Image Classification Learning Applied to Food

A beginner's attempt to convolutional neural networks

Siqi LIU
CentraleSupélec, France
Campus de Metz
me@siqi.fr

Adrien ROUSSEAU
CentraleSupélec, France
Campus de Metz
adrienjl.rousseau@gmail.com

ABSTRACT

Image classification is a major component of recent machine learning development. Many studies have proven that Deep Neural Networks with large number of parameters can be very powerful machine learning systems. However, most studies in image classification are focused on well-known data sets with well distinguished objects. In this study, we will attempt to deal with real-world images for different types of food and make some suggestions on how to improve the performance of the classification system. Although food classification is intrinsically harder given the similarity between each class, with convolutional neural networks, we proved that food classification is possible even with limited training data. We have also performed analysis on cases where our model performed poorly and attempted to provide a possible explanation, backed with confusion matrix analysis.

1. INTRODUCTION

Deep Neural Networks (DNN) are models inspired by experimental observations made on cat's visual cortex[3]. The experiment made it clear that cat's visual cortex consists of multiple layers of neurons stacked one over the other. While the layers lower in the structure tend to learn basic patterns such as activation of a particular region of the vision field, layers higher up in the architecture manage to recognize more evolved visual patterns such as motions or geometric shapes. While we cannot emulate the exact structure of our brain due to computational limits, we can nonetheless attempt to construct a drastically simplified version of our brain which still captures the essence of biological neural networks. This observation motivated the development of Deep Learning, a field of machine learning which aims at constructing powerful learning systems that extract features that are most helpful in achieving our goals, either classification or regression.

Convolutional Neural Networks (CNN) are a special kind of feed-forward neural networks that take advantages of the geometric topology of the input units and make use of shared weights in convolutional layers. By using convolutional layers combined with pooling techniques, we can considerably reduce the computational costs by limiting the number of parameters to optimize. For a moderately sized 200x200 RGB image, traditional neural network will have a first layer consisted of N neurons, each connected to 200x200x3 pixel in the initial image. Therefore, for the very first input layer of the neural network, we already have Nx200x200x3 weights to learn. This is clearly problematic when adapting the system for larger images. Rather than connecting each neuron to all pixels, CNN takes advantages of the spatial topology of pixels in images and only connect a neuron to a spatially connected pixels. By further sharing weights between each channel, the number of parameters to learn are therefore noticeably reduced, which enabled deeper and more expressive neural networks. CNN has been proved to be powerful techniques for images and video recognition tasks[4]. Details on optimizing the architecture will be discussed in the following.

In the rest of this study, we will focus on the food image data set *UEC-FOOD100*[5] which contains 100 classes of food with a few hundreds of images for each class. The sampled 100 classes are mostly popular food in Asia and more particularly in Japan. Some categories are therefore lesser known in the rest of the world. Meanwhile, typical Asian food are more difficult to differentiate given their particular culinary styles, which make the task even harder. We will mention these difficulties again in our result analysis.

2. SOFTWARE AND HARDWARE

For research purpose, we have used *pylearn2*[2] developed at LISA lab at Université de Montréal. We have used numpy, scipy, pillow, scikit-learn for data and image processing as well as matrix manipulation.

The networks were trained on a GPU workstation provided by CentraleSupélec which has a GeForce GTX 580, 4GB RAM.

3. PREPROCESSING

The *UEC-FOOD100* dataset is formatted as a set of images divided into different classes, each associated with a bounding box specified in a text file. Therefore, each image may contain multiple types of food and the first step of preprocessing is to extract the relevant part of the image. Further, we need to normalize the extracted images before feeding them as input to our Multi-Layer Perceptrons (MLP).

First of all, the images are different in size. Therefore, we had to design a method to normalize their format. The two different ways of doing this are : rejecting data or transforming data. We naturally wanted to avoid rejecting data as much as possible. On the other hand, because we wanted to preserve the visual properties contained in our data, we had to choose carefully what transformation to apply. Because the features in food are highly directional, we decided to only perform uniform scaling on our data. In order to not favor any direction, we chose that the final shape of the image would be a square. So as to keep as much informa-

tion as possible, the final method is then to crop a square as large as possible inside the image containing the food item and to resize it. With this process, unwanted transformations can occur when : the square is too small (in that case the "texture" is lost when the image is resized), or the biggest square included in the bounding box cannot contain enough information to recognize the food inside it (the ratio width/height of the bounding box is very high or very low). We established a criterion based on thresholds to reject such cases. For all the other cases, the biggest square set to be resized is chosen at the center of the image as the most discriminant properties (texture, color, shape) seemed to be at the center. The preprocessing rejects less than 0.4% of the cases.

Secondly, by examining the different classes visually, it was found that several classes were very similar or could be derived from a more general class. The division established in the original dataset seemed sometimes too specific and didn't match our vision of what the network should classify. For instance, we merged the images of 'udon noodle', 'soba noodle', 'ramen noodle' and 'beef noodle' to make a single class : 'noodle soup'. For each class, we tried to make sure that it made sense to merge the class (the visual properties appeared to be similar). The re-organization of the data in new classes reduced the number of classes from 100 to 63 (thus augmented the average number of examples per class by 58%)

4. BASELINE MODEL

We started with a convolutional neural network with 2 convolutional layers followed each by a max-pooling layer, followed by one fully connected layer. The output are computed with a Softmax layer, which normalize the likelihood for each class so that the sum over all classes equals to 1. We can therefore consider 63 outputs from Softmax layer as weighted probabilities for 63 classes accordingly. The Softmax function is given below :

$$P(y = j|\mathbf{x}) = \frac{e^{\mathbf{x}^\top \mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\top \mathbf{w}_k}} \quad (1)$$

Here, we can consider the first two layers as a convolutional feature extractor and the last fully connected layer as a classifier. While it is arguable that two fully connected layers is more expressive, our baseline model with only one hidden fully connected layer already manages to overfit noticeably on training set when trained for extended period. In fact, we can mathematically prove [1] that neural networks with a large number of hidden units and at least one hidden layer are universal approximators. We will first focus on tuning the parameters used in this baseline model and introduce other techniques to combat overfitting.

Our baseline model is given above in the form of Krizhevsky-style diagram (Figure 1).

With preliminary experiments and early stopping on validation set, we have obtained a CNN model of 64 classes with an error rate of 55.72% (**??**) on the test set. Unsurprisingly, we have encountered the problem of overfitting which we will attempt to address in the following.

5. COMBATING OVERFITTING

Undoubtedly the most common problem that challenges every machine learning researcher, overfitting is u Various techniques have since been introduced in order to tackle this problem. As we have observed on our baseline model, the system started to overfit the training set at 55% and then the metrics on the training set and the valid set set out to diverge.

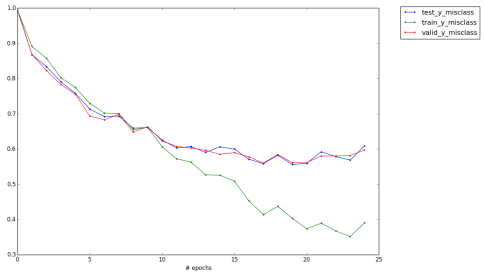

Figure 2 : Misclassification rate for baseline model architecture

Overfitting is particularly problematic when the training set is small whereas the network is sufficiently expressive. In our case, the training set is extremely small which counts for a few hundred examples per class.

In our experiment, we employed various techniques to limit overfitting and therefore, train models that generalize well on the data sets that are unseen during the training process.

5.1 Data Augmentation

The quality of the results obtained thanks to a Multi-Layer Perceptrons, especially with deep learning architectures, is very dependent on the volume of the data used to train it.

The dataset UEC-FOOD100 had been used before with different learning algorithms and provided a good basis to try our own architectures, however we knew that data augmentation was key to get optimal results.

We determined several kind of transformations to augment the volume of the training data. Because the images that we want to feed to the network in the end are ideally any picture of food taken with a smartphone, there are several natural transformations that can be applied to the original images. We implemented a chain of transformation which contained : - rotation (+/- 30 °) - brightening/darkening - image flip (left to right). The data augmentation turned the 10984 images of the dataset used to train the network into 43988 images.

5.2 L2 Regularization

As we have observed, Data Augmentation noticeably mitigated the problem of overfitting. However, we can still observe the effect of overfitting. In order to address this problem, we have attempted to apply regularisation techniques such as L2-regularization which consists of adding a Weight Decay term in the objective function as below. The Weight Decay parameter λ is another hyper-parameter which should be cross-validated over the validation set. Due to limitation on computational budget, we could only try out a few such parameters.

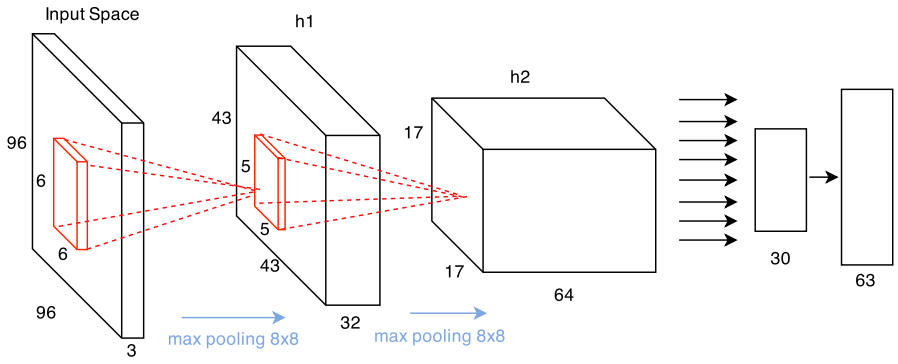

Figure 1 : Baseline model architecture

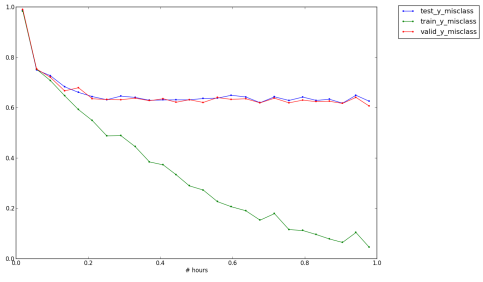

Figure 3 : Misclassification rate for baseline model without L2 Regularization

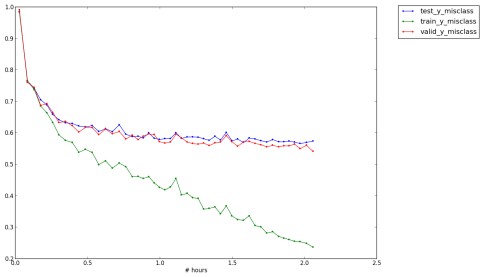

Figure 4 : Misclassification rate for baseline model with $\lambda = 0.005$

$$objective = costs + \lambda \sum_{w} w^2. \tag{2}$$

In fact, in the setting of MLP, we rely on non-linear activation functions to learn complex decision boundaries with neural networks (without which multilayer-perceptrons will be mathematically equivalent to one single layer of perceptrons). Therefore, higher weights are generally signs of overfitting since higher weights will keep the neurons out of their zone of linearity and lead to obscure and twisted decision boundaries which overfit the training and thus prevent the network from generalize. As we can observe below, the first-layer filters are smoother than the filters without L2-regularization (Figure 5) and the model overfit later in the training process (Figure 3 and Figure 4).

5.3 Dropout

Dropout is a powerful technique recently developed by Srivastava and Hinton, [6] which effectively prevents deep networks from overfitting. This technique is particularly useful for small to medium size networks when we don't have enough training data. While this is exactly our case, training with Dropout means that the time it takes for the network to see all images is also increasing.

While we did try to mitigate over-fitting with Dropout, the training process was excessively slow for our computing power and we have only studied this technique theoretically.

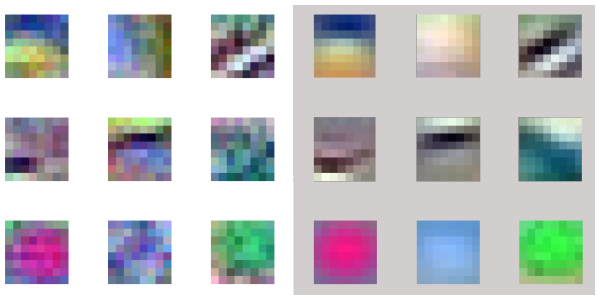

Figure 5 : Filters of the first layer of the network without L2 regularization (left) and $\lambda = 0.005$ (right)

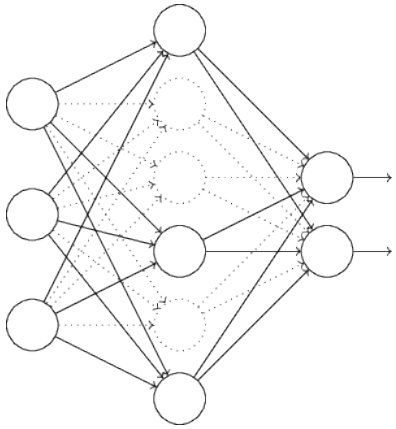

Figure 6 : Dropout : at training time, for each batch, we randomly include only a subset of neurons at each layer.

In essence, Dropout consist of training with not one network structure, but an array of such networks with more or less neurons activated in each layer, according to the *inclusion probability p*, yet another hyper-parameter to optimize. By doing so, we prevent excessive co-adaptation between neurons as one neuron can no longer rely on the output from other neurons. This forces each neuron to learn about a certain pattern on its own, which often results in better weights that generalize to unseen data.

At test time, we no longer exclude neurons but use all of them to make prediction. This is proven to be equivalent to averaging multiple networks with different structure, weighted by their respective *inclusion probability p*, yet eliminate the need to effectively train multiple networks.

5.4 Model Averaging

With *Confusion Matrix* generated from different models, we observe that different models, trained on the same training sets with same parameters, can still perform differently on different classes. This is primarily the result of different weights initialization which led *Stochastic Gradient Descent (SGD)* to optimize for different local or global minima.

Based on confusion matrices, we can analyse the error rate for each model on a per-class basis (diagonal elements in the matrix). This provides an insight on how we can average over different models. Let e_{ij} be the misclass rate for a given model i on class j, we can define the confidence that we have in this model on a given class $c_{ij} = 1 - e_{ij}$.

Suppose that we want to average over K models to classify C classes, we can construct a confidence matrix $C \in K \times C$, we can therefore normalize each column of the C with $C_{ij} = \frac{c_{ij}}{\sum_{j=1}^{K} c_{ij}}$. If the sum of confidence from all the models is zero, this means that none of the model has correctly classified any instance in this class. In this case, the final model will perform poorly, as expected, and we will simply assign equal weights to all models. With an input matrix representing an input image X, the feed-forward function f_i of model i, the final probability of for X to be of class i is therefore defined as :

$$P_i(X) = C \times Y \in K \times 1 \tag{3}$$

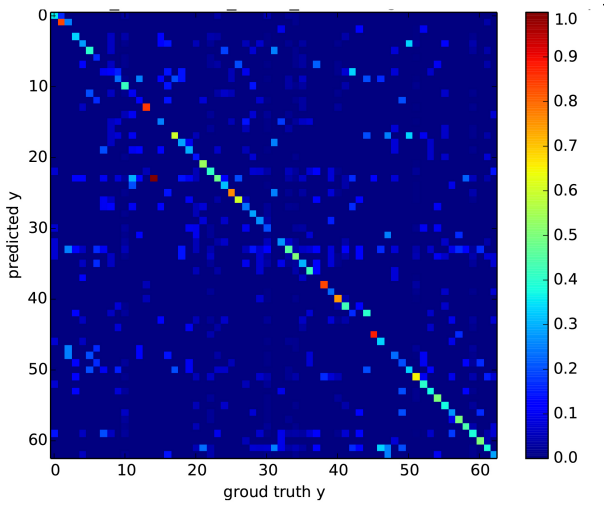

Figure 7 : Confusion Matrix for model #2

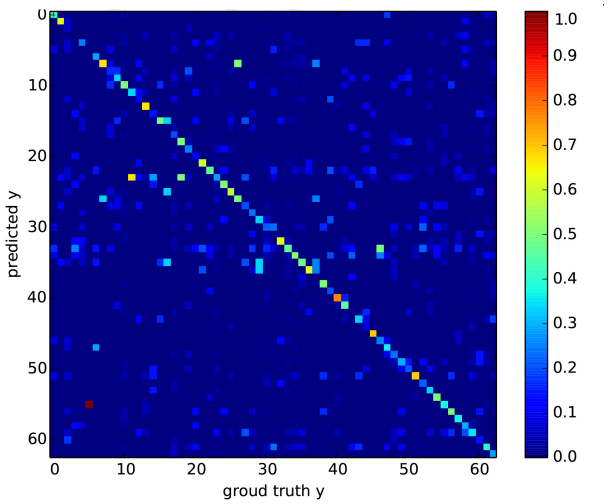

Figure 8 : Confusion Matrix for model #6

where

$$Y_i = f_i(X) \in C \times 1 \tag{4}$$

By averaging over an ensemble of models weighted by their respective confidence on a certain class, the ensemble will make the decision based on a voting mechanism. This approach is later explored in our study.

5.4.1 Experiment

With three different types of network architecture, we have trained 8 models (Tableau 1) with similar overall performance. However, on a per-class basis, different models perform differently as we can tell from the confusion matrices (Figure 7, Figure 8). By applying the weighted vote procedure described above, we have achieved a preliminary result of 0.539 on **test** set. Note that the confidence matrices was generated and evaluated for the valid set, we have therefore made sure that the final result was obtained on previously unseen test set data.

The final confusion matrix for the mixed model is pre-

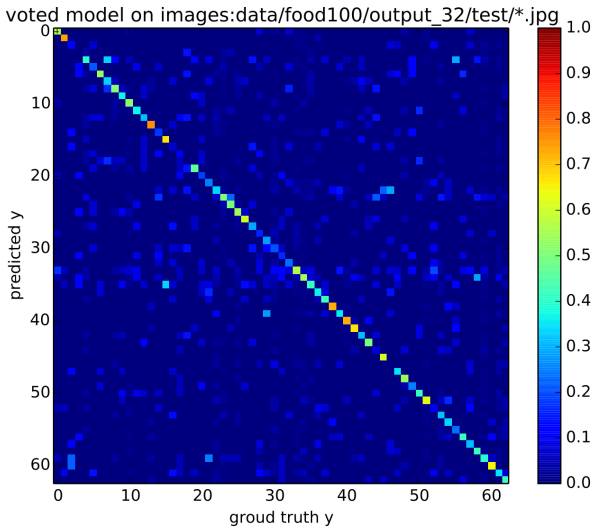

Figure 9 : Confusion Matrix for averaged model

#	valid_error	architecture
1	0.591	(C+ReLU+pooling)*3+ReLU*2
2	0.601	(C+ReLU+pooling)*3+ReLU*2
3	0.585	(C+ReLU+pooling)*3+ReLU*2
4	0.583	(C+ReLU+pooling)*3+ReLU*2
5	0.585	(C+ReLU+pooling)*2+ReLU*2
6	0.582	(C+ReLU+pooling)*2+ReLU*2
7	0.569	(C+ReLU+pooling)*2+ReLU*2
8	0.571	(C+ReLU+pooling)*2+ReLU*2
9	0.615	(C+ReLU+pooling)*2+ReLU

Table 1 : models used in model averaging procedure, with 3 different network architecture

sented on Figure 9. We can see that there are much better performance on certains classes that were previously mostly misclassed.

This experiment is only preliminary as we did not have enough time to carefully cross-validate hyper-parameters for each model and select the best mixture of network architecture. With this approach we are confident that better performance could be achieved.

6. ANALYSIS AND FUTURE WORK

Given the confusion matrix, we can tell that the model performs worst on a few classes yet exceptionally well on others. One such example is the class "pizza". While pizza is visually distinct from many other dishes, one possible explanation to its weak performance is that almost all dishes are presented with rounded light-colored plates. After downsampling, the visual difference between pizza and other dishes are much weaken. Worse even, the visual appearance of pizza is directly related to its flavor : while human can tell the difference between pizza and another dish based on its presentation, it is hard to predict what the topping of pizza as a general class should look like. In the future, it might be interesting to first introduce a binary classifier for pizza, as a general class and further divide pizza based on its topping. These kind of specialist classifier could be useful to improve performance on specific classes.

While we only focused on the TOP-1 misclass rate throughout the study, it is shown that similar classes are more easily confused with each other. Take the example of class "potage", while the class itself was correctly classified at a reasonable rate (68% TOP-1 accuracy), its most confused class is "soup" (21% of examples of "potage" are misclassfied as class "soup"), which is hardly a surprising result. If the correct answer is present among the top 5 choices presented to users, there will be hardly any confusion from the users, if any.

This observation also provides powerful insights on how we could be improve our models on a continuous basis. For example, we could propose the top 5 classes to the end-users and let users choose which class best describe the food. By crowd-sourcing real-world data, we can easily harness human intelligence and continuously improve our classification model with more accurate data that helps generalize the classification network.

7. CONCLUSION

This project is undoubtedly inspiring : we have experienced and encountered many problems that we will be facing in the field of machine learning to which we have envisioned various solutions. Throughout the project, we have experienced first-hand the major challenges in this field and discovered techniques that could be potentially powerful.

Limited data and limited computing power are and always will be major concerns for machine learning researchers and our case is no exception. While our classification performance is far from satisfying, we already have an idea on how to further improve our models continuously.

8. ACKNOWLEDGMENTS

We would like to take this opportunity to thank Dr. Jérémy FIX and Dr. Hervé FREZZA-BUET as well as Mr. Antoine CHASSANG who have been always available for discussions around this project and provide us with guidance and support. We are grateful for the computing ressources made available to us by CentraleSupélec, Campus de Metz throughout the project. Finally, we have been in constant interaction with the Pylearn community led by LISA Lab from Université de Montréal and extensively developed our project around pylearn library, without which this experiment would not have been possible.

9. REFERENCES

[1] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4) :303–314, 1989.

[2] I. J. Goodfellow, D. Warde-Farley, P. Lamblin, V. Dumoulin, M. Mirza, R. Pascanu, J. Bergstra, F. Bastien, and Y. Bengio. Pylearn2 : a machine learning research library. *arXiv preprint arXiv :1308.4214*, 2013.

[3] D. H. Hubel and T. Wiesel. Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex. *Journal of Physiology (London)*, 160 :106–154, 1962.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document

recognition. In *Intelligent Signal Processing*, pages 306–351. IEEE Press, 2001.

[5] Y. Matsuda, H. Hoashi, and K. Yanai. Recognition of multiple-food images by detecting candidate regions. In *Proc. of IEEE International Conference on Multimedia and Expo (ICME)*, 2012.

[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout : A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15 :1929–1958, 2014.