# Transfer Learning for Deep Convolutional Neural Network on Food Image Data


## Abstract

Deep Convolutional Neural Network (CNN) has drawn great attention due to its performance in object recognition tasks. However, we are still not knowledgeable enough to fully understand its principle. Lots of works have been done to discuss the success of AlexNet while for the new GoogLeNet, little work has done to analyze its architecture. We design several transfer learning experiments on food recognition tasks for both AlexNet and GoogLeNet. By analyzing and comparing the results on both architectures, we try to reveal some of the reasons for the success of GoogLeNet. We also show that Deep CNN has strong generalization ability in transfer learning across two food datasets with little overlap categories even though the target set contains just a few labeled instances per category.


## 1. Introduction

Over the recent few years, Convolutional Neural Network (CNN) shows its potential to replace the human engineered features, such as SIFT(Lowe, 1999), SURF(Bay et al., 2006) and HOG(Dalal & Triggs, 2005) etc, in real object recognition tasks. The success of CNN on large scale image set started from Krizhevsky et al(Krizhevsky et al., 2012) and their 8 layer model AlexNet in 2012 ImageNet Large-Scale Visual Recognition Challenge (ILSVRC2012), reaching a on top-5 83% accuracy . Soon after, many attempts have been made to improve the model of Krizhevsky. By reducing the size of the receptive field and stride, Zeiler and Fergus improve AlexNet by 1.7% on top 5 accuracy(Zeiler & Fergus, 2014). With the help of high performances computing systems, such as GPU and large scale distributed clusters, it is possible for researchers to explore larger and more complex architecture. By both adding extra convolutional layers between two pooling layers and reduced the receptive window size, Simonyan and Zisserman built a 19 layer very deep CNN and achieved 92.5% top-5 accuracy(Simonyan & Zisserman, 2014). While the AlexNet-like deep CNNs conquered ILSVRC, Szegedy et al built a 22 layers deep network, GoogLeNet (Szegedy et al., 2014) and won the 1st prize on ILSVRC2014, reaching a astonishing 93.33% top-5 accuracy.

Since these CNN models are trained on very large image data set, they have strong generalization ability and can be applied in many other scenarios. Applying the model pre-trained from ImageNet dataset on other object recognition dataset shows some impressive results. Zeiler et al. applied their pre-trained model on Caltech-256 with just 15 instances per class to fine-tune the model and improved the previous state-of-the-art in which about 60 instances are used for training, by almost 10%(Zeiler & Fergus, 2014). Chatfield et al used their pre-trained model on VOC2007 dataset and outperformed the previous state-of-the-art by 0.9%(Chatfield et al., 2014).

Unlike the local features such as SIFT or SURF, which present an intuitive interpretation of spatial property that is invariant with some transformations such as scaling and rotation, we still don't have enough knowledge to understand the visual features of CNN learned in each layer. Training a large deep CNN on real recognition problem is always a complicated task. The model contains hundreds of millions of parameters to learn and lots of hyper-parameters that can affect its performance. However, the truth that deep CNN outperforms other shallow models by a large margin in some real image recognition tasks encourages researchers to build deep architecture with powerful high performance hardware and larger datasets. With the help of high performance GPU clusters and data argumentation, people are more enthusiastic to explore bigger network on complex recognition problems without much interpretation which makes other researchers difficult to follow.

In this paper, we try to apply the two kinds of deep CNN model, AlexNet and GoogLeNet, on a specific real learning problem, food recognition, and try to reveal some tricks in fine-tuning the existing CNN architecture on this problem. To our best knowledge, no one has deeply studied the architecture of GoogLeNet while the architecture of AlexNet has been widely discussed. By comparing some statistics of the weights and neuron responses of these two archi-


Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

tectures, we also get some ideas why GoogLeNet is more efficient. Also, we conduct several experiments to stimulate a real world situation when the training labeled data is rare. The results reveal that Deep CNN could work well while transferring knowledge from general recognition task to specific one in this scenario.

The rest of this paper is organized as follow: in Section 2, the two food image datasets and two deep CNN architectures are introduced. In Section 3, some experimental results are shown and we also compare the performance between the deep CNNs as well as some traditional methods on these two datasets. And some discussion of the Inception's architecture and statistics are shown in Section 3. We also show some fine-tuning results when the training examples are rare for each class.

## 2. Experimental Setup

In this section, we will discuss some details about the datasets and architectures used for our experiments.

### 2.1. Models

In this paper, AlexNet and GoogLeNet are their Caffe(Jia et al., 2014) implementation and all the results for a specific CNN architecture are obtained from single model.

**AlexNet** contains 5 layers followed by the auxiliary classifier which contains 2 fully connected layers (FC) and 1 softmax layer. Each of the first two layers can be subdivided into 3 components: convolutional layer with rectified linear units (ReLUs), local response normalization layer (LRN) and max pooling layer. Layer 3 and layer 4 contain just convolutional layer with ReLUs while layer 5 is similar to the first two layers except for the LRN. For each of the fully connected layer, 1 ReLUs and 1 dropout(Srivastava et al., 2014) layer are followed.

**GoogLeNet** shows another trend of deep CNN architecture with lots of small receptive fields. Figure 1 shows the architecture a inception cell. Inspired by (Lin et al., 2013), lots of $1 \times 1$ convolutional layers are used for computational efficiency. Another interesting feature of GoogLeNet is that there are two extra auxiliary classifiers in intermediate layers. During the training procedure, the loss of these two classifiers are counted into the total loss with a discount weight 0.3, in addition with the loss of the classifier on top. More architecture details can be found from (Szegedy et al., 2014).

### 2.2. Food Datasets

Besides ImageNet dataset, there are many popular benchmark datasets for image classification tasks such as Caltech dataset and CIFAR dataset, which contain hundreds

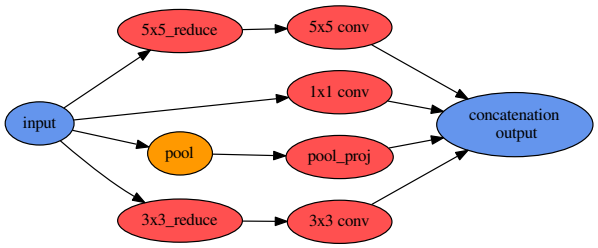


*Figure 1.* Inception Cell. $n \times n$ stands for size $n$ receptive field, $n \times n\_reduce$ stands for the $1 \times 1$ convolutional layer before the $n \times n$ convolution layer and $pool\_proj$ is another $1 \times 1$ convolutional layer after the MAX pooling layer. The output layer concatenate all its input layers.

of classes. However, in this paper, we try to focus on a more specific area, food classification. Compared to other classification tasks, there are some properties of the food (dishes) which make the tasks become a real challenge: i)food doesn't have any distinctive spatial layout: for other tasks like scene recognition, we can always find some discriminative features such as buildings or trees, etc. ii) food class is a small sub-category among all the categories in daily life, so the inter-class variation is relatively small; on the other hand, the contour of a food varies depending on many aspects such as the point of the view or even its component. So these properties make food classification catastrophic for some recognition algorithms. Therefore, the training these two models on food classification task can reveal some important aspects of themselves and help us better understand their architectures. In this paper, we used two image datasets Food256(Kawano & Yanai, 2014)[1] and Food101(Bossard et al., 2014)[2]. It is worthy to mention that PFID dataset is also a big public image database for classification, but their images are collected in a laboratory condition which is considerably not applicable for real recognition task.

**Food-256 Dataset.** This is a relatively small dataset containing 256 kinds of foods and 31644 images from various countries such as French, Italian, US, Chinese, Thai, Vietnamese, Japanese and Indonesia. The distribution among classes is not even and the biggest class (vegetable tempura) contains 731 images while the smallest one contains just 100 images. For this "small" dataset, we randomly split the data into training and testing set, using around 80% (25361 images) and 20%(6303 images) of the original data respectively and keep the class distribution in these two sets

[1]Dataset can be found http://foodcam.mobi/dataset.html

[2]Dataset can be found http://www.vision.ee.ethz.ch/datasets_extra/food-101

uniform. The collector of this dataset also provides boundary box for each image to separate different foods and our dataset is cropped according to these boundary boxes.

**Food-101 Dataset.** This dataset contains 101-class real-world food (dish) images which were taken and labeled manually. The total number of images is 101,000 and there are exactly 1000 images for each of the class. Also, each class has been divided into training and testing set containing 750 images and 250 images respectively by its collector. The testing set is well cleaned manually while the training set is not well cleaned on purpose. This noisy training set is more similar to our real recognition situation and it is also a good way to see the effect of the noise on these two architectures.

Data argumentation is an efficient way to enrich the data. There are also some techniques that can applied to enlarge the dataset such as *subsampling* and *mirroring*. The original images are firstly resized to $256 \times 256$ pixels. We crop the 4 corners and center for each image according to the input size of each model and flap them to obtain 10 crops. For the testing set, the prediction of an image is the average prediction of the 10 crops.

## 3. Experimental Discuss

Training a CNN with millions of parameters on a small dataset could always lead to horrible overfitting. But the idea of supervised pre-training on some huge image datasets could preventing this problem in certain degree. Compared to other initialized strategies according to certain distributions, the pre-trained model is initialized according to the distribution of the specific task. Indeed, this initialization has certain bias as there is no single dataset including all the invariance for natural images(Agrawal et al., 2014), but this bias could be reduced as the pre-trained image dataset increases and the fine-tuning can be benefit from this initialization.

### 3.1. Pre-training and Fine-tuning

We conduct several experiments on both architectures and use different training initialization strategies for both Food-256 and Food-101 datasets. The scratch models are initialized with Gaussian distribution for AlexNet and Xavier algorithm(Glorot & Bengio, 2010), which automatically determines the scale of initialization based on the number of input and output neurons. These two initializations are used for training the model for the original ImageNet task. The pre-trained models and fine-tune models are initialized with the weights trained from ImageNet. For the pre-trained models, we just re-train the softmax layers while all the layers are re-trained or fine-tuned for the fine-tune models.

*Table 1.* Top-5 Accuracy in percent on fine-tuned, pre-trained and scratch model for two architectures

| | AlexNet | | GoogLeNet | |
|---|---|---|---|---|
| | Food-101 | Food-256 | Food-101 | Food-256 |
| Fine-tune | **88.12** | **85.59** | **93.51** | **90.66** |
| Pre-trained | 76.49 | 79.26 | 82.84 | 83.77 |
| Scratch | 78.18 | 75.35 | 90.45 | 81.20 |

From Table 1 we can see that fine-tune from pre-trained model can boost the performance of the CNN for a specific task. Compared to other traditional computer vision methods, GoogLeNet and AlexNet improved at least 27.35% and 15.64% respectively. Considering about the noisy images in Food-101 dataset, this 78.11% accuracy using fine-tuned GoogLeNet is a really competitive one compared to any other method and it is the state-of-the-art performance of this dataset. Because no one has shown any classification result on Food-256 dataset, our result could be a competitive baseline for this dataset.

In Figure 2 we visualized the responses of the pre-trained GoogLeNet model and fined-tuned GoogLeNet model for the same input image for some layers. We can see that the responses of the lower layer are similar as the lower level features are similar. Then we can see that the decisions made by these two models is totally different. Since only the last layer (auxiliary classifier) of the pre-trained model is optimized, we can infer that the higher level features are more important which is consistent with our intuition. Also from Table 1, it is interesting to see that for the Food-101 task, the accuracy of the scratch models outperforms the pre-trained models. Since Food-101 is a relatively large dataset with 750 images per class, this indicates that if the data is sufficient, both CNN can learn proper high level features with some random initialization strategies.

From Table 1 we can see that GoogLeNet always performances better than AlexNet on both datasets. This implies that the higher level features of GoogLeNet are more distinctive compared to AlexNet and this is due to the special architecture of its basic unit, Inception. Table 3 and 4 show the weights' cosine similarity of each layer between the fine-tuned models and their pre-trained models. From the results we can see that the weights in the low layer are more similar which implies that these two architectures can learn the hierarchical features as the low level features are similar for most of the tasks and the difference of the objects is determined by the combination of these low level features. From Table 4, we can see that, in AlexNet the weights of the pre-trained and fine-tuned models are extremely similar. This can be caused by two reasons:

- Size of receptive filed. Since ReLUs are used in Both

*Table 2.* Accuracy compared to other methods on Food-101 dataset

| | RFDC(Bossard et al., 2014) | MLDS(≈(Singh et al., 2012)) | GoogLeNet | AlexNet |
|---|---|---|---|---|
| Top1 accuracy | 50.76% | 42.63% | **78.11%** | 66.40% |

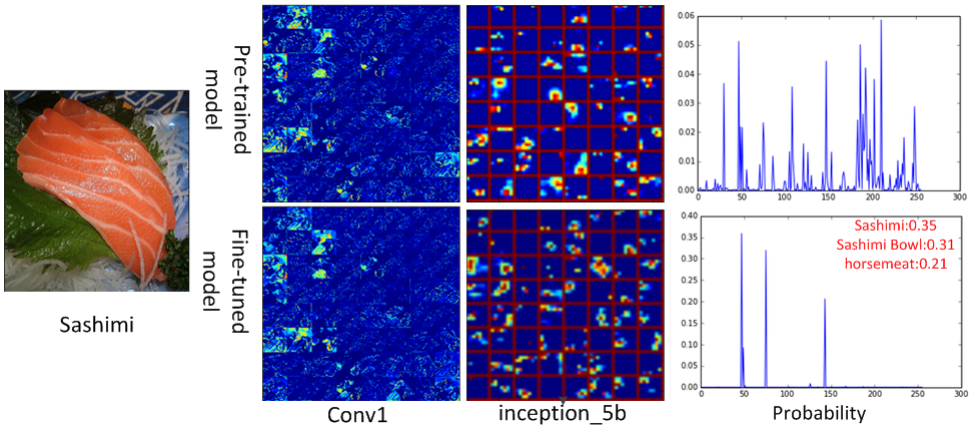


*Figure 2.* Visualization of some responses of different GoogLeNet models in different layers for the same input image. 64 neuron responses of each layer are shown. Conv1 is the first convolutional layer and Inception_5b is the last convolutional layer.

architectures, vanishing gradients do not exist. Rectified activation function is mathematically given by

$$h = \max(w^T x, 0) = \begin{cases} w^T x & w^T x > 0 \\ 0 & else \end{cases} \quad (1)$$

The ReLU is inactivated when its input is below 0 and its partial derivative is 0. Sparsity can boost the performance of the linear classifier on top, but on the other hand, sparse repesentation will make it more difficult to train as well as fine-tuning. The derivative of the filter is $\frac{\partial J}{\partial w} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial w} = \frac{\partial J}{\partial y} * x$ where $\frac{\partial J}{\partial y}$ denotes the partial derivative of the activation function, $y = w^T x$ and $x$ denotes the inputs of the layer. The sparseness of the input could lead to sparse filter derivative for back propagation. Therefore, the filters of the fine-tuned AlexNet is extremely similar. Compared to large receptive field used in AlexNet, the inception in GoogLeNet employs 2 additional $n \times n\_reduced$ convolutional layers before the $3 \times 3$ and $5 \times 5$ convolutional layers (see Figure 1). Even though the most critical purpose of these two $1 \times 1$ convolutional layer is for computational efficiency, these 2 convolutional layers tend to squeeze the sparse input and generate a dense outputs as the input of the next layer.

- The pooling strategy. In AlexNet, max pooling is applied to all the pooling layers between several convolution layers and in back propagation, the max pooling layer will pass the error to the place where it came from. Since it only came from one place of the receptive field, the back propagation error is sparse and it keeps the filters of the convolution layers stable. In GoogLeNet, even though, there is a max pooling layer in every inception, there are other 3 back propagation errors, from $5 \times 5\_reduce$ and $3 \times 3\_reduce$ that can affect the weights of the previous inception.

### 3.2. Training across the datasets

From the previous experiments we can see that pre-training on the ImageNet dataset can boost the performance of the deep convolutional neural network and the knowledge learned from the general recognition problem can be successfully transferred into our specific area. In this part, we will discuss the generalization ability within the food recognition problem. Zhou et al. trained AlexNet for Scene Recognition across two datasets with identical categories(Zhou et al., 2014). But for more complex situation, such as two similar datasets with a little overlapped categories, we are very interested in exploring whether Deep CNN can still successfully handle. Therefore, we conduct the following experiment to stimulate a more complex real world problem: transferring the knowledge from the fine-tuned Food-101 model on a target set, Food-256 dataset and continue fine-tune the model on it. To make the experiment more practical and complex, we limit the number of samples per category from Food-256 for training, because in practise, if we want to build a our own model from Deep CNN, the resource is limited and it is exhausted to collect hundreds of labeled images for each category.

The Food-101 and Food-256 datasets share about 46 categories of food even though the images in the same cat-

*Table 3.* Cosine similarity of the inceptions between fine-tuned models and scratch model for GoogLeNet

| | 1x1 | 3x3_reduce | 3x3 | 5x5_reduce | 5x5 | pool_proj |
|---|---|---|---|---|---|---|
| **food256** | | | | | | |
| inception_3a | 0.72 | 0.72 | 0.64 | 0.67 | 0.73 | 0.69 |
| inception_3b | 0.59 | 0.64 | 0.53 | 0.70 | 0.60 | 0.56 |
| inception_4a | 0.46 | 0.53 | 0.54 | 0.50 | 0.67 | 0.38 |
| inception_4b | 0.55 | 0.58 | 0.63 | 0.52 | 0.69 | 0.41 |
| inception_4c | 0.63 | 0.64 | 0.63 | 0.57 | 0.68 | 0.52 |
| inception_4d | 0.60 | 0.62 | 0.60 | 0.58 | 0.68 | 0.50 |
| inception_4e | 0.60 | 0.61 | 0.67 | 0.61 | 0.68 | 0.50 |
| inception_5a | 0.51 | 0.53 | 0.58 | 0.48 | 0.60 | 0.39 |
| inception_5b | 0.40 | 0.44 | 0.50 | 0.41 | 0.59 | 0.40 |
| **food101** | | | | | | |
| | 1x1 | 3x3_reduce | 3x3 | 5x5_reduce | 5x5 | pool_proj |
| inception_3a | 0.71 | 0.72 | 0.63 | 0.67 | 0.73 | 0.68 |
| inception_3b | 0.56 | 0.63 | 0.50 | 0.71 | 0.60 | 0.53 |
| inception_4a | 0.43 | 0.50 | 0.50 | 0.47 | 0.62 | 0.36 |
| inception_4b | 0.48 | 0.52 | 0.57 | 0.50 | 0.67 | 0.35 |
| inception_4c | 0.57 | 0.61 | 0.59 | 0.53 | 0.63 | 0.47 |
| inception_4d | 0.54 | 0.58 | 0.53 | 0.54 | 0.64 | 0.44 |
| inception_4e | 0.53 | 0.54 | 0.61 | 0.55 | 0.62 | 0.42 |
| inception_5a | 0.43 | 0.47 | 0.53 | 0.45 | 0.57 | 0.34 |
| inception_5b | 0.36 | 0.39 | 0.46 | 0.38 | 0.52 | 0.37 |

*Table 4.* Cosine similarity of the layers between fine-tuned models and scratch model for AlexNet

| | conv1 | conv2 | conv3 | conv4 | conv5 | fc6 | fc7 |
|---|---|---|---|---|---|---|---|
| food256 | 0.997 | 0.987 | 0.976 | 0.976 | 0.978 | 0.936 | 0.923 |
| food101 | 0.996 | 0.984 | 0.963 | 0.960 | 0.963 | 0.925 | 0.933 |

*Table 5.* Sparsity of the output for each unit in GoogLeNet inception for training data from Food101 in percent

| | 1x1 | 3x3_reduce | 3x3 | 5x5_reduce | 5x5 | pool_proj |
|---|---|---|---|---|---|---|
| inception_3a | $69.3 \pm 1.3$ | $69.6 \pm 1.1$ | $80.0 \pm 1.0$ | $64.1 \pm 2.2$ | $75.8 \pm 1.6$ | $76.2 \pm 5.4$ |
| inception_3b | $92.8 \pm 0.9$ | $76.5 \pm 0.9$ | $94.7 \pm 0.9$ | $71.6 \pm 2.3$ | $94.4 \pm 0.5$ | $94.7 \pm 1.6$ |
| inception_4a | $90.9 \pm 0.9$ | $70.0 \pm 1.2$ | $93.8 \pm 1.1$ | $63.3 \pm 4.0$ | $91.9 \pm 1.8$ | $95.1 \pm 2.0$ |
| inception_4b | $71.9 \pm 1.6$ | $67.5 \pm 1.2$ | $75.4 \pm 1.0$ | $58.5 \pm 2.6$ | $78.9 \pm 1.6$ | $85.6 \pm 3.6$ |
| inception_4c | $75.1 \pm 2.4$ | $72.6 \pm 1.3$ | $81.0 \pm 2.0$ | $66.3 \pm 6.1$ | $79.7 \pm 3.6$ | $88.1 \pm 3.3$ |
| inception_4d | $87.3 \pm 2.7$ | $78.0 \pm 2.2$ | $88.0 \pm 1.6$ | $67.9 \pm 3.1$ | $88.9 \pm 2.8$ | $93.0 \pm 2.2$ |
| inception_4e | $91.8 \pm 1.1$ | $62.3 \pm 2.2$ | $91.0 \pm 2.5$ | $49.5 \pm 3.7$ | $94.0 \pm 1.0$ | $92.3 \pm 1.5$ |
| inception_5a | $78.7 \pm 1.6$ | $66.5 \pm 1.7$ | $82.3 \pm 2.6$ | $59.9 \pm 3.2$ | $86.4 \pm 2.3$ | $87.1 \pm 2.6$ |
| inception_5b | $88.2 \pm 2.3$ | $86.8 \pm 1.6$ | $83.3 \pm 4.4$ | $84.0 \pm 3.1$ | $81.4 \pm 5.3$ | $94.7 \pm 1.5$ |

egory may vary across these two datasets. The types of food in Food-101 are mainly western style while most types of food in Food-256 are typical Asian foods. We compared the top-5 accuracy of different size of subset for Food-256 on different pre-trained model and the results are shown in Table 6. The ImageNet columns denote the pre-trained model trained only on ImageNet images and the Food101_ft columns denote the pre-trained model trained on ImageNet images and then fine-tuned on Food-101.

From the result of Table 6 we can see that, with this second round of transfer learning, both CNNs can achieve around 95% of the accuracy trained on full dataset while just using about half of them (50 per class, 12800 of 25361 images). This indicates that when there is not enough labeled data, with its strong generalization ability, Deep CNN trained from general task can still achieve satisfying result and perform even better when an additional relevant dataset is involved. This encouraging result can attract more people to use Deep CNN for their specific task and continue to explore the potential of the existing architecture as well as designing new ones.

## 4. Conclusion

In this paper, we compared two different deep convolutional neural network architectures and their transferring ability on food datasets. Both architectures have shown their potential on generalization ability and we provide the state-of-the-art using fine-tuned GoogLeNet on Food-101 dataset, the winner of ILSVRC2014, shows its great ability on transferring the knowledge between different tasks with the help of the special designed unit, Inception. Intensively used $1 \times 1$ convolutional layers in Inception reduces both computational cost and training complexity which leads to the final success of the whole architecture. Moreover, in a more practical situation such as transfer learning within a specific area with a few labeled instances for the target set, both of these two architectures show some encouraging result in transferring knowledge across the dataset and could encourage people to explore more potential on Deep CNN.

## Acknowledgments

## References


Agrawal, Pulkit, Girshick, Ross, and Malik, Jitendra. Analyzing the performance of multilayer neural networks for object recognition. In *Computer Vision–ECCV 2014*, pp. 329–344. Springer, 2014.

Bay, Herbert, Tuytelaars, Tinne, and Van Gool, Luc. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pp. 404–417. Springer, 2006.

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

Chatfield, K., Simonyan, K., Vedaldi, A., and Zisserman, A. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014.

Dalal, Navneet and Triggs, Bill. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pp. 886–893. IEEE, 2005.

Glorot, Xavier and Bengio, Yoshua. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Jia, Yangqing, Shelhamer, Evan, Donahue, Jeff, Karayev, Sergey, Long, Jonathan, Girshick, Ross, Guadarrama, Sergio, and Darrell, Trevor. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pp. 675–678. ACM, 2014.

Kawano, Y. and Yanai, K. Automatic expansion of a food image dataset leveraging existing categories with domain adaptation. In *Proc. of ECCV Workshop on Transferring and Adapting Source Knowledge in Computer Vision (TASK-CV)*, 2014.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Lin, Min, Chen, Qiang, and Yan, Shuicheng. Network in network. *CoRR*, abs/1312.4400, 2013.

Lowe, David G. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pp. 1150–1157. Ieee, 1999.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Singh, Saurabh, Gupta, Abhinav, and Efros, Alexei A. Unsupervised discovery of mid-level discriminative patches. In *Computer Vision–ECCV 2012*, pp. 73–86. Springer, 2012.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout:

*Table 6.* Top5 Accuracy for transferring from Food101 to subset of Food256

| | AlexNet | | GoogLeNet | |
|---|---|---|---|---|
| instances per class | ImageNet | Food101_ft | ImageNet | Food101_ft |
| 20 | 68.80 | **75.12** | 74.54 | **77.77** |
| 30 | 73.15 | **77.02** | 79.21 | **81.06** |
| 40 | 76.04 | **80.23** | 81.76 | **83.52** |
| 50 | 78.90 | **81.66** | 84.22 | **85.84** |
| all | 85.59 | **87.21** | **90.66** | **90.65** |

A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1): 1929–1958, 2014.

Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, and Rabinovich, Andrew. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

Zeiler, Matthew D and Fergus, Rob. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*, pp. 818–833. Springer, 2014.

Zhou, Bolei, Lapedriza, Agata, Xiao, Jianxiong, Torralba, Antonio, and Oliva, Aude. Learning deep features for scene recognition using places database. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., and Weinberger, K.Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 487–495. Curran Associates, Inc., 2014.