

# Classification of Food Images Using Convolution Neural Networks

Group 7

Joanne Chung, McLain Wilkinson, Ruonan Jia


# Outline

---

- Introduction
- Project Process
- Data Overview
- Data Preprocessing
- Network / Model Design
- Results
- Conclusion
- Future Work
- Reference

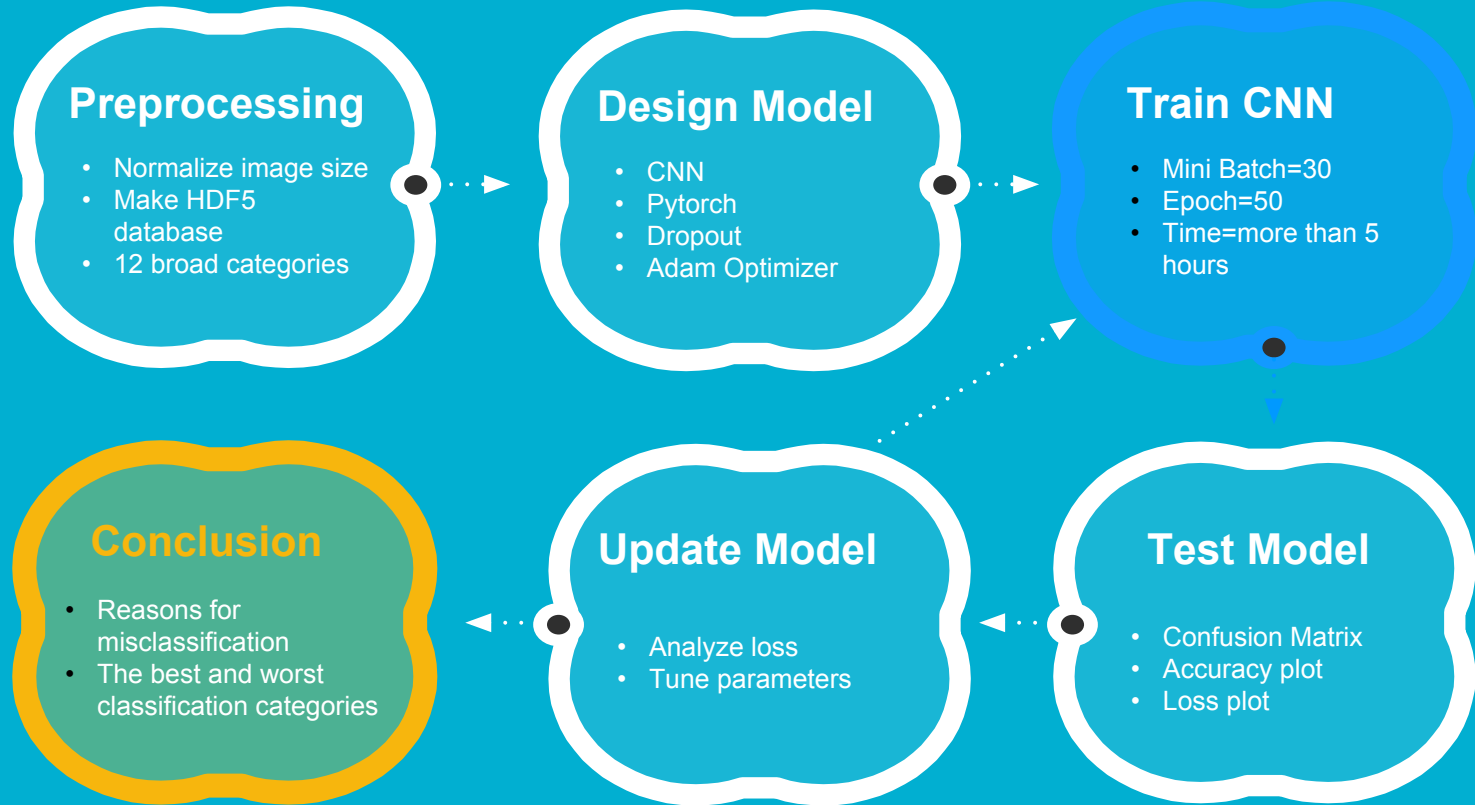
# Introduction

---

- Services such as  contain pictures of menu items relevant to restaurant but do not automatically tag images -
  - rely on self tagging from users, which makes it difficult to find pictures through search
- For a more challenging problem, we wanted to try to build such a classification system that could identify the food item contained in the image
- This problem is difficult by nature - food items themselves can be visually ambiguous, since they may take on a variety of shapes, colors, and textures and an image may contain more than 1 dish

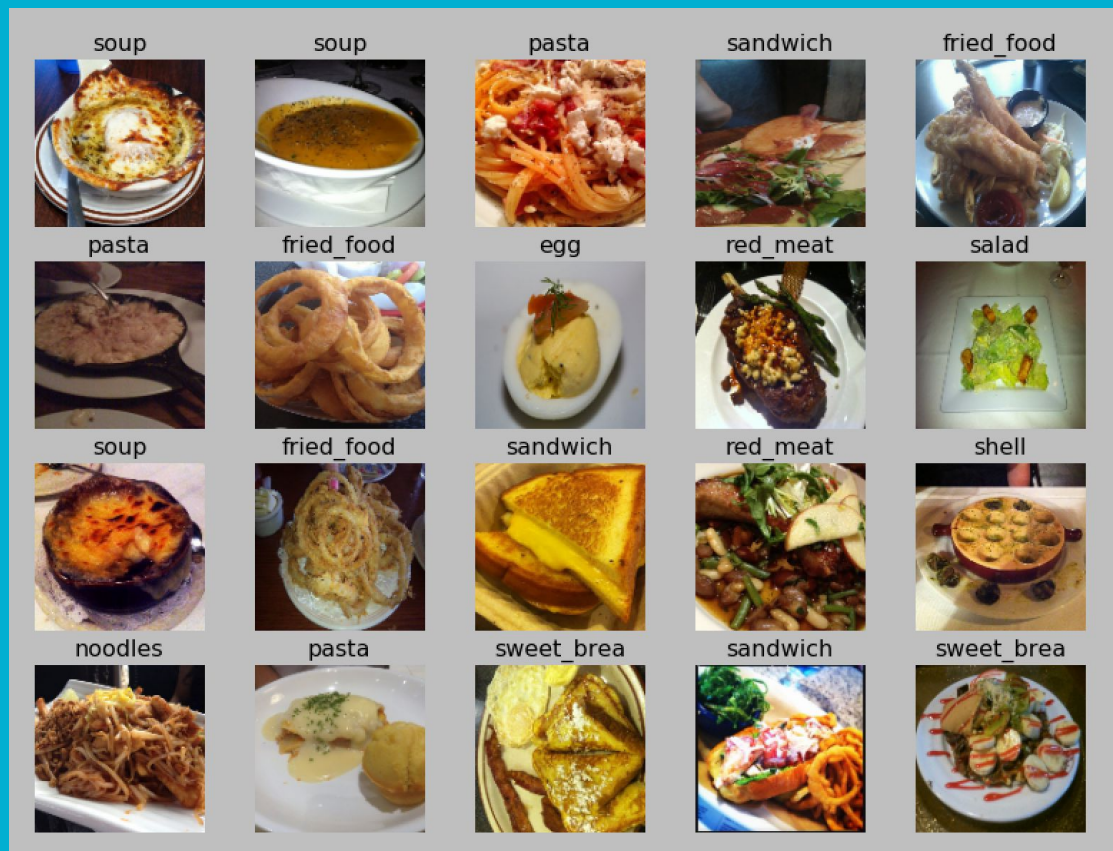
# Project Process

---



# Data Overview

---



# Data Overview (continued)

---

- Found dataset on Kaggle (<https://www.kaggle.com/kmader/food41>)
- Sample datasets use HDF5 database files
- Images folder contains .jpg images (101 categories of 1000 images each)
- Create database using these images
- We want to combine some categories and remove others to increase the number of training examples per class and reduce number of total classes



# Data Preprocessing

Is a  a sandwich?

## HDF5 database

Create python script convert 101,000 .jpg images to HDF5 database files containing labeled color images of reduced size I/O

## Resize Image

Typical .jpg image size ~512x512 pixels - want to reduce to 128x128 with 3 channels for color (clear image without taking too much storage)

## Combine classes

Combine classes to create 12 relatively broad classes. Difficult to determine what classes could reasonably be combined

## New Dataset

New dataset contains 48,000 total images with ~4,000 images per class

## Split Data

Split into training and testing sets using 80/20 split stratified by class

# Data Preprocessing (continued)

<b>Smooth Dessert</b>	Chocolate mousse	Frozen yogurt	Ice cream			
<b>Red Meat</b>	Filet mignon	Pork chop	Prime rib	steak		
<b>Egg</b>	Bibimbap	Deviled eggs	Eggs benedict	Huevos rancheros		
<b>Pasta</b>	Gnocchi	lasagna	Macaroni and cheese	Ravioli	Spaghetti bolognese	Spaghetti carbonara
<b>Soup</b>	French onion soup	Hot and sour soup	Lobster bisque	Miso soup		
<b>Salad</b>	Caesar Salad	Greek salad				
<b>Fried Food</b>	Fish and chips	French fries	Fried calamari	Onion rings	Poutine	
<b>Sandwich</b>	Club sandwich	Grilled cheese sandwich	Hamburger	Hot dog	Lobster roll sandwich	Pulled pork sandwich
<b>Noodles</b>	Pad thai	Pho	Ramen			
<b>Cake</b>	Carrot cake	Chocolate cake	Cup cakes	Red velvet cake	Strawberry shortcake	
<b>Sweet Breakfast</b>	French toast	Pancakes	Waffles			
<b>Shell</b>	Escargots	Mussels	Oysters			



# Cake

Cupcake



Red  
Velvet  
Cake



Strawberry  
Shortcake



Chocolate  
Cake



Carrot  
Cake



# Network / Model Design

---

- Pytorch Framework
- 6 Convolution/Max Pooling layers & 1 Fully Connected Layer with dropout
- Number of layers & size of kernels limited by input image size
  - Kernel size = 10 for first 4 layers, 4 for 5th layer, 2 for last convolution layer
  - Numbers of kernels (ranging from 64-384 per layer)
- Batch size: 40
- Cross Entropy Loss
- Adam Optimizer
- Experiment with learning parameters & adjust when necessary to improve total accuracy on test set

# Parameter Selection

- Numbers kernels decreasing reduce neurons before fully connected layer also capture the most features at the first layer
- Kernel size high to low (visualize the feature map pattern better)
- Layers=2, 4, 6

Model 1		Model 2		Model 3	
Kernel size	10	Kernel size	5	Kernel size	10
Number of kernels	128*4-64	Number of kernels		Number of kernels	
Layers	6	Layers	6	Layers	6
Accuracy	52%	Accuracy	57%	Accuracy	54%
Time	1523 s	Time	1569s	Time	

Epochs=5  
Time cost is high

# With and without dropout node layer

---

We don't need this slides, we need dropout for almost all of our neural network  
Overfitting happens when data is not enough, very hard to be underfitting in neural network

Epochs not going to affect overfitting????

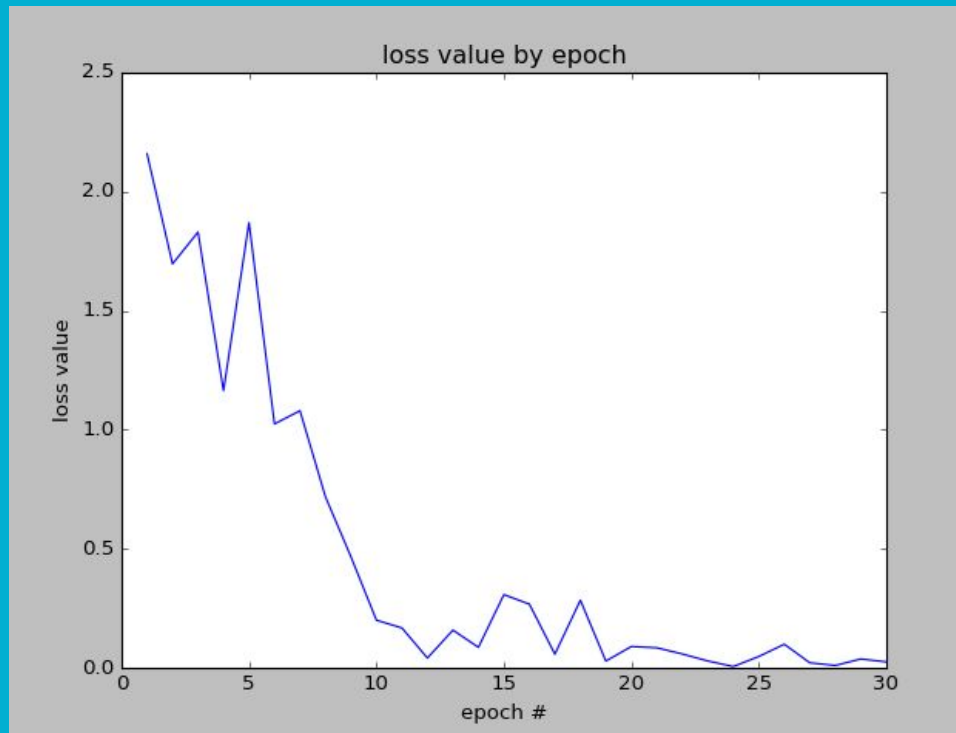
We will put dropout at fully connected layer just in case overfitting  
impossible underfitting in Neural Network

# Network / Model Design

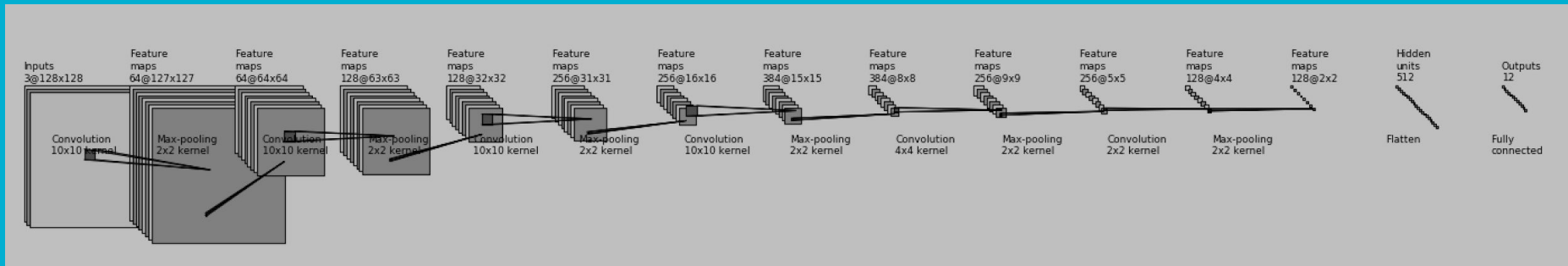
---

## Training Loss

- How many epochs is enough?
- Loss value stays roughly constant after ~25 epochs
- 12 epochs VS 25 epochs
- Time Cost Worth it?



# Network Diagram



6 convolution/MAXpooling layers

Kernel Size[10, 2, 10, 2, 10, 2, 10, 2, 4, 2, 2, 2]

Image sizes through the convolution/pooling layers:

128-->127-->64-->63-->32-->31-->16-->15-->9-->5-->4-->2

# Our Favorite Equations

---

- Convolution size equations are helpful to calculate sizes of outputs in convolution layers

$$w = (w + 2 * \text{PAD} - \text{kernel size}) / \text{STRIDE} + 1$$

$$h = (h + 2 * \text{PAD} - \text{kernel size}) / \text{STRIDE} + 1$$

$$\text{CONV1: } w = (128 + 2 * 4 - 10) / 1 + 1 = 127 \quad h = (128 + 2 * 4 - 10) / 1 + 1 = 127$$

$$\text{Pool1: } w = (127 + 1) / 2 = 64 \quad h = (127 + 1) / 2 = 64$$

# Results

Output and accuracy of  
model on 9600 test  
images after 30 training  
epochs

training time: 2h 38m

60% overall accuracy

individual class  
accuracies also listed  
here

```
Epoch [30/30], Iter [900/960] Loss: 0.0324  
training time: 9490.65 seconds
```

```
-----  
Testing...
```

```
-----  
Results
```

```
Accuracy of the model on the test images: 60 %
```

```
Individual class accuracy:
```

smooth_des	290 correct / 600 total	=>	48.33 % accuracy
red_meat	533 correct / 800 total	=>	66.62 % accuracy
egg	427 correct / 800 total	=>	53.38 % accuracy
pasta	795 correct / 1200 total	=>	66.25 % accuracy
soup	618 correct / 800 total	=>	77.25 % accuracy
salad	244 correct / 400 total	=>	61.00 % accuracy
fried_food	660 correct / 1000 total	=>	66.00 % accuracy
sandwich	591 correct / 1200 total	=>	49.25 % accuracy
noodles	351 correct / 600 total	=>	58.50 % accuracy
cake	655 correct / 1000 total	=>	65.50 % accuracy
sweet_brea	273 correct / 600 total	=>	45.50 % accuracy
shell	358 correct / 600 total	=>	59.67 % accuracy

```
class with highest accuracy: soup 77.25 %
```

```
class with lowest accuracy: sweet_brea 45.50 %
```

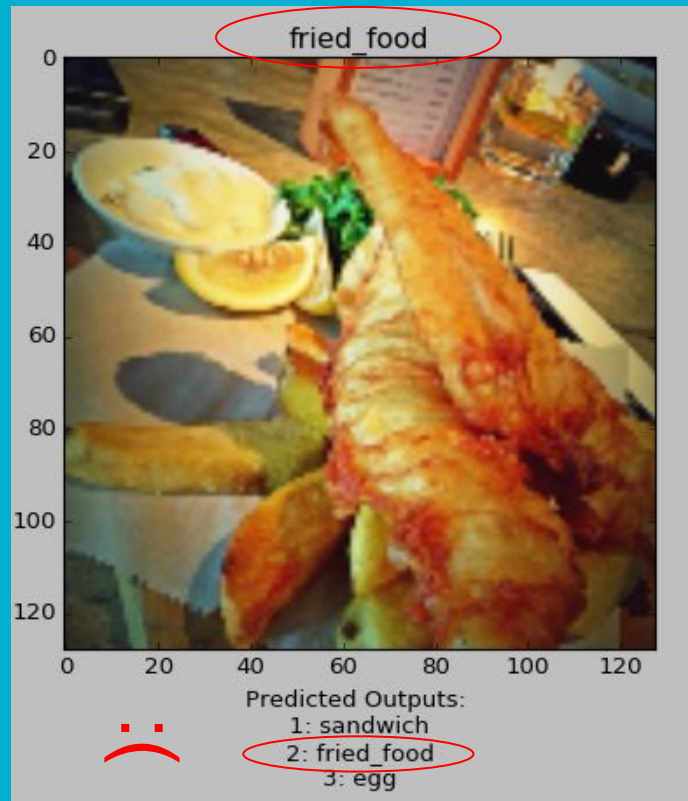
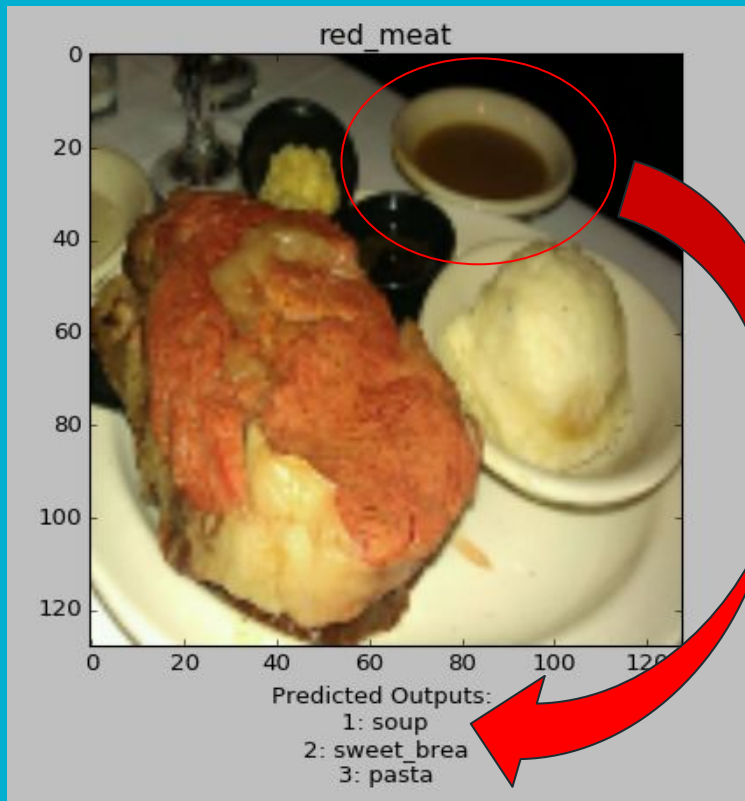


# Results (continued)

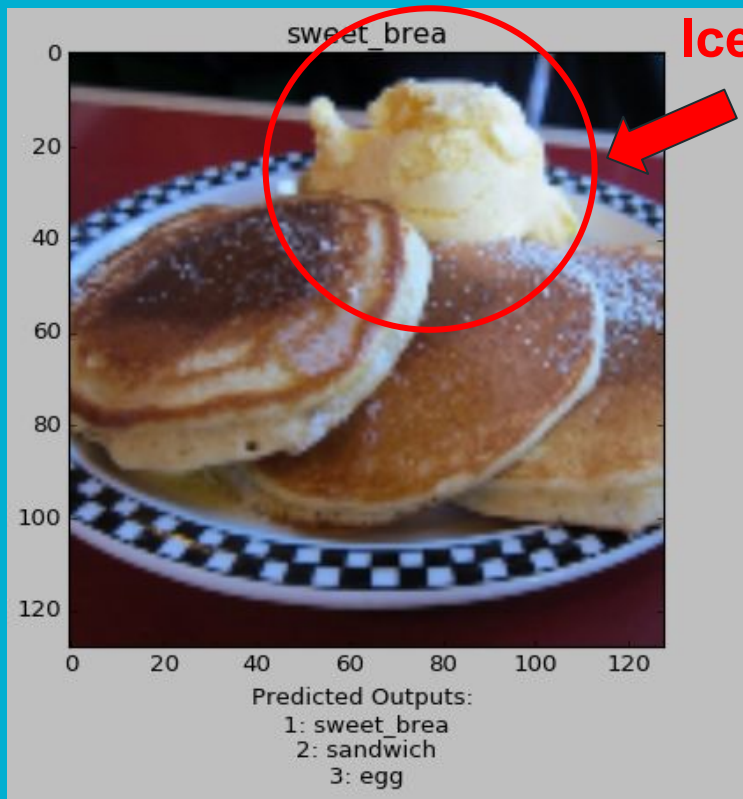
## Confusion Matrix

	Smooth Dessert	Red Meat	Egg	Pasta	Soup	Salad	Fried food	Sandwich	Noodles	Cake	Sweet Breakfast	Shell
Smooth Dessert	290	16	20	12	21	3	20	33	10	79	24	10
Red Meat	51	533	45	42	11	10	24	71	19	67	31	59
Egg	19	29	427	54	17	40	25	68	31	23	24	25
Pasta	24	27	90	795	44	33	100	80	78	29	62	35
Soup	15	13	19	67	618	1	9	21	37	12	25	12
Salad	3	8	9	17	1	244	3	13	13	7	2	8
Fried Food	20	27	29	52	14	4	660	135	19	15	46	12
Sandwich	22	38	47	29	11	19	68	591	19	37	36	8
Noodles	7	19	21	48	24	24	11	20	351	1	5	12
Cake	108	48	30	21	12	9	29	68	6	655	52	32
Sweet Breakfast	25	24	48	41	14	3	36	84	7	50	273	29
Shell	16	18	15	22	13	10	15	16	10	25	20	358

# Conclusion (continued)



# Potentially Confusing Images



Ice Cream???

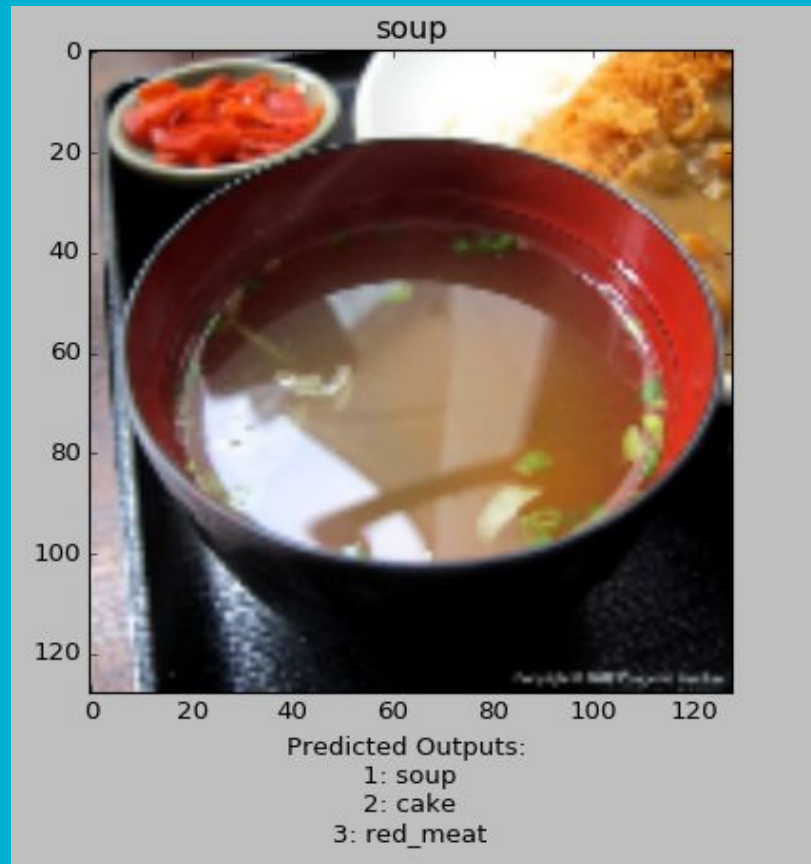


# Conclusion (continued)

---

## Soup

- 77% accuracy for this category
- Most “uniform” food class in our dataset
  - Typically consists of dark liquid in a circular bowl
- The image pictured clearly displays a bowl of soup centered in the frame



# Conclusion (continued)

---

- Achieving ~60% accuracy significant improvement over ~8-9% (12 classes) with random guess
- Concept of food is such that an item cannot be accurately defined or classified by appearance alone, dooming our approach from the start
- Many images contained more than one menu item, potentially of multiple classifications. (may need preprocessing to instruct the network which object(s) to focus on)
- Some of our target classes (such as pasta) were constructed by combining visually distinct categories (gnocchi, spaghetti, lasagna, macaroni and cheese, etc take on many different shapes and colors), negatively affecting the network's ability to recognize patterns and features within a class

# Future Work

---

- **Images Pre-Processing:**

- **Food image:** Color; Outline; Texture; Padding before resize keep the shape same proportion; Random cropping and flipping
- **Tools:** SIFT (Scale-Invariant Feature Transform) / Open CV
- **Approach:** Outline of the food Color Descriptor; Extract effective information (take out the plate or table in the image)

- **CNN parameters:** Kernel size (smaller like 5 or 3); Number of Kernels(from high to low); Batch Normalization

- **Post-Processing:**

- work on the lowest accuracy category (relabel or go)
- More than 1 class in one picture

# Reference

---

- Python Script for illustrating CNN

[https://github.com/yu4u/convnet-drawer/blob/master/convnet\\_drawer.py](https://github.com/yu4u/convnet-drawer/blob/master/convnet_drawer.py)