Project Report

Food Recognition using Generative Adversarial Networks

1. Literature Survey

Several existing works do use computer vision algorithms to tackle the food recognition task but only work in laboratory conditions where the food items are well separated and the number of categories is small. Furthermore, most of these methods use traditional, hand-crafted visual features, and only use machine learning at the classification stage.

Deep learning algorithms overcome the drawbacks of traditional machine learning algorithms based on hand designed features. These algorithms have the inherent capability to do so. Many human information processing mechanisms (eg. Speech and vision) have deep hierarchical structures which cannot be modeled accurately by a traditional machine learning algorithm. This is where deep learning comes in as it is inspired by these same deep hierarchical structures which exist in human speech and production systems.

The birth of the deep learning techniques lies in the concept of layer-wise greedy learning. Generally, training a large feed forward net with backpropagation leads to 'vanishing gradient problem', i.e., the lower layers don't actually know how to transform the inputs so as that they are useful to higher layers. In deep learning, greedy layer-wise learning was introduced which could transform the input in lower layers in a greedy, problem-agnostic manner. This transforms the input at every layer in such a manner that it is always useful for the higher layer.

Thus, deep learning algorithms consist of a hierarchical architecture with many layers, each of which constitutes a non-linear information processing unit.

1.1 The Food Recognition Problem

Real world food images have considerable noise in terms of non-uniform background and presence of other objects and multiple food items. This in turn reduces the effectiveness of the traditional recognition methods to a considerable degree. A rough estimation of the region in which targeted food item is present would help to raise the accuracy in such cases. Two approaches have been taken towards this, using standard segmentation and object detection methods or asking the user to input a bounding box providing this information. CNN based semantic segmentation has also been used with significant improvement in accuracy. Apart from this there have been attempts at designing algorithms specifically for localizing food items in an image.

In order to reduce the search space for the classifier, and provide better results, geotagging has been used to identify the restaurant and search for matching food item in its menu. There has been certain progress in using ingredient level features to identify the food item. A variant of this method is the usage of pairwise statistics of local features. In the recent years CNN based classification has shown promise producing excellent results even on extensive and large datasets with non-uniform background.

CNN based classification still has its own shortcomings since the currently available food datasets are generally small in size. This goes against the nature of the CNN method itself which needs large datasets to produce more accurate results. To further increase the accuracy of the problem, Generative Adversarial Methods (GAN's) can be used. Thus, we can get lots of images of food and then use the smaller set of annotated images to train the GAN's and make an accurate classifier.

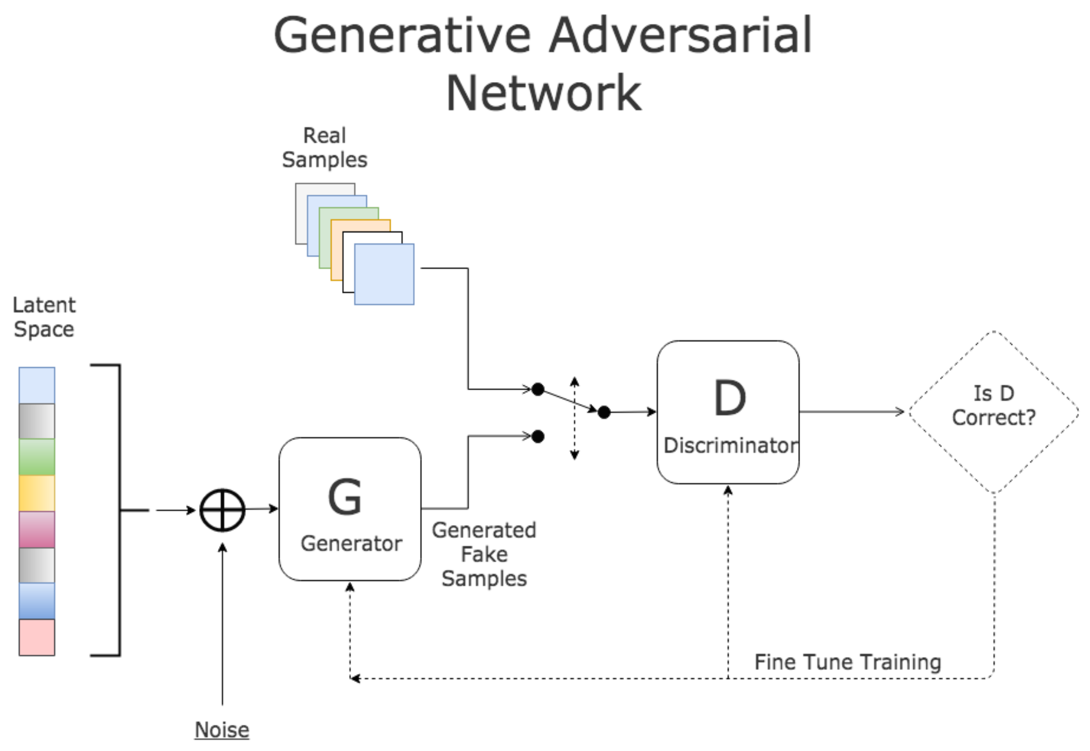
1.2 Theoretical Introduction to GAN's

Learning models are many times classified into Generative and Discriminative models. They can be described as:

- A **discriminative** model learns a function that maps the input data x to some desired output class label y. In probabilistic terms, they directly learn the conditional distribution P(y|x).

- A **generative** model tries to learn the joint probability of the input data and labels simultaneously, i.e. P(x,y). This can be converted to P(y|x) for classification via Bayes rule, but the generative ability could be used for something else as well, such as creating likely new (x, y) samples.

Both types of models are useful, but generative models have one interesting advantage over discriminative models – they have the potential to understand and explain the underlying structure of the input data even when there are no labels. This is very desirable when working on data modelling problems in the real world, as unlabelled data is of course abundant, but getting labelled data is often expensive at best and impractical at worst.



The main idea behind a GAN is to have two competing neural network models. One takes noise as input and generates samples (and so is called the generator). The other model (called the discriminator) receives samples from both the generator and the training data, and has to be able to distinguish between the two sources. These two networks play a continuous game, where the generator is learning to produce more and more realistic samples, and the discriminator is learning to get better and better at distinguishing generated data from real data. These two networks are trained simultaneously, and the hope is that the competition will drive the generated samples to be indistinguishable from real data.

The analogy that is often used here is that the generator is like a forger trying to produce some counterfeit material, and the discriminator is like the police trying to detect the forged items. It can also be thought of as a minimax two player game.

1.3 GAN Algorithm

```
for number of training iterations do
    for $k$ steps do
        • Sample mini batch of m noise samples $(z^1, z^2, ...., z^m)$ from noise prior $p_g(z)$.
        • Sample mini batch of m examples $(x^1, x^2, ...., x^m)$ from data generating distribution $p_{data}(x)$.
        • Update the discriminator by ascending its stochastic gradient.
              $$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [log D(x^i) + log(1 - D(G(z^i)))]$$
    end for
    • Sample mini batch of m noise samples $(z^1, z^2, ...., z^m)$ from noise prior $p_g(z)$.
    • Update the generator by descending its stochastic gradient.
          $$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} [log(1 - D(G(z^i)))]$$
end for
```

2. Dataset Used

ETH Food-101 is a database consisting of 1000 images of 101 classes of most popular food categories picked up from foodspotting.com. The top 101 most popular and consistently named dishes were chosen and 750 training images were extracted. Additionally, 250 test images were collected for each class, and were manually cleaned. On purpose, the training images were not cleaned, and thus still contain some amount of noise. This comes mostly in the form of intense colours and sometimes wrong labels to increase the robustness of the data. The idea is to increase the robustness of any classifier trained on the dataset.

3. Progress of Work

3.1 Workflow

The focus is on a system that can deal with non-uniformity of the background and is extendible to out of sample test images. This would ensure that the system can be used to design a ready-to-use product. A calorie estimation system can be divided roughly into two units: a recognition system and a volume estimator. As of now, the focus is on the recognition system and a workflow has been designed for it.

Images have to be preprocessed to eliminate any kind of bias due to the intensity/illumination of the image. Techniques such as Histogram Equalization can be used for such purposes.

Adversarial training is then performed on the Generator over the food images as follows:

- **Generator**: Takes some random noise as input, and transforms it, outputting a sample of data. The goal of the generator is to eventually output diverse data samples from true data distribution.
- **Discriminator**: Takes a sample of data as input and classifies it as real (from the data distribution) or fake (from the generator). The goal of the discriminator is to be able to discriminate between the real and discriminated images with high precision.

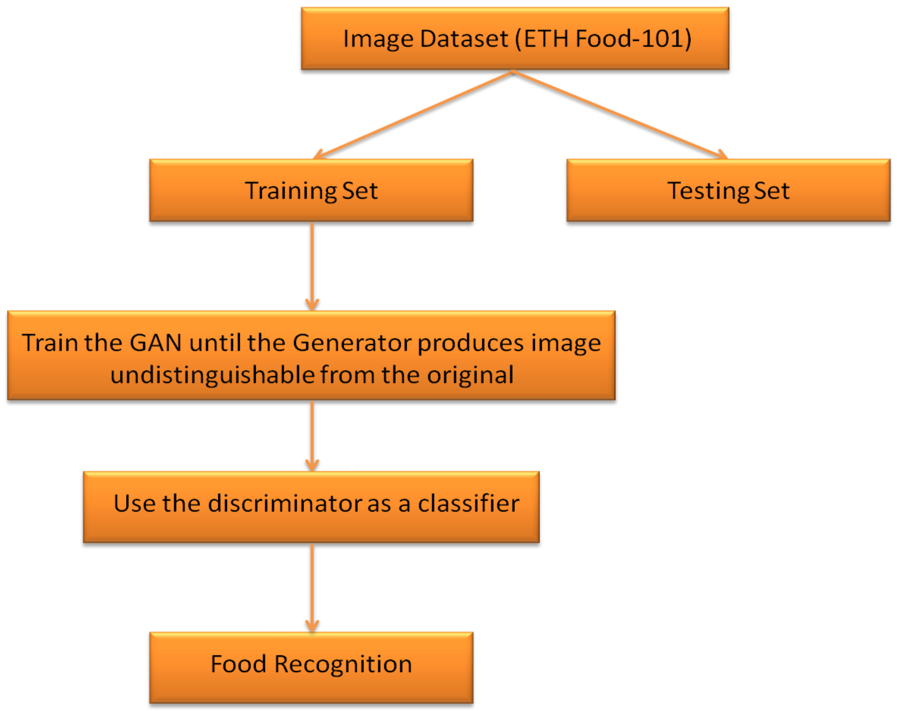


Image Dataset (ETH Food-101)
Training Set
Testing Set
Train the GAN until the Generator produces image undistinguishable from the original
Use the discriminator as a classifier
Food Recognition

A natural question that arises is, which network internalizes the understanding of the problem, the generator or the discriminator? The discriminator is the network which learns the nature of the problem better and thus can be used as a feature extractor for the images. These extracted features can then be used to train a multi-label classifier to recognize different food items.

3.2 Experimental Setup

As part of initial experiments, I have tried to perform the training of GAN's using MNIST and CIFAR-10 datasets. Both of these datasets have images with smaller size and thus are ideal for initial model testing.

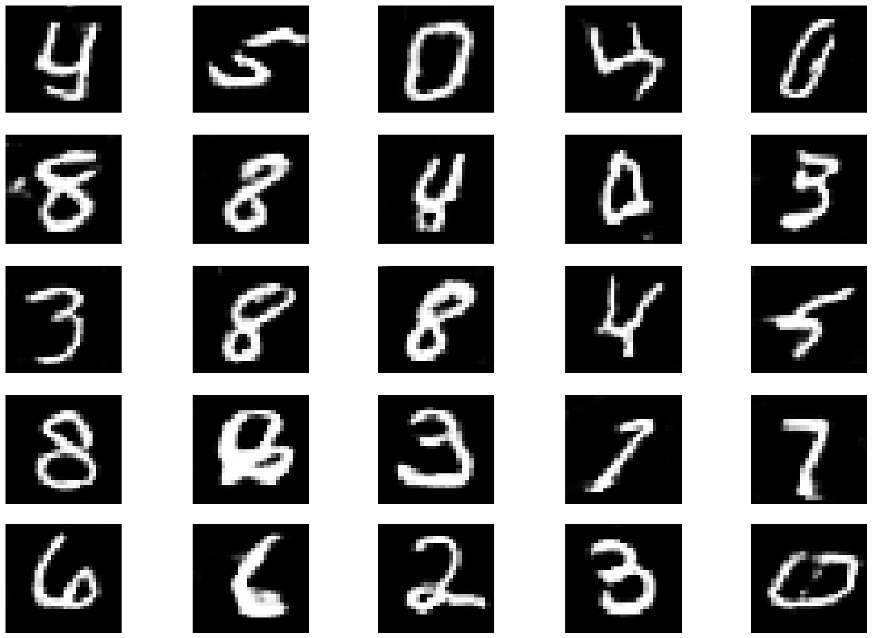
Also, the discriminator has been given (n+1) output instead of 1 output, in which each of the n outputs represent the n output classes and the last output gives the probability of the image being fake or real. This kind of training is called semi-supervised learning and has been found to give better classification results in literature.

- GAN type : DCGAN
- Optimizer : ADAM
- Activation : Leaky ReLu (in all hidden layers), Sigmoid (in discriminator's output layer)
- Batch Normalization applied to all layers except the generator's output layer and discriminator's input layer.

The generated MNIST images look like:

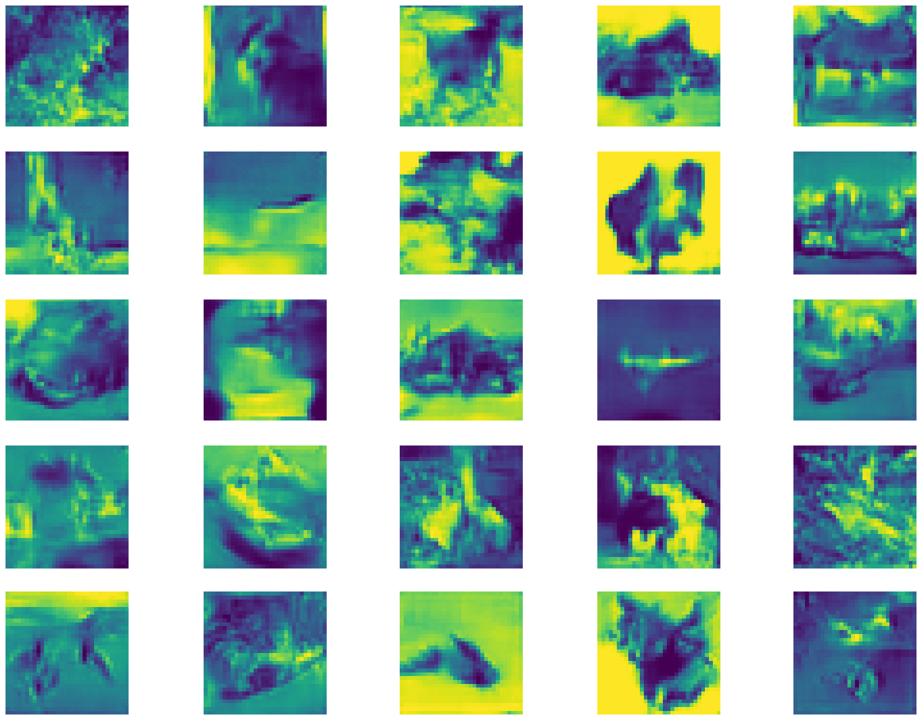
- Image Size : 28 X 28 X 1

- Epochs: 20,000
- Saved the weights and ran the dataset on just the discriminator initialized with saved weights.
- While running the data on the training and test set, preserved the output from the second-last layer of the discriminator in a numpy array and used it for training an SVM classifier with an RBF kernel from scikit-learn and testing it.
- Test accuracy = 0.77 or 77%



The generated CIFAR-10 images look like:

- Image Size : 32 X 32 X 3
- Epochs : 90,000
- Saved the weights and ran the dataset on just the discriminator initialized with saved weights.
- While running the data on the training and test set, preserved the output from the second-last layer of the discriminator in a numpy array and used it for training an SVM classifier with an RBF kernel from scikit-learn and testing it.
- Test Accuracy = 0.34 or 34%



3.3 Future Work

- Improve the performance on the CIFAR-10 dataset.
- Apply the same experimental setup to the ETH-101 food dataset but with some image preprocessing.

4. References

- Austin Myers, Nick Johnston, Vivek Rathod, Anoop Korattikara, Alex Gorban, Nathan Silberman, Sergio Guadarrama, George Papandreou, Jonathan Huang and Kevin Murphy,"Im2Calories: towards an automated mobile vision food diary," in ICCV, 2015.
- Lukas Bossard, Matthieu Guillaumin and Luc Van Gool, "Food-101: Mining Discriminative Components with Random Forests," in ECCV, September 2014.
- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, et al, "Generative adversarial nets," in NIPS, 2014.
- Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in NIPS, 2012.
- Weibo Liua, Zidong Wanga, Xiaohui Liua, Nianyin Zengb, Yurong Liuc and Fuad E. Alsaadid, "A survey of deep neural network architectures and their applications," in Neurocomputing, 2017.

- Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu and Michael S.Lew, "Deep learning for visual understanding : A review," in Neurocomputing, 2016.
- Ian Goodfellow, "NIPS 2016 Tutorial: Generative Adversarial Networks," in NIPS, 2016.
- Alec Radford,Luke Metz,and Soumith Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in arXiv preprint arXiv:1511.06434, 2015.
- Antonia Creswell , Tom White , Vincent Dumoulin , Kai Arulkumaran , Biswa Sengupta and Anil A Bharath, “Generative Adversarial Networks: An Overview” in arXiv preprint arXiv:1710.07035v1, 2017.