DETECTING FOOD IN IMAGES

Sidharth Mudgal sidharth@cs.wisc.edu
Li Liu lliu262@wisc.edu
Dongning Wang dw@cs.wisc.edu
Jinman Zhao jzhao237@wisc.edu

ABSTRACT

People nowadays tend to give a lot of attention to food and health care in general. As a result, there are several situations when we need to detect whether an image contains food. In this project, we tried to use several well-known machine learning algorithms to solve the problem of detecting food in images and to figure out which algorithm yields the best results and why. We also tried to use a feature detection technique SIFT, to extract significant features in those images to aid the accuracy of some of our algorithms. The results show that Convolutional Neural Networks work best with accuracies at about 95.4%. However, using SIFT to detect features for our models did not help.

Keyword: Food image detection, Neural Networks, Feature Detection, SVM

1. INTRODUCTION

People nowadays tend to give a lot of attention to food and health care in general. Detecting food in images would be the first step for many fitness applications. For instance, consider a system to automatically analyze how healthy we eat, based on the photos we take. In all these cases, we need a technique to identify whether a given image contains foods.

With this requirement, building a machine learning model to recognize images containing food would be our best bet, considering the scale of the problem. So our question now becomes which algorithm would be the most effective and why.

Since there is a multitude of algorithms that can be used to approach the problem, we narrowed our focus to the most well-known ones. We tried using variants of Single Layer Perceptron (SLP) [1], Multiple Layer Feed Forward Perceptron (MLP) [2], Support Vector Machine (SVM) [3] and Convolutional Neural Network (CNN) [4]. We also went further and tried enhancing our models by using SIFT [10, 11] features and dropout in certain learning algorithms to improve the prediction accuracy. The results show that adding hidden layers and using convolutional layers yield significant improvements, using dropout produces some improvements and that features detected by SIFT are not useful for this task.

The rest of this paper is organized as follow: Section 2 describes what data we used and how we used them. Section 3 explains the proposed method. Section 4 shows how we did the experiment. Section 5 we present the experimental results, and in Section 6 we analyze our results and draw conclusions.

2. DATASET

To get an unbiased mix of data, we combined UECFOOD100, STL-10 and CVPR-09 for training and testing in this paper. The images are of different sizes, we used a thumbnail function to resize and crop them into 96x96 pixels. Namely, we rescale the image so that its width or height, whichever is smaller, into 96 pixels, and crop out the centered 96x96 square.

This method almost always preserves foods in the images according to our experience. There are in total 24000 images in the training set and 4000 images in the testing set. The detailed composition is listed as following.

2.1 Food Images

UECFOOD100 contains 100 categories of food about 14000 images. We took 6/7 images out of each category and put them into our training dataset as positive data.

2.2 Non-food Images

To avoid bias from lighting condition, we select 7000 indoor and 5000 outdoor images.

2.2.1 Outdoor Images

STL-10 has 5000 training images, which are outdoor views or animals. We used all of them.

2.2.2 Indoor Images

CVPR-09 has 67 indoor categories and 15620 images. We removed 13 categories such as bar, buffet, bakery and others, which may contain foods. From the rest we randomly chose 7000 images and use them as negative data.

3. METHODS

3.1 Basic Neural Network

3.1.1 Single Layer Perceptron

We decided to start from the most basic neural network, the single layer perceptron. Single layer perceptron is a kind of neural network that only has two layers: an input layer and an output layer. Two layers are fully connected. It has no hidden layer.

For output layer, in order to figure out what help improving the accuracy, we decided to adopt softmax [5] in SLP as the same as in CNN which is more effective with two output units. We used the cross-entropy cost function [7] as our objective function.

3.1.2 Multi-Layer Feed Forward Perceptron

After we tested the SLP, we added 2 hidden layers, with sigmoid [8] activation functions, to form a Multi-Layer Feed Forward Perceptron, using backpropagation algorithm to train the network. We also tried using 3 hidden layers. Further, we used Nesterov momentum [9] to avoid local minima and to speed up convergence.

Hoping to get better performance and compare to the SVM, we also tried using features detected by SIFT as inputs to the MLP. Since the detected feature vectors are much smaller than original image matrix, we only used two hidden layers with much fewer neurons in this case.

3.2 Feature Detection

We used SIFT (Scale Invariant Feature Transformation) [10, 11] to extract features from the images. SIFT is a widely used method for object recognition. When the same object presents in different images, the object is usually transformed by a combination of translation, scaling, rotation or change of illumination. It is designed to capture key information which is invariant under such transformations. The algorithm starts with converting a given image into gray scale, and then goes through four steps to find a collection of key points and gives numeric vectors to describe each key point.

SIFT consists of four steps. In the 1st step, it computes the differential of Gaussian (DoG) of the grayscale image. Then, the algorithm filters out some candidate key points which are

mis-identified from low contrast area and edges. SIFT will divide 360º into 36 intervals, and construct a histogram by adding up the magnitude of gradient vectors of DoG near the key point location, then find the peak orientation in the histogram in the step 3. At the end, SIFT packs the local information around each key point into 128 numbers which are called the descriptor at the location. These vectors of size 128 can then be used as features.
To predict the labels, we first convert each test image into SIFT vectors and use the model to predict each of them, then determine the label of the test image by majority vote.

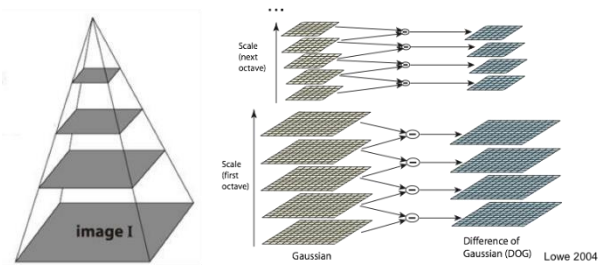

Fig 1: Visualization of step 1 of SIFT algorithm

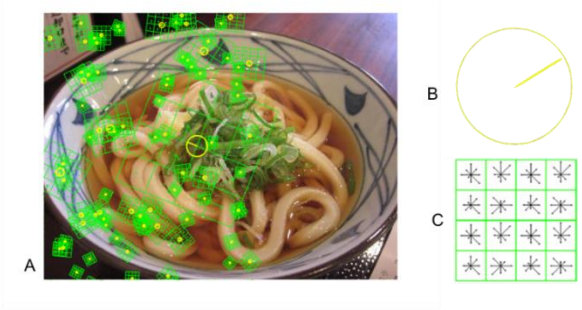

Fig 2: Visualization of SIFT descriptors

3.3 SVM

We use Support Vector Machine (SVM) to compare with neural network based machine learning methods. SVM addresses binary classification by constructing a hyperplane in the vector space of features to best separate feature vectors having different classes (maximizing margin). However, it is not always the case that the data is linearly separable. To address this issue, we make use of soft-margins, which make a trade-off between maximum margins and the degree of misclassification of some training vectors using slack variable. Secondly we use kernels to transform feature vectors into higher- or even infinite- dimensional spaces so that they are easier to separate. We used the radial basis function (RBF, or Gaussian) for our project. Training an SVM requires optimizing a quadratic objective function. This is typically a very computationally expensive process. So in practice, an iterative method is used to obtain an approximate solution.

3.4 Convolutional Neural Network (ConvNet)

A convolutional neural network is a multi layer neural network that uses 3 additional ideas: *local receptive fields*, *shared weights and biases*, and *pooling.*

Local receptive fields: In a regular MLP, each hidden unit is connected to every neuron in the previous layer. However, in a ConvNet, we only make connections within small, localized regions of the previous layer.

Shared weights and biases: Each hidden unit in a layer in a CNN shares the weights and biases (of its local receptive field) with every other neuron in the same layer.

Pooling: A pooling layer in a CNN takes each feature map output from the convolutional layer and prepares a condensed feature map.

We also used dropout [12] for training, which involves training only a random subset of neurons in each iteration. It can helps prevent complex co-adaptations in which a feature detector is only helpful in the context of several other specific feature detectors.

4. EXPERIMENTS

Experiments are designed to first compare different flavors of neural nets applied directly on original images and second to see whether feature extraction helps in enhancing classification accuracy. Finally, an SVM is used to compare it against the neural network based models. Refer to Table 1 for an overview of our experiment design.

	Neural Networks			SVM
	Logistic regression	Regular MLPs	ConvNet	
No Feat. Extraction	Y	Y	Y	N
With Feat. Extraction	N	Y	N	Y

Table 1: Experiment design overview

4.1 Model configurations

We trained the following learning models based on the methods described earlier using package pylearn2. The architecture for ConvNets is written using standard ConvNet architecture notation. For models using SIFT features, we used resized pictures to perform SIFT on their grayscale convertion using MATLAB package VLFeat. Neural Network models 4.1.1 - 4.1.6 were trained using stochastic gradient descent with mini-batch size 100.

4.1.1.Single Layer Perceptron (SLP)

This is our baseline model. It is very similar to logistic regression.

4.1.2.Fully connected Multi-Layer Perceptron with 2 layers (MLP 2L)

An MLP with 9160 neurons in the 1st layer and 1620 neurons in the 2nd layer. We tried adding more neurons in the 1st layer to make it as large as the 3-layer MLP but the validation accuracies turned out to be lower.

4.1.3.Fully connected Multi-Layer Perceptron with 3 layers (MLP 3L)

An MLP with 15000 neurons in the 1st layer, 3700 neurons in the 2nd layer and 2800 neurons in the 3rd layer.

4.1.4.Convolutional Neural Network with 2 layers

A ConvNet with architecture: input-100C5-MP2-170C5-MP2-output

4.1.5.Convolutional Neural Network with 3 layers

A ConvNet with architecture: input-50C5-MP2-170C5-MP2-580C5-MP2-output

4.1.6.Convolutional Neural Network with 3 layers and dropout

A ConvNet with the same architecture as 4.1.5

4.1.7.Fully connected Multi-Layer Perceptron with 2 layers using SIFT features

An MLP with 150 neurons in the 1st layer and 50 neurons in the 2nd layer.

4.1.8.SVM using SIFT features

We used the SVM implemented in scikit-learn with an RBF kernel.

4.2 Tuning hyperparameters

4.2.1 Neural nets

We tried several configurations of each learning model to find the best learning algorithm. Further, since each type of model has several hyperparameters that tend to significantly affect their performance, we used Spearmint, a hyperparameter optimization library to pick them efficiently. Spearmint works by modeling learning algorithms as Gaussian processes.

To pick the best hyperparameters in a reasonable amount of time, we stopped each model after a few number of epochs (1-10 depending on model training time). Although this is not ideal, we considered this as an approximation of the true optimum.
The following parameters were tuned for neural network models:

- Log learning rate (all NNs)
- Log range of initial weights (all NNs)
- Number of neurons in each layer (all NNs)
- Kernel norms (4.1.4-4.1.6) column norms (4.1.1-4.1.3, 4.1.7)
  The norms of each kernel/filter in case of convolutional layers and each column in case of other layers are restricted as a form of regularization.
- Initial weights scale factor (4.1.6):
  The initial weights are multiplied by a scaling factor to accommodate for the dropout

For some hyperparameters we tune their log value instead, since its only necessary for them to be in the right range. Further, the optimizer typically tends to perform poorly otherwise since it gives unnecessary importance to small differences.
We reduce the learning rate by a factor of 10 for the best performing neural network models when they are near convergence, and train them longer to obtain better results. We reduced the rate at the 100th epoch for models 4.1.2-4.1.5 and at the 150th epoch for 4.1.6.

4.2.2 SVM

We tune on the parameter C which trade-off with large margin and small error. Namely bigger C tolerate less violation of the margin, and gives us better training accuracy at the risk of overfitting. To compare with epochs in neural networks, we observe the performance of SVM in the change from the number of iteration steps.

5. RESULTS

5.1 Test error vs epochs plots

To evaluate the models, we plotted graphs of test error vs. epochs for all our models for for dataset sizes (Fig 3 to Fig 10) with the best hyperparameters. Since the SVM is not trained by gradient descent and instead relies on a method which iteratively improves the solution, the graph for SVM uses number of iterations as the x-axis. The figures show how training data size affects learning, how long each model takes to converge and how each model's accuracy changes over time.

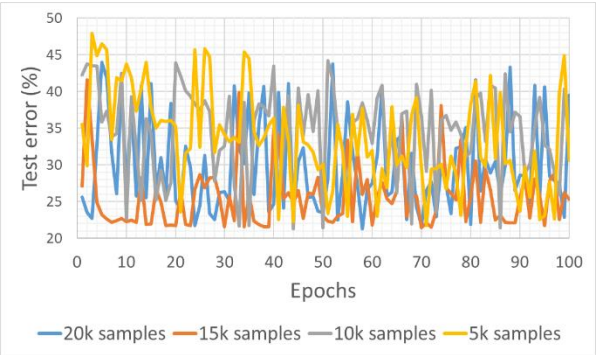

Fig 3: Test error vs epochs for SLP

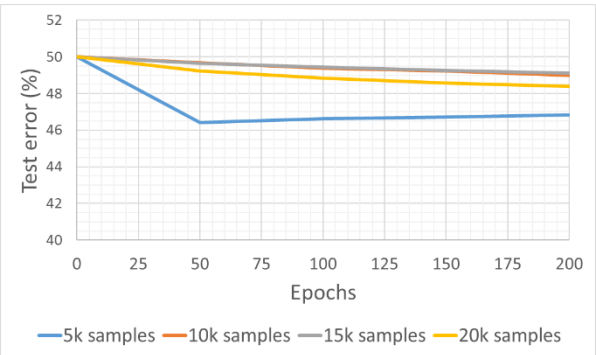

Fig 4: Test error vs epochs for MLP with SIFT

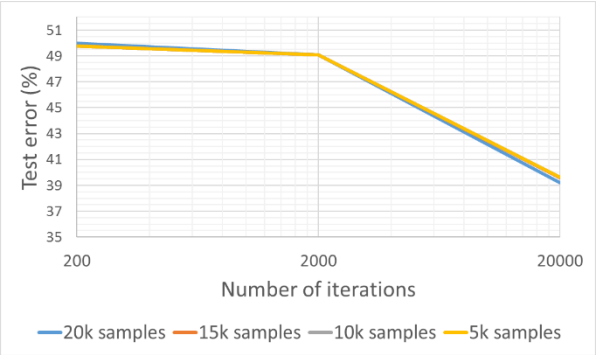

Fig 5: Test error vs number of iterations for SVM with SIFT

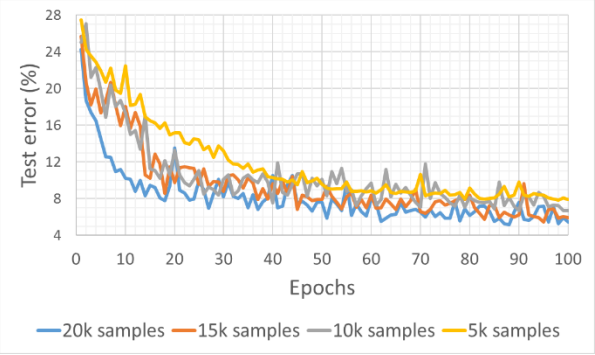

Fig 6: Test error vs epochs for 3 layer ConvNet with Dropout

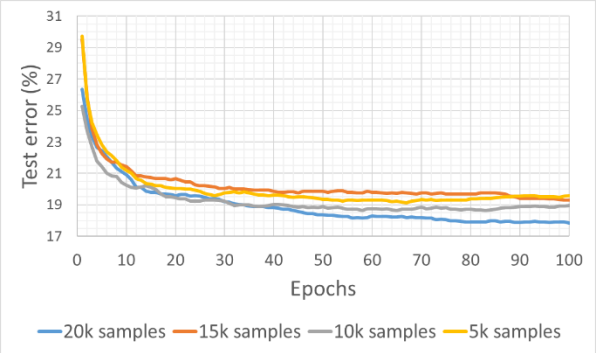

Fig 7: Test error vs epochs for 2 layer MLP

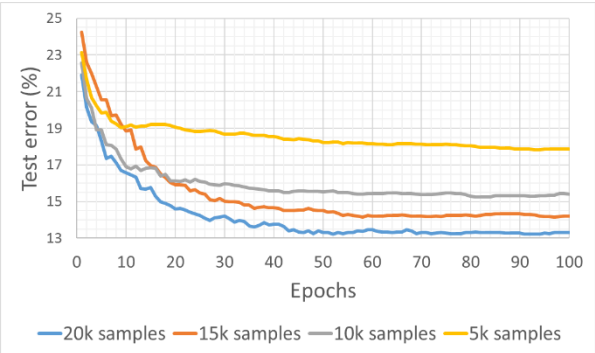

Fig 8: Test error vs epochs for 3 layer MLP

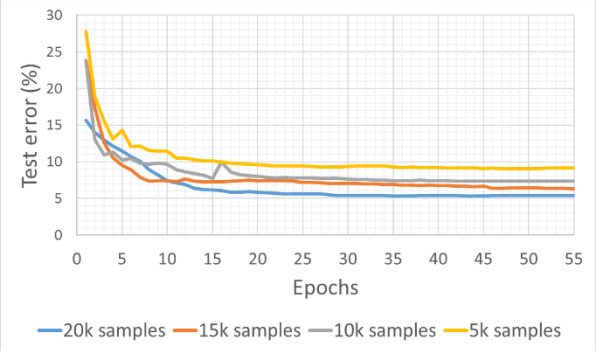

Fig 9: Test error vs epochs for 2 layer ConvNet

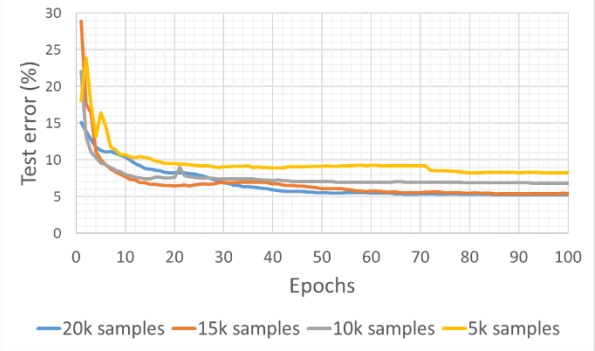

Fig 10: Test error vs epochs for 3 layer ConvNet

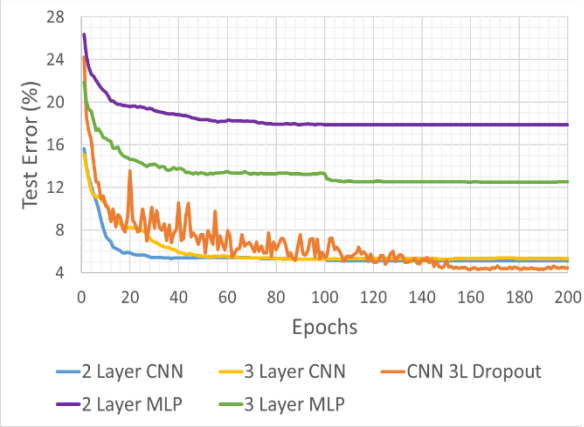

Fig 11: Test error vs epochs for best NN models with 20k instances

Learning Model	Test Error *	Training Error *
Single layer NN (SLP)	21.75	19.27
2 layer MLP w/SIFT	46.82	49.25
SVM with SIFT	39.23	37.34
2 layer MLP	17.85	12.73
3 layer MLP	12.49	0.005
2 layer ConvNet	5.12	0.0
3 layer ConvNet	5.30	0.0
3 layer ConvNet w/dropout	4.57	0.003
* Recorded when the validation error was at its lowest. All values are approx. percentages.		

Table 2: Comparison of test and training errors

5.2 Comparison of all models

In order to have a better perspective of all our models, we plotted the test results of the best models using the entire training set. The test and training errors corresponding to the lowest validation errors are listed in Table 2. The training error for MLP using SIFT features is computed using training images and does not correspond to the training error for the SIFT vectors that the model is actually trained on (which is much lower).

6. ANALYSIS OF RESULTS

6.1 Convergence

We notice that the 2-layer Convolutional neural network converges fastest. We observe a lot of noise in all the error metrics of the SLP model and the 3 layer ConvNet with dropout. In the case of SLP, we assume this is because of the presence of several local minima concentrated in small regions of the objective function since the output neuron is directly connected to the inputs. In case of ConvNet with dropout, the noise is expected since the subset of neurons that are trained changes with each batch. We also observe that adding dropout increases the time to convergence as expected. The training of SVM involves no randomness with convergence guaranteed theoretically. However, practically maximum iteration steps need to be specified to avoid long running time.

6.2 Accuracy

As expected, we notice a significant improvement in test accuracy with standard MLPs and even more in the case of ConvNets as compared to the logistic regression based model.
We notice that the test accuracy of our standard 2 layer MLP improved noticeably (>4%) when an additional layer was added. We speculate that adding more hidden layers would improve the accuracy further. Using convolutional networks increased the accuracy much further (>8%). We notice that adding an extra convolutional layer did not help improve accuracy and assume that this is because the model begins to overfit towards the end, as is evident from the zero training error, negating the benefits of adding another layer. Overfitting tended to occur when we reasonably restricted the norm of the kernels to regularize the network. Adding dropout, in a way, acted as a regularization technique as the training accuracy was no longer zero when using it.
On the other hand, neither SVM nor MLP produces satisfactory results with SIFT features as they produce test errors of about 39% and 46% respectively. This is very likely because by design, SIFT captures invariances under rescale, rotation, and illumination changes. In the case of food recognition, we have much more complex information which cannot be encoded simply as transformations. For instance, two burgers in two images they may not be related to each other by any transformation mentioned above. Yet they are both burgers. In addition, the SIFT descriptors doesn't encode color information around key points at all while color typically conveys a lot of information about the presence of food. Nevertheless, SVM performs better than MLP when using SIFT feature vectors, which implies the elevation of dimensions provided by the RBF kernel function helps.
Notably SVM performs almost the same with different sample sizes. One possible reason is that the feasible feature space is relatively small compared to the number of SIFT vectors in the training set (over 200k even for the 5k samples in image). Since the amount of

information that can be encoded in a 128 dimensional feature space is limited, more training examples may have lesser effect on the resulting boundary as compared to a 96 × 96 dimensional feature space.

6.3 Future Work

For further study, we can use pre-training or other data pre-processing techniques to improve results. Moreover, we can try other enhancements for ConvNets published in recent machine learning conferences.

Besides, we can also broaden the scope of our problem by classifying not only whether or not an image contains food but also detecting what kind of food it contains. Although a more challenging task, food categorizing can be very useful for purposes like determining how healthy your meal is.

7. REFERENCES

[1] B.Widrow, and M.E.Hoff, Adaptive switching circuits, *Proc. Of WESCON Conv. Rec.*, part 4, pp.96-140, 1960

[2] Reed, R. D., & Marks, R. J. (1998). *Neural smithing: supervised learning in feedforward artificial neural networks.* Mit Press.

[3] Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*,*20*(3), 273-297.

[4] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

[5] Bridle, J. S. (1990). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing (pp. 227-236)*. Springer Berlin Heidelberg.

[6] Rubinstein, R. Y., & Kroese, D. P. (2013). The cross-entropy method: a unified approach to combinatorial optimization, *Monte-Carlo simulation and machine learning.* Springer Science & Business Media.

[7] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). *Learning internal representations by error propagation* (No. ICS-8506). CALIFORNIA UNIV SAN DIEGO LA JOLLA INST FOR COGNITIVE SCIENCE.

[8] Han, Jun; Morag, Claudio (1995). "The influence of the sigmoid function parameters on the speed of backpropagation learning". In Mira, José; Sandoval, Francisco. *From Natural to Artificial Neural Computation.* pp. 195–201.

[9] Bengio, Y., Simard, P., and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5:157–166, 1994.

[10] Lowe, David G. (1999). Object recognition from local scale-invariant features. *Proc. 7th International Conference on Computer Vision (ICCV'99)* (Corfu, Greece): 1150-1157.

[11] Lowe, David G. (2004). Distinctive image features from scale-invariant key points. International Journal of Computer Vision 60(2): 91-110.

[12] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1), 1929-1958.