powerful, making a costly work more affordable. This project tries to tackle the challenge of product classification by utilizing photographs to solve at least a portion of the problem.In this scenario, the goal is to classify a limited number of product subcategories inside a larger core category, furniture.

1.2 Problem statement

The purpose of this project is to develop a product category recognizer using photos of furniture, as previously stated. The solution must produce the most likely category of the item depicted in the picture given an image of a product.
To overcome the problem, researchers used an annotated dataset of these products to train a classifier with Deep Learning algorithms. The following are the duties that are involved in general:
• Split images into train, test, and validation sets by re-sizing, normalization, and/or extracting features.

• When given an image, train the classifier to output the probabilities of each category.
• Prepare a basic algorithm that employs the classifier to output the class of the product existing on an unseen image.
The final model should output the product's class with an acceptable degree of certainty.

1.3 Metrics

Some criteria must be utilized to assess the model's quality and predictability of the response it will produce. Because the dataset is balanced, accuracy will be employed as a measure.
After training, the **accuracy** will be assessed as follows:

$$acc(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} 1(\hat{y}_i = y_i)$$

2. Analysis

2.1 Data exploration

The dataset used in the iMaterialist Challenge (Furniture) at FGVC5 (Kaggle, 2018), a Kaggle Competition, was released. Originally, the competition's public dataset included over 180000 photos for training. There are 128 different classes in which images can be classified.
Due to funding and computational power constraints, just a portion of the classes and pictures were used for this project. Images and data sets for the top five classes were created. Three datasets of increasing sizes were created: one with 1000 photographs, another with 2500 images, and the third with 5000 images. The 5000 pictures dataset was used to train the models presented here.

2.2 Exploratory visualization

The 5 classes present in the dataset are: Mug, Cup (of Glass), Pillow, Desk and Table lamp.
Figure 1 shows examples of pictures of each of these classes.

(a) Mugs

(b) Cups

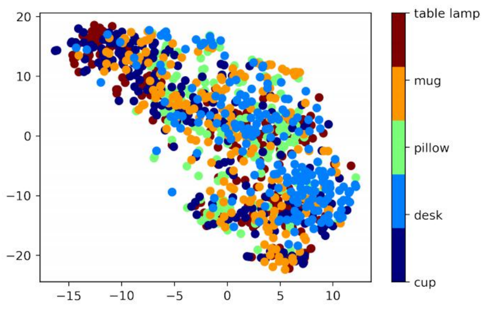
(b) Pillows

(d) Desks

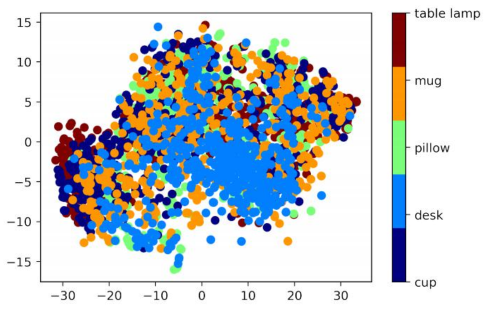
(e) Table lamps

Figure 1: Example of pictures of each class

The t-SNE clustering map is shown in Figure 2. A t-SNE (perplexity = 20) was used to examine whether there was any discernible pattern in the data. Unfortunately, the narrative didn't reveal much owing to the intricacy of the graphics. There are certain concentrated regions, such as table lights and desks, but t-SNE was unable to distinguish between the other classes.



(a) t-SNE of 1000 images



(b) t-SNE of 2500 images

Figure 2: t-SNE plots (perplexity = 20)

All datasets in this project are fully balanced; they have been pre-processed and prepared such that each class has the same number of photos (see the README for download links). Images are scaled to 224x224 with three channels throughout the model's training.

2.3 Algorithms and Techniques

This research employed a Convolutional Neural Network (CNN), which is a cutting-edge architecture for solving problems like pattern recognition in photos. CNN is a kind of Neural Network that is typically used to solve visual imaging difficulties. CNN excels at this task in part because of its spatial invariant ability, which allows it to detect patterns that may be translated, moved, or distorted.
This spatial invariant ability, which enables CNN to perform so well for pictures, can be attributed to the convolution layer's capacity to recognize high-level features of an image, such as edges, curves, straight edges, and colour.To do this, the convolution layer employs filters (also known as convolution kernels), which slide across the whole picture from top to bottom, delivering the input to the activation layer and resulting in a 2-dimensional activation map.
One of the methodologies used in this research is called Transfer Learning, which is approximately equivalent to taking a portion of a previously learned (weights) Neural Network and utilizing it in a new one while training the remainder for a new task. The Convolutional layers were used as a *feature extractor* to precisely identify the high-level features of an image, such as edges, curves, straight edges, and colour, which were already described about CNNs.
The Transfer Learning technique has the advantage of re-purposing a prior and costly effort that was utilized to create these Convolutional layers. They are costly to construct from the ground up.
Even on strong and specialized hardware, the best known CNNs taught to detect a wide range of objects from photos take several hours, days, or even weeks to train.
So, the model developed for this project that employs the Transfer Learning technique was far less expensive to train, yet it still works admirably if it is fed a decent pre-trained model.

2.4 Benchmark

The goal for the final solution is to achieve an accuracy of between 80 and 90 percent.
A model was constructed and trained from the ground up to serve as a standard and a baseline (including the Convolutional Layers). The standard model. The hypothesis is that the feature extractor model that employs another pre-trained model outperforms the vanilla model.
For a fair comparison, the same dataset was utilized in both situations.

3. Methodology

3.1 Data preprocessing

Images were downsized to 224x224 (needed by the pre-trained model) for the Transfer Learning methodology, and the *pre-processed input* method (of Keras and TensorFlow) was used to adjust the input to the shape required by the pre-trained model. Images were also scaled to 224x224, input normalized, and turned into tensors for the benchmark model (Vanilla).
There was no data augmentation approach used. Using half of the dataset (2500 photos) instead of the complete dataset (5000 images) yielded identical results during the testing, demonstrating that augmentation wasn't necessarily essential to get the final model to meet the initial expectations. The vanilla model may benefit from the method, but as long as it is only the baseline model, no effort was expended to improve it for the time being.

3.2 Implementation

As previously stated, two types of models were created for this project. The implementation details as well as the architecture of each one are listed below. Some *utility* classes were used in both situations to make training and testing operations easier and to compare the models more readily. The TRAINER, TRAINEDMODEL (encapsulates the trained model and provides various ways to interact with it), TRAINPLOTTER and TRAINTESTDATASET.
can all be found in the source code (utilities module).

3.2.1 Vanilla model, the benchmark model

The vanilla model, which is a fairly standard Convolutional Neural Network architecture without any usage of a pre-trained model or any other advanced component, is one model created to handle the challenge. It includes 2D-Convolutional layers with *ReLU* as activation functions, *MAXPOOLING* in the middle to minimize resolution and complexity, and a *GLOBALAVERAGEPOOLING* layer at the end that also flattens the 2D structure into 1D input. As a classifier, a fully connected structure is used, with the output layer values squished using *Softmax*. The figure 3 depicts the architecture of the building.

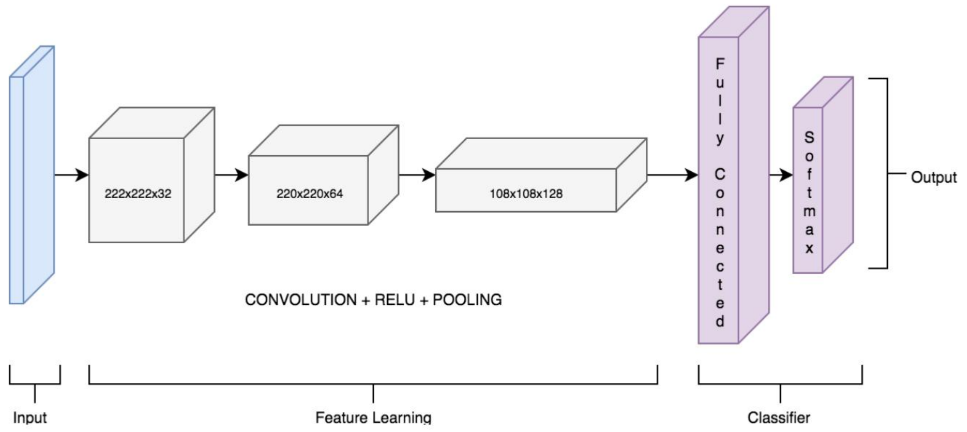


Figure 3: Vanilla basic architecture

One of the challenges of starting from scratch was to efficiently develop and train the model without wasting too much computing resources, manage training time, and make excellent use of the train, test, and validation sets in order to maintain a sustainable training method and avoid overfitting. After a series of changes and tests, the *Adam* was chosen as the winner.
The model's size and depth were managed to keep memory use below this threshold while also keeping the model sophisticated enough to extract patterns from data and perform well on its goal. Originally, the goal of this study was to employ a pre-trained model because of the historical and impressive outcomes obtained with this

method in similar problems. The solution based on the pre-trained model is compared to the vanilla model.

3.2.2 Transfer learning-based model

The project's core model was built on transfer learning techniques, which include leveraging a portion of a pre-trained model to get high-quality results faster and with less computer resources. The ResNet50 (Microsoft, 2018) was used as the pre-trained model. One of the reasons I chose ResNet50 was because I had prior personal experience with it, and since despite its complexity, the model does not demand a lot of computer resources to provide decent results. The weights of the ResNet50 model from the *ImageNet* challenge were used as a pre-trained model. The final design (excluding the internal architecture of the ResNet50) is rather straightforward as a result of using a pre-trained model. Another excellent argument to employ pre-trained models appears to be that the resultant architecture of the trainable section of the overall model is simple enough to train quickly. The resulting architecture is seen in Figure 4.

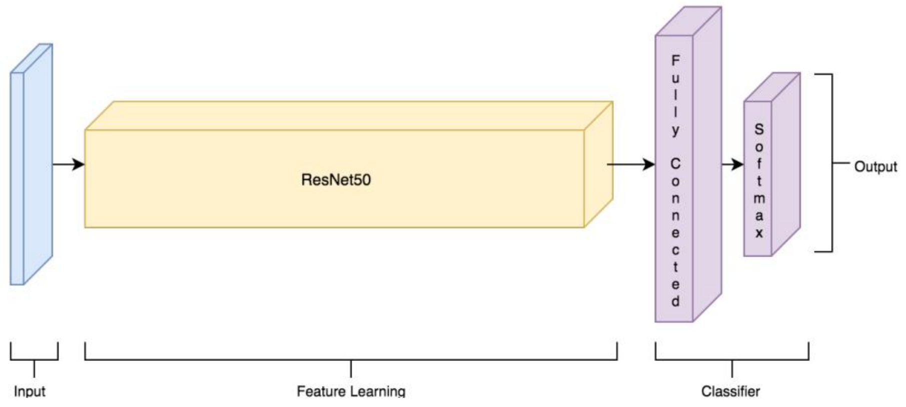


Figure 4: Transfer-learning based basic architecture

The model is essentially part ResNet50 (Convolutional Layers) with the final classification model modified to accomplish the project's goal, as indicated in the diagram. The ResNet50 extracts features for the trainable model as an input. The trainable model consists mostly of Fully-Connected layers with *Dropout* and *Softmax* as output layer activation.
Preparing the dataset was one of the first challenges in building this model. The only element of the model that can be trained is the

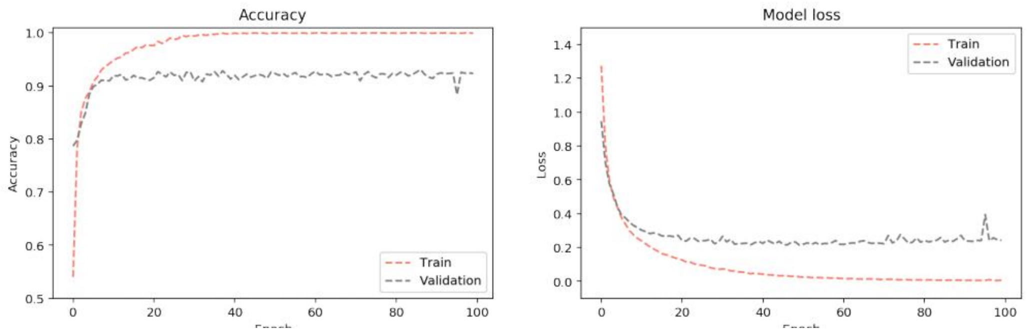
fully-connected layers (as mentioned earlier). This implies that the characteristics of pictures in the collection must first be extracted; these features are frequently referred to as *bottleneck features*. As a result, creating methods to extract these characteristics was part of the code, and to speed up the process later, the bottleneck aspects of the pictures were stored to the disc for caching reasons.
The create *bottleneck features* technique coordinates the extraction of bottleneck characteristics as *generated_bottleneck_features* method.
Images are imported and shrunk to 224x224x3, features are extracted, and the result is moulded into 2048-size vectors, allowing it to be fed into a fully-connected model that can't directly interact with multi-dimensional input.
Following the completion of the bottleneck extraction, the training method was created in the same manner as the vanilla model, utilizing common utilities classes such as TRAINER, TRAINEDMODEL, and so on.

3.3 Refinement

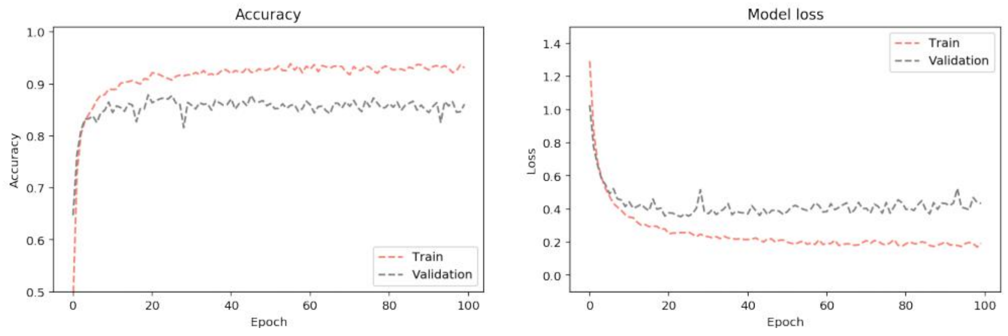
An enormous overfitting problem was discovered during the training of the model using ResNet50. Initially, the final classifier consisted of of fully connected layers, using the ResNet50 features as input. A *Dropout* layer was placed between two dense layers of the classifier and fine adjusted in an attempt to close the gap or make the model more general.
Adding a *Dropout* lowered the training and validation accuracy slightly, but it narrowed the distance between the two curves, indicating a robust model. Because *Dropout* refers to disregarding certain neuron's activation during training, it had an impact on overall accuracy as long as the number of epochs was kept constant. This does not necessarily imply that the model is bad; it may just require more training epochs.
Because our goal is to create a general and robust model for classifying furniture from photographs on the internet or in product catalogue, it must be broad and durable.

to make the input relevant by changing it at random Other refinements will be welcomed in the future as upgrades to narrow the gap even more and increase overall quality.

Figure 5 depicts the train history after 100 epochs using the model without a Dropout layer, whereas Figure 5 depicts the identical version with the Dropout layer included.



(a) Without dropout after 100 epochs



(b) With dropout (0.3 of drop probability) after 100 epochs

Figure 5: Comparing the addition of a dropout layer

4. Results

4.1 Model Evaluation and Validation

In this section both models are evaluated and their results discussed.

4.1.1 Vanilla model, the benchmark model

After 100 epochs with a batch size of 16, the vanilla model's final version had an accuracy close to 70% on the test set (10% of the dataset). 3250 photos were used exclusively for training, 501 for testing as unseen images, and 1249 for validation out of a total of 5000. The training/validation accuracy and loss over epochs are shown in Figure 6.

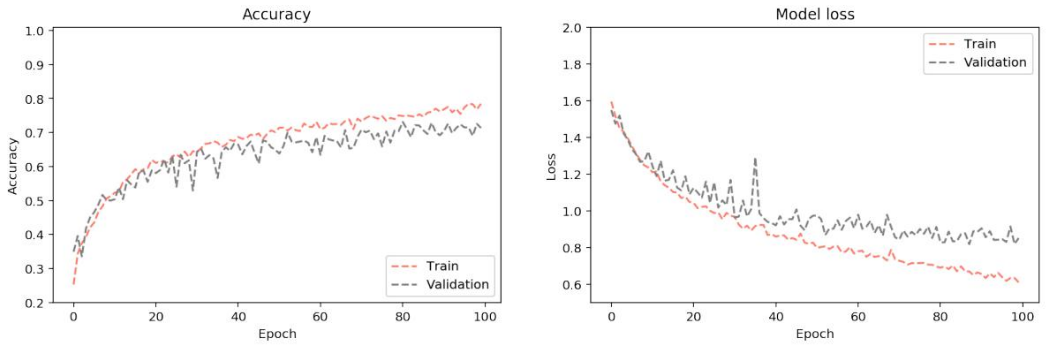


Figure 6: Vanilla model training/validation accuracy and loss curves

4.1.2 Transfer learning-based model

In practically all tests, the final version of the primary model, the transfer-learning-based, demonstrated an accuracy of roughly 83 percent on the test set, almost 13 percentage points higher than the benchmark model. One problem discovered was a minor overfit. A dropout layer was introduced as a regularizer to decrease and regulate the overfitting. The training/validation accuracy and loss over epochs are shown in Figure 7.

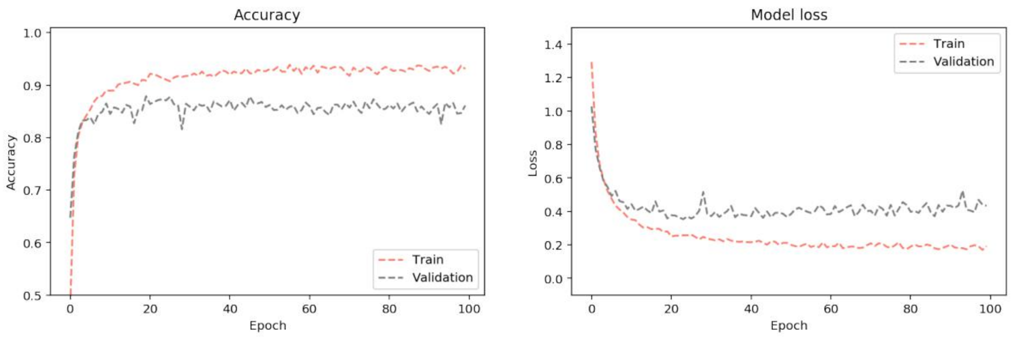
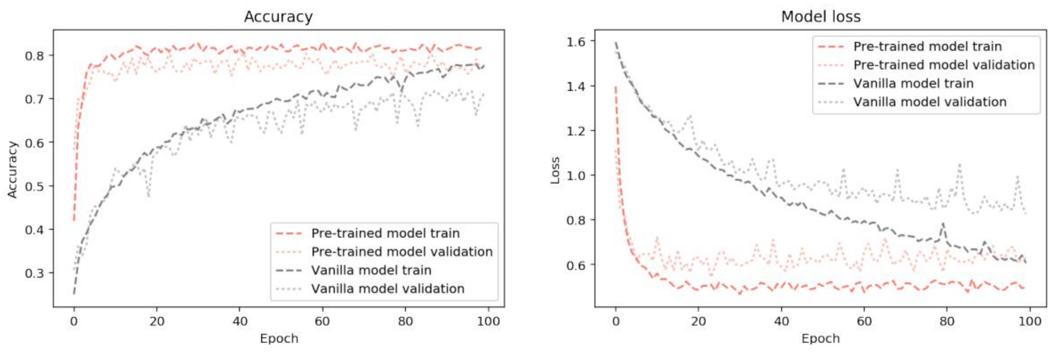


Figure 7: Transfer learning (ResNet50) training/validation accuracy and loss curves

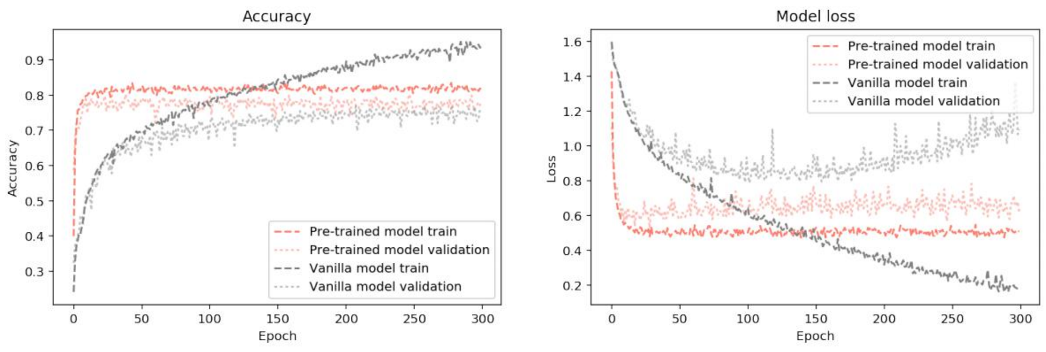
4.1.3 Comparing models

The training curves for both models are shown in Figure 8. Both of them were trained using the same dataset and 100 epochs at first. The pre-trained model achieves its optimum performance in a few

epochs and then remains steady without significant progress thereafter.
The gap is reduced later in the vanilla model.



(a) After 100 epochs



(b) After 300 epochs

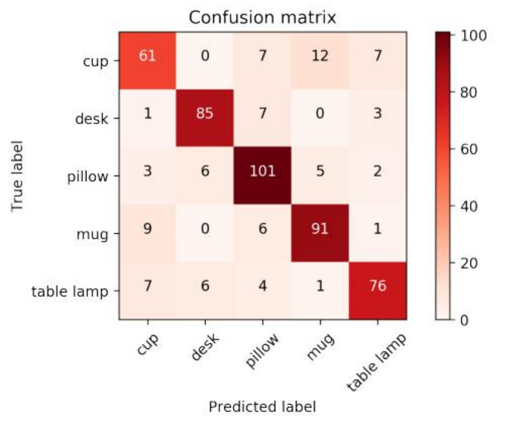
Figure 8: Comparing model’s performance

Giving the vanilla model extra epochs in an attempt to surpass the transfer learning based model did not completely succeed. In fact, it increased training metrics, but validation metrics did not improve at the same pace; as seen in Figure 8, validation loss began to worsen after 150 epochs. The vanilla model was struck by the overfitting problem simply by increasing the number of epochs. This might indicate that the model isn't generalizable and is too complicated for the situation.

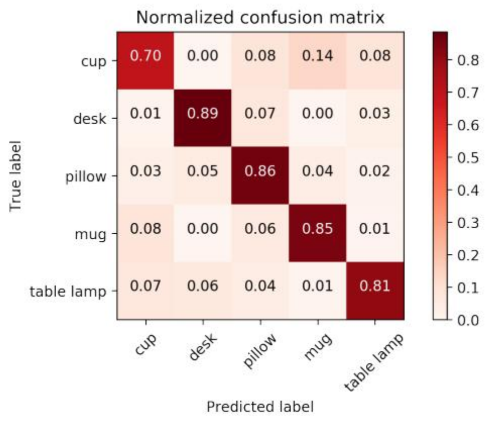
4.2 Justification

The final version of the primary model, which was based on transfer learning, had an overall accuracy of close to 83 percent, which was roughly 13 percentage points higher than the benchmark model. The benefits of this model are that it takes significantly less time and resources to train because it uses a pre-trained model.

It met at least one expectation, which was to outperform the standard model in terms of both quality and resources used (time and computational).
Cups and *Mugs*, as seen in the confusion matrix in 9, are the classes where the model makes the most errors; this seems plausible given how similar the items' forms are.
The biggest difference between them, at least according to our data, is that the *Cup* is mostly comprised of clear glass.



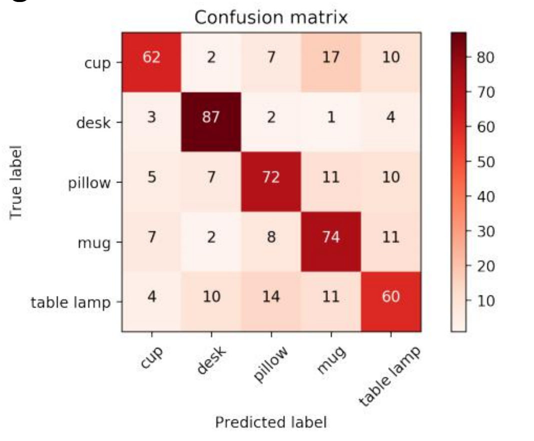
(a) Non-normalized



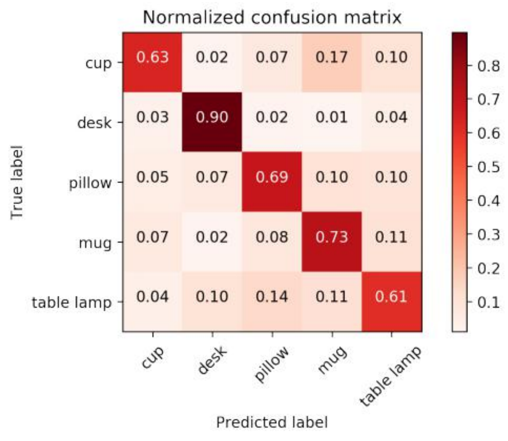
(b) Normalized

Figure 9: Confusion matrix of test set on transfer-learning based model

It is reasonable to conclude that the transfer learning based model outperformed the Vanilla model by comparing the results using the confusion matrix of the transfer learning based model displayed in Figure 9 and the confusion matrix of the Vanilla model shown in Figure 10.



(a) Non-normalized



(b) Normalized

Figure 10: Confusion matrix of test set on Vanilla model

5. Conclusion

5.1 Free-Form Visualization

The final model, which used transfer learning techniques to detect furniture in photographs from the internet, was able to do so. As seen in Figure 11, it successfully classified the products.

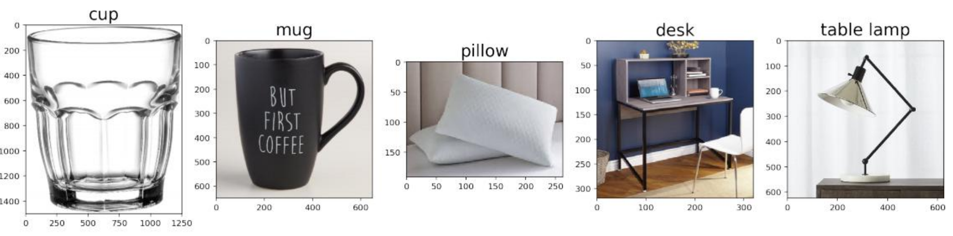


Figure 11: Classified images from internet using ResNet50-based model

5.2 Reflection

The entire process of building this end-to-end solution was:
• Find an interesting problem to solve and publicly available data.
• Prepare the dataset.
• Explore the dataset and check it up
• Create and train and benchmark model.
• Create and train the final model itself.
• Prepare to use the trained models to classify unseen images from internet.

The most difficult component was creating the models and doing adequate training. Initially, this was due to the large amount of resources required, as well as the fact that the model was often too sophisticated or too basic to discover patterns in data. Training, regulating the overfit, and convergence were equally difficult at initially, and it is a never-ending process. However, once a model begins to converge, boosting accuracy and minimizing loss, it is only a question of time and fine tuning to improve it. Of course, you may always make improvements to the model. Because there isn't a clear end, you'll have to find out when it's good enough for your needs.

5.3 Improvement

There are a few things that might be done to better the project as a whole. One, in my opinion, is to properly employ Keras generators in order to decrease memory utilization and therefore allow for the inclusion of more photos. Unfortunately, updating Keras to adopt the new API resulted in several unanticipated changes, as well as the appearance of faults. It's difficult to live on the ledge. To prevent sabotaging the entire effort, a downgrade was required.
Another possible improvement is to investigate the vanilla model further by using Data Augmentation and/or sophisticated image prepossessing to improve model result quality. Although the Vanilla-version wasn't the main model and only served as a benchmark, the results of the Vanilla model were interesting and promising, at least for me. Putting extra effort into it can pay off.

References

1. Mart´ın Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffffrey Irving, Michael Isard, et al. TensorFlow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

2. Dieter Fensel, Ying Ding, Borys Omelayenko, Ellen Schulten, Guy Botquin, Mike Brown, and Alan Flett. Product data integration in B2B e-commerce. *IEEE Intelligent Systems*, 16(4):54–59, 2001.

3. Kaggle. iMaterialist Challenge (Furniture) at FGVC5, 2018. URL `https://www.kaggle. com/c/imaterialist-challenge-furniture-2018`.

4. Alex Krizhevsky, Ilya Sutskever, and Geoffffrey E Hinton. ImageNet classification with deep Convolutional neural networks.

5. *Advances in neural information processing systems*, pages 1097–1105, 2012.

6. Microsoft. Deep Residual Networks, 2018. URL https://github.com/KaimingHe/ deep-residual-networks.