Classifying food items by image using Convolutional Neural Networks

Derek Farren
Stanford University
dfarren@stanford.edu

Abstract

Grocery items image classification is a well researched problem. However, until the recent announcements from Amazon regarding their "Just walk out" technology used in Amazon Go, most Computer Vision techniques used in the state of the art research did not involve neural networks. However, recently a research group from the University of Freiburg released the most complete grocery items image dataset openly available. They also developed the most accurate classification model for that dataset using convolutional neural networks. I this research I propose a model to classify the The Freiburg Groceries Dataset that is more accurate than the state of the art.

1. Introduction

Amazon Go's recent announcement has brought attention to grocery image detection in Computer Vision. The shopping experience, according to Amazon, is made possible by the same types of technologies used in self-driving cars. That is, computer vision, sensor fusion, and deep learning technologies. With "Just Walk Out" technology, users can enter the store with the Amazon Go app, shop for products, and walk out of the store without lines or checkout. The technology automatically detects when products are taken or returned to shelves and keeps track of them in a virtual cart. When the shopping is finished, users leave the store and their Amazon account is charged shortly thereafter.

At the heart of this technology, there is a Computer Vision model classifying grocery items by their image caught in a video camera. This work proposes a model to accomplish such a task.The model proposed is more accurate than the state of the art [1].

This work also proposes a greedy algorithm that improves the network performance by changing its architecture. This algorithm is called Guided_Prunning and it proved to be very helpful in situations where there is some large areas of convexity in the network architecture vs. network accuracy function.

2. Related Work

A fair amount of work has been done using Computer Vision on groceries datasets.

A real-time product detection system from video is presented in [2]. Some effort for matching database images on an input image is shown in [3] by using scale-invariant feature transform (SIFT)[4] vectors in an efficient manner. Another study focuses on logo detection in natural scenes by spatial pyramid mining [5].In [6], the authors apply planogram extraction based on image processing by using a combination of several detectors. SIFT matching and optical character recognition are some of them.

However, because most of the grocery image datsets are privately owned, not much improvement has been done in this area until last year soon after Amazon announced their Amazon Go stores.

A new dataset was released. The Freiburg Groceries Dataset [1] is a dataset consisting of 5,000 256x256 RGB images covering 25 different classes of groceries, with at least 97 images per class. The authors collected all images from real-world settings at different stores and apartments. In contrast to existing groceries datasets, this dataset includes a large variety of perspectives, lighting conditions, and degrees of clutter. Overall, the images contain thousands of different object instances. Examples for each class can be seen in Figure 2. This dataset is currently the state of the art used in grocery Computer Vision testing.

The authors also proposed a classifier on this dataset, where they re-trained the CaffeNet architecture and achieved a mean accuracy of 78.9%.

Also, this work proposed Guided_Prunning, a greedy algorithm that improves the network performance by manipulating some Hyperparameters. Hyperparameter optimization is an important research topic in machine learning, and is widely used in practice [9] [10] [11] [12]

. Despite their success, these methods are still limited in that they only search models from a fixed-length space. In other words, it is difficult to ask them to generate a variable-length configuration that specifies the structure and connectivity of a network. In practice, these methods often work

Figure 1: Example of images in the dataset.

better if they are supplied with a good initial model [10] [11] [12]. There are Bayesian optimization methods that allow to search non fixed length architectures [14] [15], but they are not very flexible. The best available method is a Reinforced Learning approach to hyperparameter tuning developed by Google [8].

3. Method

The proposed model is simpler than the CaffeNet model used by the state of the art, and achieves an accuracy of 89.12%, which is an almost 11 percentage point improvement from the state of the art.

After many failed attempts to tune a CNN that performed better than the state of the art I decided to implement the state of the art and prune or add layers and modify parameters until I make it perform better. I realized that the original model was overfitting, so I could probably increase its performance.

Tests were taking too long on a one GPU machine so I created an algorithm for a more guided and efficient testing. That algorithm is called Guided_Prunning and is shown in Algorithm 1.

The Guided_Prunning algorithm is a greedy algorithm that improves the model's accuracy step by step. It takes an array with all the parameters of the staring model and returns a new array with parameters that make that starting model perform better. I used the following parameters:

1. The filter size of each of the convolutional layers.
2. The number of filters of each of the convolutional layers.
3. The number of convolutional layers.
4. The size of each of the fully connected layers.
5. The number of fully connected layers.

On each iteration, the model adds δ to a parameter. This δ is a function of the parameter itself. For number of layers and filter sizes, δ randomly return a value $\in \{1, -1\}$. For fully connected layers size, δ randomly returns a value $\in \{512, -512\}$

I did not tune the hyperparameters with this algorithm because the space is much larger for these parameters and thus the algorithm would take too long.

4. Experiments

Initially, since the state of the art was using CaffeNet, I implemented that as an initial baseline. Then, I implemented other popular CV models, namely Alexnet and Googlenet. However, I wasn't able to get a better performance from these models. Probably one of the reasons is that tuning them required long hours of training with the limited GPUs I had.

Because of that, I picked the best model I had (CaffeNet) and started tuning it with Guided_Pruning (shown in Algorithm 1). This approach allowed me to improve the original CaffeNet's performance considerably.

4.1. CaffeNet: The original model presented in [1]

Caffe[7] was developed by the Berkeley Vision and Learning Center and community contributors. Caffe is easily customizable through configuration files, easily extendible with new layer types, and provides a very fast ConvNet implementation (leveraging GPUs, if present). It comprises 5 convolutional layers, each followed by a pooling layer, and 3 fully-connected layers. I customized the original CaffeNet so that the model takes input image size (256,

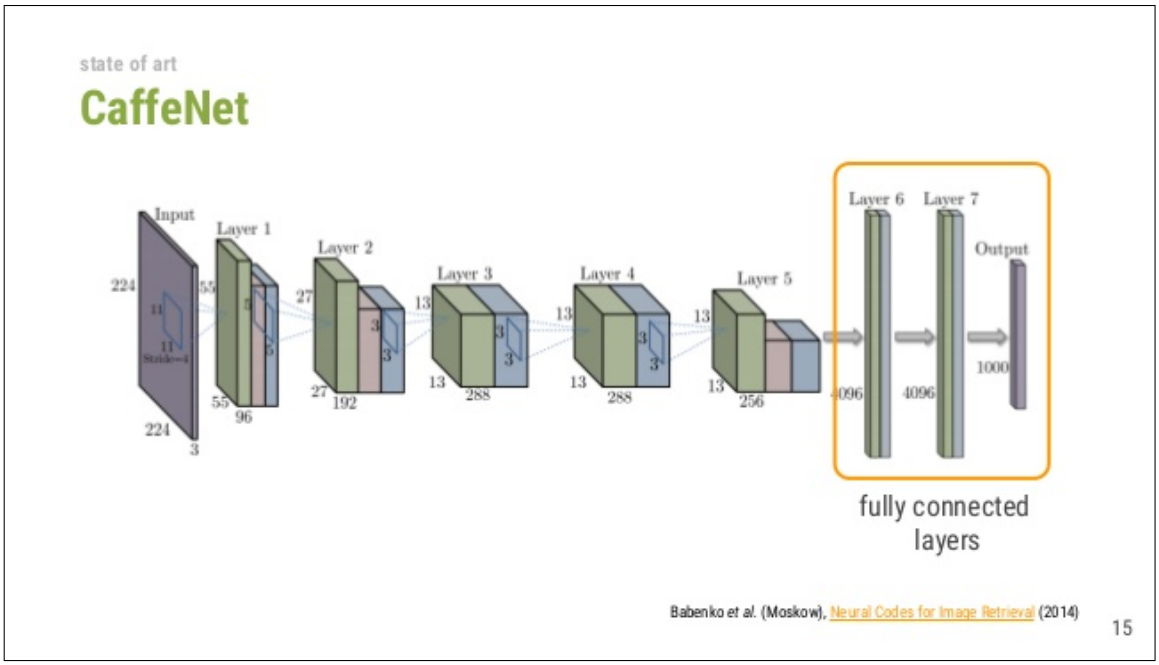

Figure 2: CaffeNet model used in [1]

256, 3) and the final fully connected layer instead outputs 25 scores - one for each class.

This model was the state of the art and had an accuracy of 78.9% on the Freiburg dataset in [1]. I was able to tune the model up to an accuracy of 77.03

4.2. Alexnet

I implemented at a modified version of AlexNet, a CNN architecture that uses ReLU non linearities. The architecture is as follows:convrelupoolnorm2convrelu3poolfcreludropout2fc, followed by a softmax loss function. The original model takes inputs of size (227,227,3) and outputs scores for 1000 different classes (for the ImageNet challenge). I instead modified it so that the model takes input images size (256, 256, 3) and the final fully connected layer instead outputs 25 scores - one for each class.

After tunning, this model had an accuracy of 73.2%.

4.3. GoogleNet

GoogLeNet is a 22-layer CNN, containing Inception Modules. It uses convolutions, max-pooling layers, ReLU non-linearities, and the softmax loss function. Each inception module consists of multiple convolutions (with different filter sizes) and max-pools that are concatenated together. The original GoogleNet takes inputs of size(224,224,3) and outputs scores for the 1000 ImageNet classes. I modified the model to have an input size(256, 256,3) and a final FC layers to output scores for 25 classes, as I did with AlexNet.

This model achieved an accuracy of 75.64%. It is worth mentioning that training this model takes about 4 hours using one GPU.

4.4. Guided_Prunning

Running the Guided_Prunning algorithm on 100 iterations produced the results shown in figure 3.

The model's architecture after Guided_Prunning ran was the following:

Input: 256x256x3 image

1. 11x11x3 CNN layer + Max Pool + Batch Norm
2. 5x5x16 CNN layer + Max Pool + Batch Norm
3. 3x3x96 CNN layer + Max Pool
4. 2048 Fully connected layer
5. 2048 Fully connected layer

Output: 25 logits

All activation functions are ReLu, i.e. the element-wise maximum between 0 and the input x.The use of this function allows a deep network to be trained more quickly, as its gradient is non-saturating. The discovery of this activation function was a key in deepening convolutional neural networks.

Algorithm 1: Guided_Prunning

Data:
`parameters` = array with all parameters in the model;
`model(parameters)` = `CaffeNet`;
`best_accuracy` = 0;
`best_parameters` = `parameters`;
Result:
`best_parameters`
while `current_accuracy` $\leq$ `goal_accuracy` **do**
  `param_index` = take random index from parameters;
  `parameters[param_index]` += δ(`param_index`);
  `accuracy` = `model(parameters).accuracy`;
  if `accuracy` > `current_accuracy` **then**
    `bast_accuracy` = `accuracy`;
    `best_parameters` = `parameters`;
  end
end

$$ReLu(x) = max(x, 0)$$

Since the original model was overfitting, I used a 50% drop-out rate on every ReLu layer. The drop-out layers serve as regularizers in the learning process, preventing overfitting. I tried using an L_2 regularization but the results were worst. The final output is a probability prediction for each of the 25 classes using Softmax. The Softmax layer output is:

$$\sigma(\sum_i w_i x_i + b)$$

This gives, for each class i, $P(y_i = 1; x_i, w)$. For each sample x, the class i with the maximum Softmax output is the predicted class for sample x.

Batch normalization is a means of dramatically reducing training convergence time and improving accuracy [16]. The CNN found by Guided_Prunning, had a batch normalization layer after the first two convolutional-max pool layers. These layers really helped not only in speeding the model, but also seem to make the model more accurate as can be seen in Figure 3.

I trained the networks using the Softmax loss function, which is defined here for each image i. f_{yi} is defined as the value of the Softmax function for the correct class of the image, and the other f_i are the values of the Softmax function for the incorrect classes:

$$L_i = -\log \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$$

Finally, in each iteration, I used back-propagation to calculate the gradient with respect to the loss at each layer, then used an Adam policy to adapt the learning rate and change in each layers weights over time. Adam was a much better choice than vanilla gradient descent because Adams convergence rate is much faster than vanilla gradient descent and computational resources was a bottleneck in my research. l is the learning rate I specify at that layer, and m and v are initialized to 0. The Adam algorithm updates the weights vector. This update takes into account the concept of momentum of the learning rate and an approximation of acceleration.

$$m = \beta_1 m + (1 - \beta_1)\frac{\partial w}{\partial L}$$

$$v = \beta_2 v + (1 - \beta_2)(\frac{\partial w}{\partial L})^2$$

$$w = w - \frac{lm}{\sqrt{v} + \epsilon}$$

The resulting model has a testing accuracy of 89.12% and is, interestingly, less complex than the original CaffeNet model. The model was trained for 100 epochs with a batch size of 500. The training evolution can be seen in Figure 4. It is very interesting to note that there is no overfitting in this model. That is because it is a smaller model than the original CaffeNet, and it's better suited for this kind of images where the setting and lighting doesn't change so much. A grocery store usually has good lighting and its products are front faced and visually uncovered.

There are a few interesting takeaways from this experiment. First, each time that Guided_prunning showed a considerable increase in the model's accuracy was when a layer was dropped. This shows that the original model was too deep and complex for the problem. Second, the filter size did not make much of a relative difference in improving the model's performance.

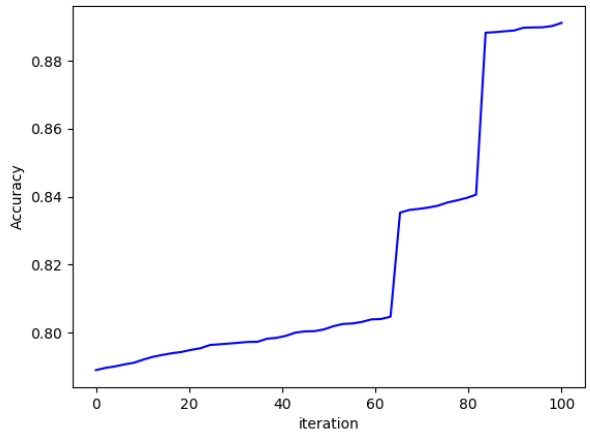

Figure 3: Guided_prunning running evolution: Note the steps. These happened when the algorithm dropped a full convolutional layer from the original model. In this case, due to overfitting, a smaller model was more accurate.

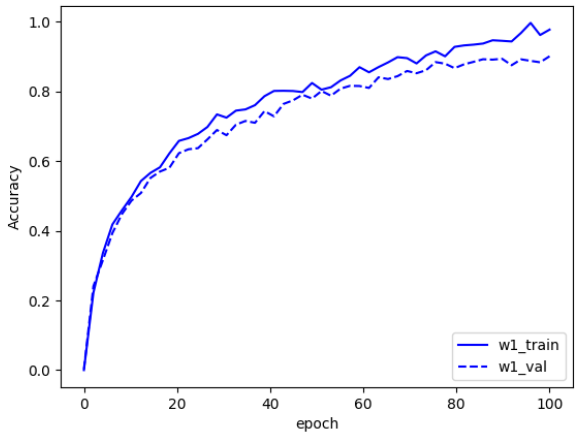

Figure 4: Training evolution of our proposed model: note that there is no overfitting. The final model is simpler, yet more accurate.

Most saliency maps show that the main feature the model used to classify is either text or edges.

This results show that using CV in a grocery environment is completely reasonable and that all the hype behind Amazon Go is funded.

5. Conclusion

This work presents an approach for retail product classification on grocery shelves and uses a novel dataset. I propose a model that is simpler and more accurate than the state of the art by 11 percentage points. I also propose Guided_Prunning, a greedy algoritm that prunes a convolutional neural networks looking for a more accurate one.

This research tested different CNN architectures (CaffeNet, AlexNet, GoogleNet, and the network found by Guided_Prunning) and surprisingly the simplest network was the most accurate one. A big takeaway is how sensitive to overfitting CNN are.

6. Future Work

In a future work I plan to evolve Guided_Prunning into a Reinforced Learning algorithm. Work has been done in this area [8]. One big challenge is the large resources needed to run such models.

References

[1] Philipp Jund, Nichola Abdo, Andreas Eitel, Wolfram Burgard. The Freiburg Groceries Dataset. 1, 2, 3

[2] A. Auclair, L. D. Cohenand N. Vincent, How to use SIFT vectors to analyze an image with database templates, Adaptive Multimedia Retrieval, ser. Lecture Notes in Computer Science, N. Boujemaa, M. Detynieckiand A. Nrnberger, Eds., vol. 4918. Springer, 224236, (2007). 1

[3] T. Winlock, E. Christiansenand S. Belongie, Towardreal-time grocery detection for the visually impaired,CVPRW,49 56, (2010).

[4] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision, vol. 60, no. 2, 91110, (2004).

[5] J. Kleban, X. Xieand W. Y. Ma, Spatial pyramid mining for logo detection in natural scenes, IEEE International Conference on Multimedia and Expo (2008). 1

[6] A. Opalach, A. Fano, F. Linaker, and R. Groenevelt, Planogram extraction based on image processing, Patent US 8 189 855, (2012). 1

[7] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, Trevor Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding." 1

[8] GBarret Zoph, Quoc V. Le, "Neural Architecture Search With Reinforcement Learning." 1

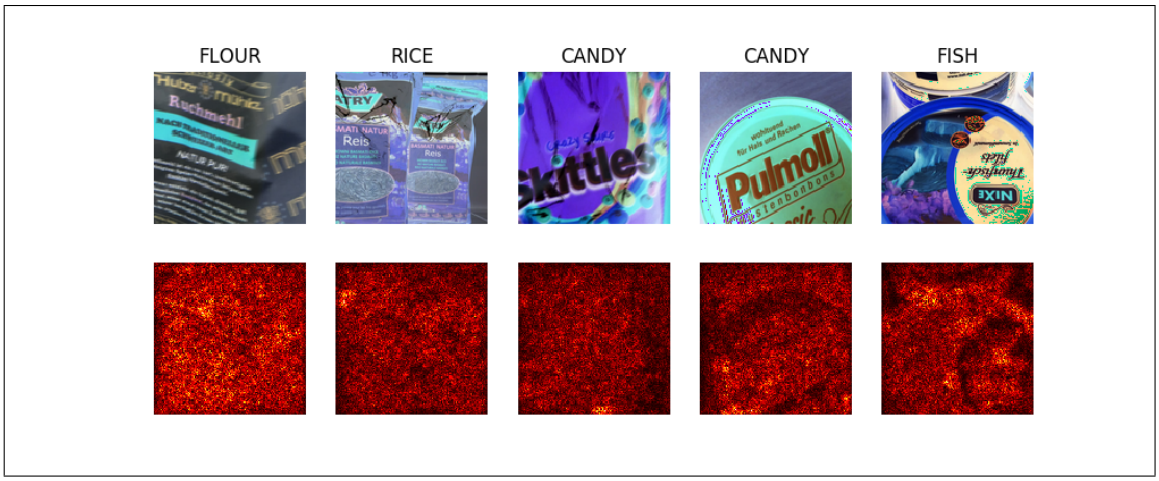

Figure 5: Saliency map of a random set of images

[9] James Bergstra, Remi Bardenet, Yoshua Bengio, and Balzs Kgl. Algorithms for hyper-parameter optimization. NIPS, 2011. 2

[10] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization.JMLR, 2012. 2, 5

[11] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. "Practical Bayesian optimization of machinelearning algorithms". In NIPS, 2012. 1

[12] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mostofa Ali, Ryan P. Adams, et al. "Scalable bayesian optimization using deepneural networks". In ICML, 2015. 1, 2

[13] Shreyas Saxena and Jakob Verbeek. "Convolutional neural fabrics". In NIPS, 2016. 1, 2

[14] James Bergstra, Daniel Yamins, and David D Cox. Making a science of model search: "Hyperparameter optimization in hundreds of dimensions for vision architectures". ICML, 2013. 1, 2

[15] Hector Mendoza, Aaron Klein, Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Towards automatically-tuned neural networks. In Proceedings of the 2016 Workshop on Automatic Machine Learning, pp. 5865, 2016.

[16] S. Ioffe and C.Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift.arXiv preprint arXiv:1502.03167, 2015. 2

2

4