Convolutional Neural Networks for Food Image Classification

Applications of Deep Learning

Carles Poles-Mielgo
M.S. Data Science Candidate: University of New Haven
44 Tehama Street
San Francisco, CA 94105, USA
carles.poles@gmail.com

***Abstract*—Wellio's recommendation system allows a user to ask for food recipes through a conversational interface. When retrieving the search results, Wellio ranks them using various criteria. Wellio has noticed that their customers tend to choose recipes that are more visually appealing when the results are displayed. Hence, Wellio needs to enable the search to also rank on appearance of the recipe. This paper addresses using deep learning to classify appropriate images for recipes. A dataset of 20,000 was used using transfer learning with the InceptionV3 architecture obtaining an 88% accuracy on the evaluation set.**

Keywords—deep learning, convolutional neural networks, image classification, getwellio.com

I. Introduction

We live in a society where time is a commodity, many individuals and families find themselves without the energy required to prepare a healthy and nutritious meal. Wellio wants to solve this paradigm with a simple application where any user can search over a million food recipes that have been gathered from hundreds of different websites related to food and cooking.

A given user that goes to the web based Wellio platform can search for any particular dish he or she wants to cook, and then choose from many different preparations. Each of the recipes is presented with users' reviews, a list of steps to prepare it, the ingredients required, its nutritional value, and most important, with a picture of the dish.

Wellio has found by gathering web usage data that users will abandon the web platform if the picture of the recipe is not appealing, therefore, losing a customer.

As an example, the vast majority of the recipes have been obtained from food.com, where individuals share their food preparations. The images from the associated recipes are taken from such individuals and are not professional, often taken with their smartphones and show pictures that are blurry, taken too close, or presented in a way that is not appealing, and then, they are uploaded to food.com. Even though the recipe may enjoy very good reviews, the visual impact of a bad picture has a negative effect on the user's perception.

The challenge is to find the best image for a given recipe to prevent customer abandonment by initially classifying images from appealing to not appealing.

II. Overview

A. Hypothesis

Personal visual aesthetics is a very personal choice [1] as a picture that may be appealing to somebody else, it may not to a different person. A strong training data set of pictures is a critical component to develop a convolutional neural network with a given architecture to classify any food image as "tasty" or not "tasty". Such training sets needs to contain a balanced amount of food images with high quality and food images with low quality.

Once the classifier is trained and provides a good accuracy on the testing dataset, it can be used to select the best image once a user searches for a particular dish and recipe, displaying the best picture along the preparation steps and ingredients.

B. Theory of the Solution

Since our problem has two possible outcomes (either the picture is good or not) a classification supervised machine learning model is the right solution to implement. Also, since we need to extract features from images, a convolutional neural network is the proper algorithm to implement.

Many different architectures of layers can be devised when designing a particular algorithm, where each layer extract features from the output of the previous layer using deep architectures.

Feature learning algorithms are able to find common patterns that are important to differentiate among classes so the features can be extracted to be used in a classification problem. In the context of deep learning, convolutional layers are very good at finding features in images to make them available to the next layer which generates a hierarchy of nonlinear features that grow in complexity. Each feature can be interpreted as a filter that takes an input image that is filtered using such filter.

We can only learn complex features with deep hierarchies of nonlinear features by stacking them up. In order to generate features that contain more information, we need to first transform the first features to get more complex ones that contain information that will be useful to distinguish among classes.

Now, hierarchical feature learning suffers from vanishing gradient, so these architectures perform poorly. This can be solved using deep learning, which uses new methods to generate deep hierarchies of nonlinear features thanks to the combination of the use of GPUs with activation functions that offer better gradient flow.

A layer is the building block of deep learning, and normally gets a weighted input and transforms it with nonlinear functions that are then passed to the next layer A given layer contains one type of activation function. The first and last layers are the input and output layers respectively, and the ones in between are called hidden layers.

In conjunction with above, we can use convolution, which is a mathematical operation describing how to mix pieces of information: the features with the convolutional kernel to form a transformed feature map. We can think of convolution as a filter, where the kernel filters the feature map to obtain some information (for example, use one kernel type to filter for edges and discard other information).

Therefore, convolutional filters can be interpreted as feature detectors: the input (feature map) is filtered for a certain feature (the kernel), and the output will be large if the feature is detected in the image.

A convolutional neural network uses convolutional layers that filters input for useful information. Each layer has parameters that are learned so the filters can be adjusted automatically to extract the most useful information for a given task.

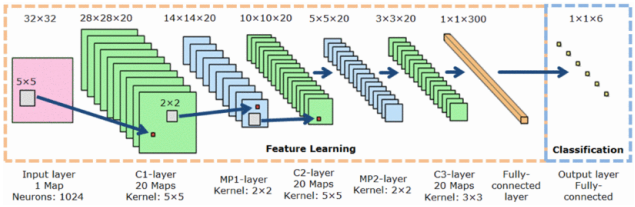


Fig. 1. Layers of a convolutional neural network [10].

C. Prior Literature

The paper titled "ImageNet Classification with Deep Convolutional Networks" by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton [8] describes a large, deep convolutional neural network that was used to win the 2012 ILSVRC (ImageNet Large-Scale Visual Recognition Challenge) which achieved a top 5 test error rate of 15.4%. In the paper, the group discussed the architecture of the network (which was called AlexNet) that uses a relatively simple layout, compared to modern architectures: five convolutional layers, max-pooling layers, dropout layers, and 3 fully connected layers.

The network was used for classification with 1000 possible categories.

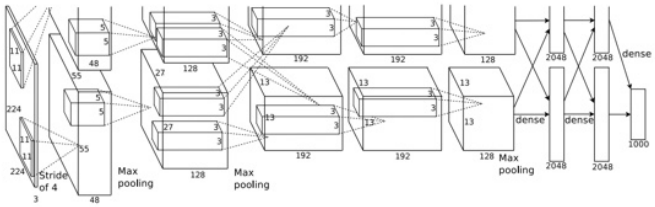


Fig. 2. Architecture of the AlexNet convolutional neural network [8].

In 2013, Matthew Zeiler and Rob Fergus published "Visualizing and Understanding Convolutional Neural Networks" [11] where a new convolutional neural network called ZF Net was described as a slightly modified AlexNet and showing how to visualize the filters and weights correctly.

Karen Simonyan and Andrew Zisserman [12] in 2014 created a nineteen-layer convolutional neural network that used 3x3 filters with stride and pad of 1, along with 2x2 max pooling layers with stride 2. It worked well on both image classification and localization tasks. It reinforced the notion that convolutional neural networks have to have a deep network of layers in order for this hierarchical representation of visual data to work.

Google introduced in 2014 the Inception module, called GoogLeNet [13] which consists in a twenty-two-layer convolutional neural network. A big difference with the other architectures is that that not everything happens sequentially, and some pieces of the network are happening in parallel. The inception module allows to perform all pooling operations or convolutional operations in parallel. The whole architecture uses nine inception modules, resulting in over one hundred layers in total, and not using fully connected layers.

III. Implementation

A. Choice of Tools

All work has been developed using Google's Cloud infrastructure. It works on Python 2.7, and therefore, a local environment with Python 2.7 is required to work with Google Cloud by installing Google Cloud Platform SDK. Keras using Tensorfow as backend is the API of choice to develop and implement different convolutional neural networks.

InceptionV3 have been also used to implement an image classification model with transfer learning from imaging preprocessing, training and evaluation the model, saving it and making predictions, as well as visualize it using Tensorflow.

A small web application in Flask is also used to demonstrate the use of the model by making live predictions of images.

B. *Choice of Data*

The majority of the recipes have been provisioned from food.com, been eipicurious.com the second source of data.

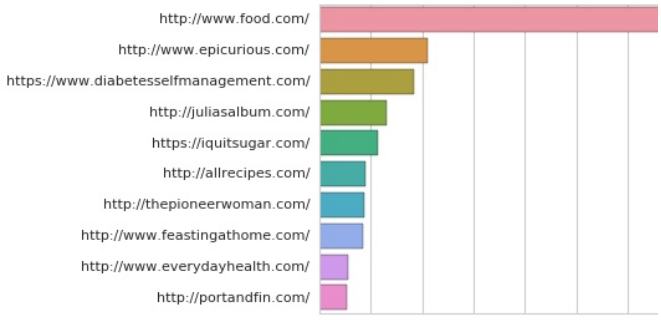


Fig. 3. Barplot indicating the source of the top websites from where recipes are obtained.

Food.com is a very popular destination for people searching for food recipes that are created by amateurs who also take pictures of their dishes using their smartphones, resulting in images of low quality. On the contrary pictures of dishes from epicurious.com have professional grade. Most of the picture reviews come from Pinterest, as opposed to Facebook (been the first social media channel specialized in image sharing).

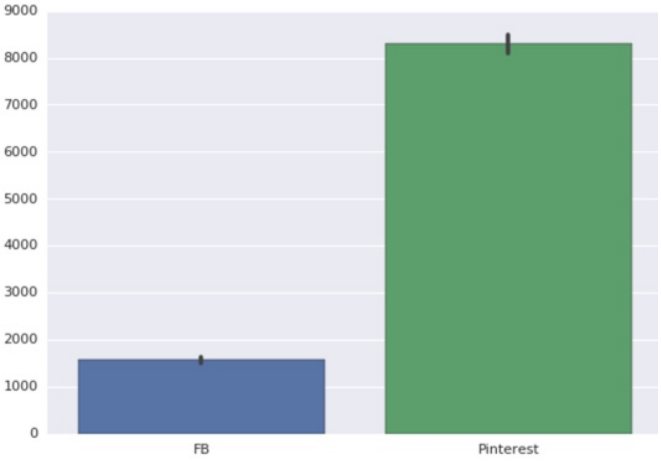


Fig. 4. Most of the food picture rating come from Pinterest.

Since most of the high rating for images come from epicurious.com and the lowest ones from food.com, it was decided to download and use images from epicurious.com as "tasty" ones and images from food.com as "not tasty" ones to curate our dataset.

Fig. 5. On top, an image for a chicken recipe from food.com which is of low quality, and below an image from epicurious.com of professional grade.

The dataset was prepared using a balanced number of images from both websites. Images come in different sizes, mostly 480 x 640 pixels (height and width).

The URLs to download the images were obtained by using BigQuery: 10,000 images from each of the above-mentioned websites were downloaded.

C. *Choice of Models*

Two convolutional neural networks architectures have been used to train models:

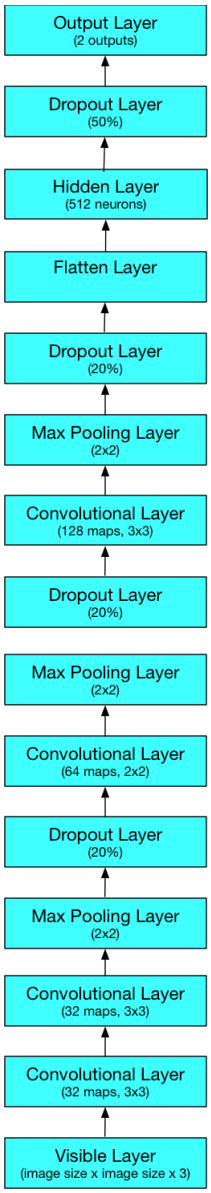


Fig. 6. An architecture composed of 14 convolutional layers based on F. Chollet research [9].

A simpler variation of the previous architecture has been also used to train different models:

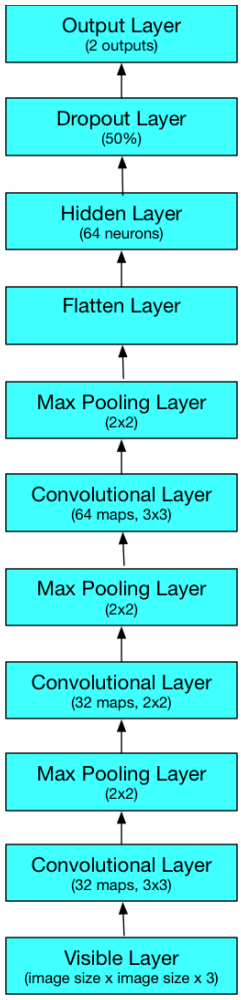


Fig. 7. An architecture composed of 10 convolutional layers.

All images were pre-processed using Keras to different sizes to create different datasets: 25x25, 50x50 and 100x100. The sizes were chosen arbitrarily and having in mind the MNIST dataset where images have a size of 28x28, and great accuracy has been achieved in different convolutional architectures.

IV. Experimental Results

The table below summarizes the results using the architecture with 14 layers:

TABLE I.

Image Size	**14 Convolutional Layers (100 epochs)**	
	Accuracy	***AUC***
25	81.48%	81.52%
50	83.87%	83.88%
100	87.10%	87.11%

Similarly, for the architecture depicted on Figure 7:

TABLE II.

Image Size	10 Convolutional Layers (100 epochs)	
	Accuracy	*AUC*
25	77.08%	77.09%
50	81.75%	81.77%
100	82.43%	82.48%

Data augmentation was also used and it was only performed using the architecture with 10 layers. The table below summarizes the findings:

TABLE III.

Image Size	10 Convolutional Layers (data augmentation – 25 epochs)	
	Accuracy	*AUC*
25	81.68%	81.75%
50	88.67%	88.65%
100	89.20%	89.22%

Here we show a ROC curve that corresponds to the 14-layer architecture and images with size 100, 50 and 25.

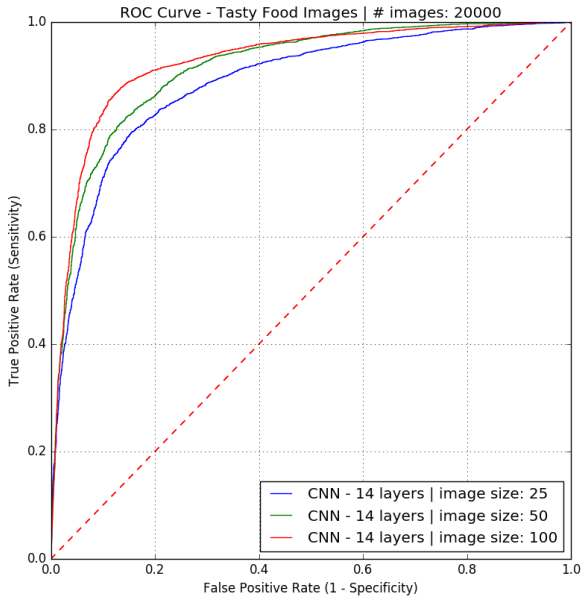


Fig. 8. ROC curve obtained with 20,000 images resized to 100x100, 50x50 and 25x25 using a 14-layer convolutional neural network architecture.

We also display the confusion matrix for the 14-layer architecture using images of size 100x100.

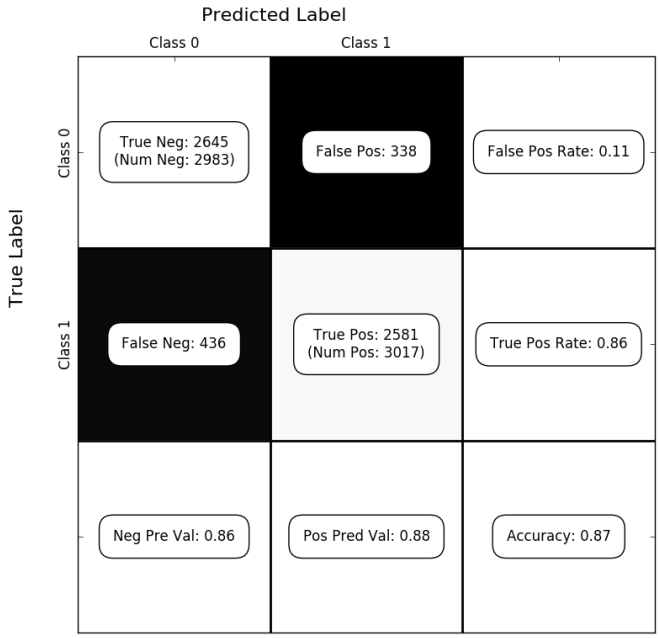


Fig. 9. Confusion matrix obtained with 20,000 images resized to 100x100.

The corresponding ROC curve that corresponds to the 10-layer architecture and images with size 100, 50 and 25:

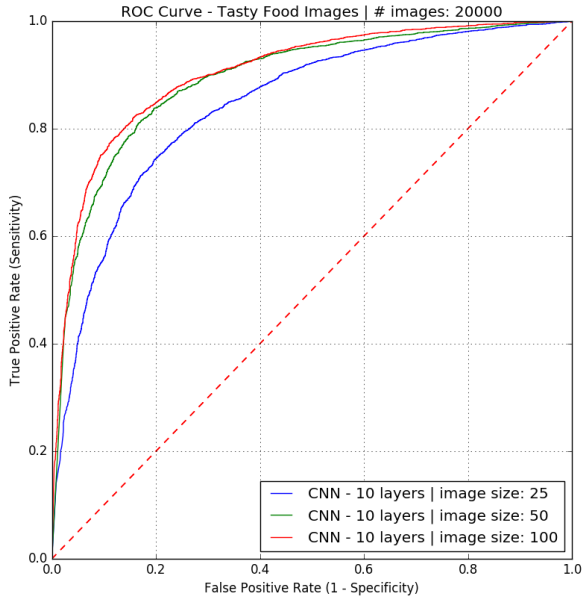


Fig. 10. ROC curve obtained with 20,000 images resized to 100x100, 50x50 and 25x25 using a 10-layer convolutional neural network architecture.

Similarly, let's display the results using data augmentation with the 10-layer architecture. The corresponding ROC curve:

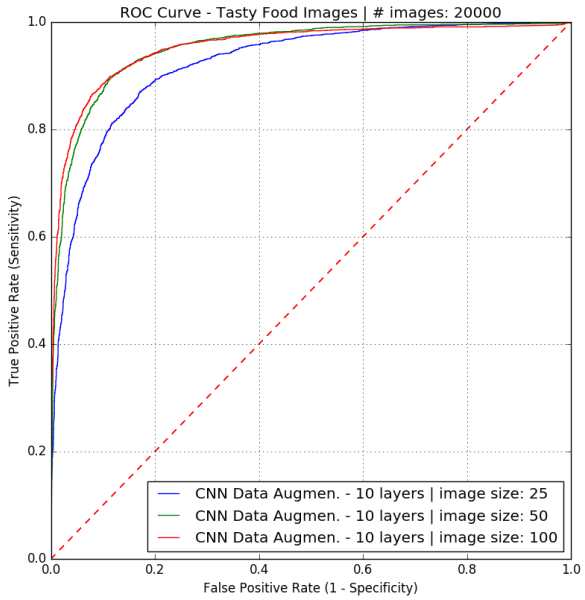


Fig. 11. ROC curve with data augmentation with 20,000 images resized to 100x100, 50x50 and 25x25 using a 10-layer convolutional neural network architecture.

Transfer learning was attempted using the InceptionV3 architecture. The 2,048 features from the tensor name 'pool_3:0' were used to extract the features of our 20,000 images.

Different models were trained using the features that were extracted, like logistic regression, Naïve-Bayes, random forests and linear SVM, but the accuracy obtained in all cases were around 50%.

Feature importance from random forests revealed that no feature was relevant at all to explain the low accuracy of the models. This is caused by the fact that the embeddings are too complex for these classifiers.

Another attempt using image based transfer learning with InceptionV3 has been implemented on Google Cloud ML: it uses Apache Beam (running on Cloud Dataflow) and PIL to preprocess the images into embeddings, then train and serve a model on Cloud ML. It supports distributed training on Cloud ML. We use InceptionV3 penultimate "bottleneck" layer to train a new top layer that can recognize other classes of images, like "tasty" or "not-tasty" in this project: the new top layer does not need to be very complex, and that we typically don't need much data or much training of this new model, to get good results for our new image classifications. It's called "bottleneck" to refer to the layer just before the final output layer that actually does the classification. This penultimate layer has been trained to output a set of values that is good enough for the classifier to distinguish between "tasty" or "not-tasty" images.

Preprocessing means that we extract an image features from the "bottleneck" layer which is the penultimate layer of the Inception network. This is achieved by loading the saved Inception model and its variable values into TensorFlow, and run each image through that model.

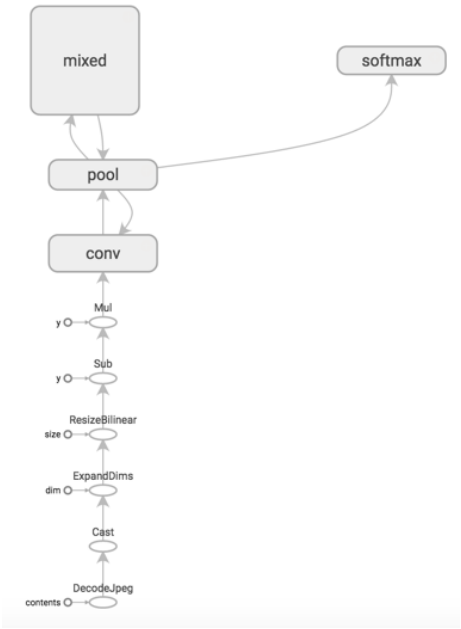
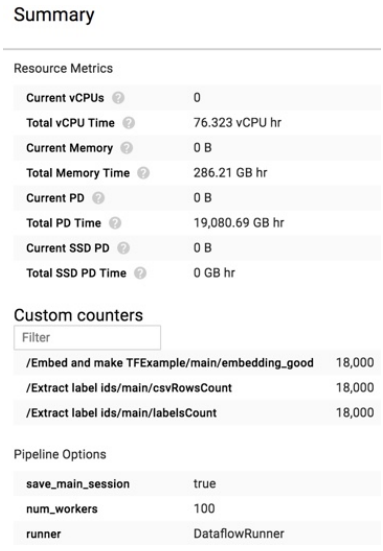


Fig. 12. InceptionV3 graph model.

Each image is processed to produce its feature representation (an embedding) which is a k-dimensional vector of floats (in our case, 2,048 dimensions). The preprocessing includes converting the image format, resizing images, and running the converted image through a pre-trained model to get the embeddings.

The reason this approach is so effective for bootstrapping new image classification is that these 'bottleneck' embeddings contain a lot of high-level feature information useful to InceptionV3 for its own image classification.



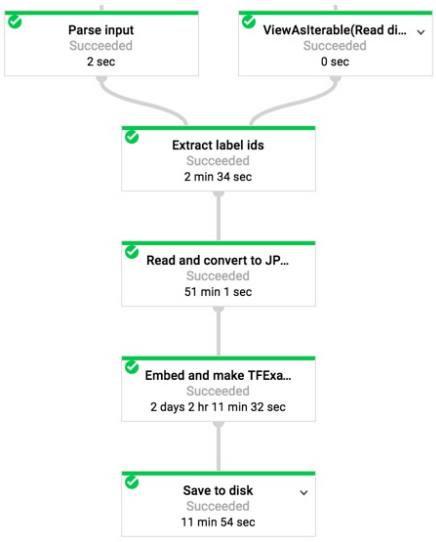


Fig. 13. Embedings created for training set. View from Google Dataflow.

The neural network will comprise a single fully-connected layer with RELU activations and with one output for each label ("tasty", "not tasty") to replace the original output layer.

Once the job is completed, the model and a graph are saved.

We can create a prediction locally: we need to download a given image, and since the image needs to be passed via JSON, we have to encode the JPEG string first.

Tensorboard has been used to visualize loss and accuracy among others.

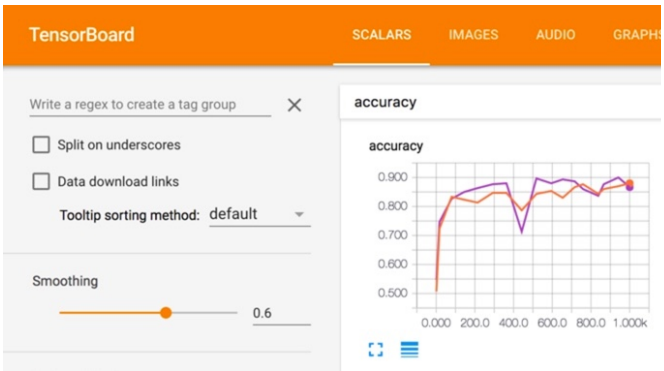


Fig. 14. Looking at the accuracy using Tensorboard.

Embeddings can be also be visualized in Tensorboard.

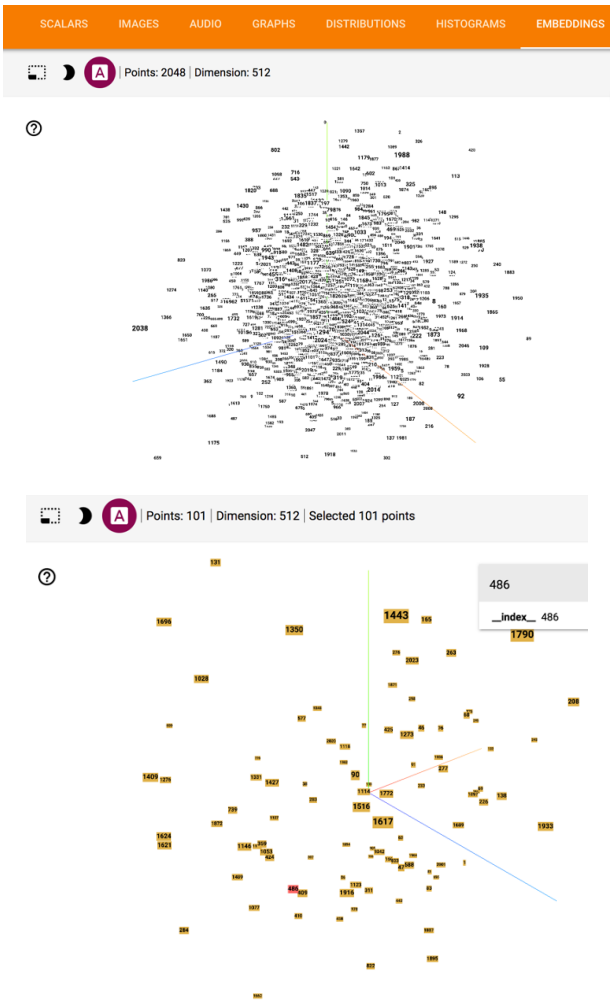


Fig. 15. Using PCA in Tensorboard, we can look at the embeddings as well as isolate some of them.

Computational graphs can be also visualized in Tensorboard.

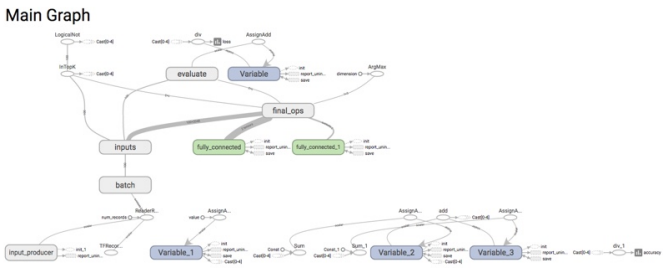


Fig. 16. A view of a complete computational graph in Tensorboard.

Finally, using a Flask application, we can visualize predictions on a given image. Using an image from epicurious.com:

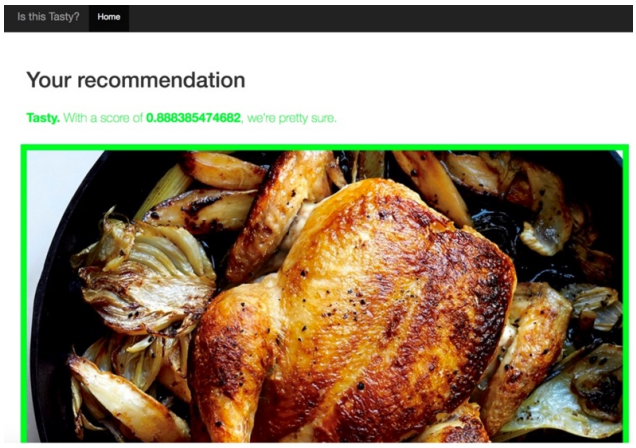


Fig. 17. Making a prediction using a picture from epicurious.com. The picture is professional grade.

Making another prediction from a picture from food.com:

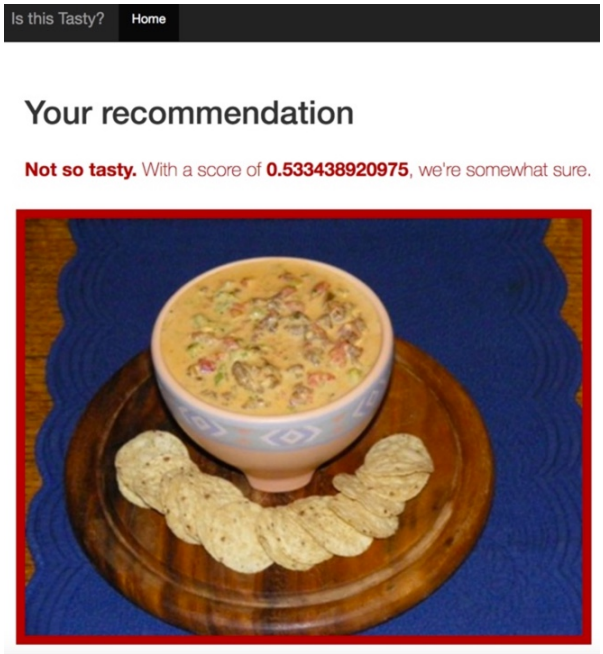


Fig. 18. Making a prediction using a picture from food.com. The picture is amateur and fuzzy.

V. Conclusion And Future Research

Two major outcomes from this project are that the preprocessing size of the images is relevant: the bigger the size, the most accuracy in a model, and that image augmentation greatly impacts the accuracy of a model.

Also, a deeper convolutional neural network architecture leads to better accuracy.

Using InceptionV3 to implement transfer learning can be easily modified and adapted for any image classification problems with different images. The new top layer does not need to be very complex, and that we typically don't need much data or much training of this new model to get good results for our new image classifications. Transfer learning is extremely useful as it used a very deep architecture that has been trained on millions of images.

Future work will lead to implement a convolutional neural network with high accuracy in production to minimize customer defection as well as increase the training dataset with more food images, and continue development neural network architectures to extract ingredients and its nutritional value from the best image for a given recipe.

All work can be found on this github repository: https://github.com/carlespoles/DSCI6051-student.

Acknowledgment

I want to thank the guidance and support provided by Sivan Aldor-Noiman and Erik Adrejko from Wellio towards the success of this project, as well as my fellow intern Pranava Prabhakaran and my advisor Alessandro Gagliardi and Conor Murphy, members of the faculty of the University of New Haven.

References

[1] A. Shaji and G. Yildirim, “Personalized Aesthetics: Recording the Visual Mind using Machine Learning,” https://devblogs.nvidia.com/parallelforall/understanding-aesthetics-deep-learning/, March 2017.

[2] V.S. Ramachandra, W. Hirstein, “The science of art: a neurological theory of aesthetics experience,” Journal of Consciousness Studies, 1999.

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE 86(11): 2278–2324, 1998.

[4] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, et al. “Handwritten digit recognition with a back-propagation network,” In Advances in neural information processing systems, 1990.

[5] Y. LeCun, F.J. Huang, and L. Bottou, “Learning methods for generic object recognition with invariance to pose and lighting,” In Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on, volume 2, pages II–97. IEEE, 2004.

[6] Y. LeCun, K. Kavukcuoglu, and C. Farabet, “Convolutional networks and applications in vision,” In Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on, pages 253–256. IEEE, 2010.

[7] A. Berg, J. Deng, and L. Fei-Fei, “Large scale visual recognition challenge 2010,” www.imagenet.org/challenges, 2010.

[8] A. Krizhevsky, I. Sutskever, G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” Advances in Neural Information Processing Systems 25 (NIPS 2012).

[9] F. Chollet, “Building powerful image classification models using very little data,” The Keras Blog, https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html, June 2016.

[10] B. Pavol, Y. Rafailovich Nikitin, P. Božek, "Robotic Grasping System Using Convolutional Neural Networks," American Journal of Mechanical Engineering 2.7 (2014): 216-218.

[11] M. Zeiler, R. Fergus, "Visualizing and Understanding Convolutional Networks," arXiv:1311.2901v3 (2013).

[12] K. Simonyan, A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv:1409.1556 (2014).

[13] C. Szegedy, We. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, “Going Deeper with Convolutions," arXiv:1409.4842v1 (2014).