**Team 8 - ALCOVE - Attention to the right details with detailed attention.**

Heena Khan, James Phillips, Luis Chunga, Elijah Barbour, Mason Thieman, Matthew Radice, Steven Sheffey

Yellow is plagarized… fully remove the yellow parts in final part but those parts are where the info is pulled from

Green is information that should be implemented in the paper only if ours lines up with theirs.

Purple is app info.

Basically fill in the blanks and elaborate on the parenthesis parts and whalla we are done. Well discussions and conclusions is still needed. And abstract. That’s about it.

Image Recognition Software for a Web App

# Introduction and Background

Students at MTSU developed a free food-finding app during HackMT where anybody can search for free food around them just by entering their location, and anybody can donate their food just by filling in information about the food type and quantity they are offering. Currently, the food donor must fill this information every time they are hosting food. We are going to incorporate an image recognition neural network into this web app to determine the food information automatically from pictures.

## Motivation

In order to reduce the amount of work required to donate food, we will be implementing a Neural Network to automatically identify the donor’s food through a picture. Not only will this project reduce the work required to donate food on the app, it will also reduce the amount of mislabeling errors. This will not only be more convenient for the donor, but also will be more convenient for the organization taking the food. In implementing this Neural Network into the app, we prevent food wastage and help provide food to someone who needs it without paying for it. The image recognition feature will add great value to the app.

## Reference Related Work on Similar Problems or that has used similar approaches

We researched a similar project named “Food-101; Mining Discriminative Components with Random Forests”. This project focused on creating a Neural Network that helps organize and classify different types of food. We used the same dataset (Food-101) that was used in their paper. FIND PAPERS WITH MATH IN IT, AND REFERENCE THEM HERE. BASICALLY, DO METHODS FIRST AND OUTLINE WHAT WAS DISCUSSED IN METHODS HERE.

## Our Main Objective

We are planning on building a Convolutional Neural Network (CNN) using TensorFlow to classify images. CNNs are efficient in correlating and understanding a large amount of data in high-resolution images, and they are best known for their ability to recognize patterns present in images. The goal is to train a CNN to be as accurate as possible in identifying different cuisines and their servings, this model will then be implemented as a web app using React-js on the front end

# Methods

## Mathematical Execution

* The input images were scaled down to 80 × 80 and cropped to 64 × 64 pixels randomly by the cuda-convnet python module
* Our images are 289 x 289. Nothing external just we are running the code with 289 x 289
* The input contains 128 × 128 × 3 neurons, representing the RGB values for a 128 × 128 × 3 image.
* Because we scaled the images down to \_\_\_\_\_\_ we were then able to have an input layer that consists of \_\_\_\_ (128 x 128 x 3 neurons) \_\_\_\_\_.
* **Watch the video on Dr. Phillips’s site to contribute to this section**

## Theoretical Execution

* The dropout that randomly eliminate a portion of neurons from the network was used to reduce possible overfitting. The dropout rates of 0.25 and 0.5 were set for the third convolutional-pooling layer and the fully-connected layer, respectively

In order to reduce the probability of overfitting, we implemented a dropout rate of \_\_\_\_\_ for the \_\_\_\_\_th layer, a \_\_\_\_\_ dropout rate for the \_\_\_\_\_\_th layer, and a \_\_\_\_\_\_ dropout rate for the \_\_\_\_\_th layer.

* In accordance with Krizhevsky’s CNN [7], we use local response normalization (LRN) for the normalization after pooling layers

Similar to Yuzhen Lu et. Al., In order to normalize after pooling the layers, we used \_\_\_\_\_\_\_\_ (with respect to Krizhevsky’s CNN network that was recognized as the winner of the 2012 ImageNet Large-Scale Visual Recognition Challenge competition.)

* The network has four layers of hidden neurons (three convolutional-pooling and one fully-connected), apart from a final layer of output neurons (the input is not considered as a layer)

Our Convolutional Neural Network is comprised of an input layer of \_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_ (X layers of hidden neurons), and one more layer for output. (if you have a count of the number of neurons for any of the layers, then tell me and I can put them here.)

* The first convolutional-pooling layer uses a local receptive field (also known as convolutional kernel) of size 7 × 7 with a stride length of 1 pixel to extract 32 feature maps, followed by a max pooling operation conducted in a 2×2 region; the second and third convolutional-pooling layers use 5×5 and 3×3 local receptive fields, resulting in 64 and 128 feature maps, respectively, and the other parameters remain unchanged. The fourth layer is a fully-connected layer with 128 rectified linear units (ReLU) neurons, and the output layer has 10 softmax neurons that correspond to the ten categories of food.

The first layer uses a kernel size of \_\_\_\_\_(input info about kernel size and info about the purpose of the first layer)\_\_\_\_\_. We used \_\_\_\_\_\_\_\_\_ as the activation function for the first layer. The second layer uses a kernel size of \_\_\_\_\_\_\_(input info about kernel size/purpose of the second layer)\_\_\_\_\_ (info about activation function). The third………….. There are \_\_\_\_\_\_\_ output neurons that represent our food categories. (obviously add more/less information with respect to the actual number of layers used in our CNN)

* Information about the theoretical aspect (large scale approach) of implementing this into the app.

## Practical Execution

* We divided the dataset into six sets: four sets were used for training, one for validation, and one for testing
* In order to successfully train and test the network, we divided our dataset. \_\_\_\_\_\_\_\_ was used for training the network, \_\_\_\_\_\_\_\_ was used to validate (blah blah blah) and the remaining \_\_\_\_\_\_ was used to test the trained network.
* Training a CNN model requires to select a set of hyper-parameters, among which the leaning rate η is the the most critical one affecting the training performance. (What are the hyperparameter values, what are the hyperparameters, and why did you pick those numbers
* We used a learning rate of \_\_\_\_\_\_\_\_\_ and a momentum value of \_\_\_\_\_\_\_\_ (and other hyperparameter info). (At first, we tried a learning rate of \_\_\_\_\_ and momentum value of \_\_\_\_\_, but we found that \_\_\_\_\_\_\_\_\_\_\_\_(accuracy was too low and/or training took forever/// or other reasons why those values were not used)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_. In researching and testing different values for the learning rate and momentum, we found the learning rate of \_\_\_\_\_\_ and momentum value of \_\_\_\_\_ to be the most effedctive. We concluded that these values allowed us the best results as far as accuracy and training speed.
* WHAT IS THE batch size? Number of epochs used

For our CNN, we used a batch size of \_\_\_\_\_\_\_\_\_ and \_\_\_\_\_\_\_ epochs. We used (these values because\_\_\_\_\_)

* At the initial stage, the training loss was large, resulting in a large learning rate to speed up the training process; gradually, the learning rate decreased with the loss, which helped avoid overshooting the best result
* Explain this but fit it within our experiences.
* The network was trained with the stochastic gradient descent (SGD) algorithm with a cross-entropy cost function1
* Nani?! If we did something like this, then put it here.
* Info about the processes to implement the CNN into the app.

# Results

* Tell the overall accuracy here
* Apple and pizza gave the two highest recognition rates, which were mainly because the two categories had a large number of training images; while french-fry and broccoli was the two hardest categories, the majority of which were misclassified into another distinct class
* Basically, explain the results of each of the labels/tags. If there are subset accuracy values, tell them here and further explain the reason behind the accuracy of the network for each tag.
* Charts showing progression/accuracy and loss? (IS IT BASED ON DATA AUGMENTATION OR NOT?!)

**Charts here**

* According to the accuracy and loss curves, it seemed that the CNN model could be further improved by increasing the training epochs

Looking at our results, For the CNN model to improve (modifications are needed in) \_\_\_\_\_\_\_\_\_.

* Compared with Table 2, the CNN model resulted in the overall accuracy improved from 56% to 90%, and the average recognition rate from 0.78% to 94%
* Was there any values of accuracy between other values/models other than the one we had? If so, then put info here.

# Discussion and Conclusion

# Abstract

.

# References

1. Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discrimi-native components with random forests. In European Conference on Computer Vision,2014

<https://www.bing.com/search?q=food+recognition+convolutional+neural+networ&cvid=84af24551cfe4f85ab7e8449bd024889&FORM=ANNTA1&PC=U531> …..……… <https://arxiv.org/pdf/1612.00983.pdf>