Food-101 Classification using Densenet161:

Problem Definition:

Recent growth in nutrition related diseases globally have increased awareness in keeping healthy nutritional habits. Healthy diets reduce the risks of reactions to food intolerance, weight problems, malnutrition and some forms of cancer. There are several applications in existence with which we can manually keep track of what we eat and identify food items before consumption. These applications however require that previous experience with the food item for easy identification. An important question however is: what happens if we see a food item for the first time and need to identify it? Automated tools of food identification will be of help in such case. The advent of Convolutional Neural Networks (CNN) and deep learning based architectures have opened opportunities for the realization of such automatic tools. Indeed, a lot of mileage has already been made in neural networks based image food classification. However, there are still gaps in accuracy of existing methods implemented,

Deep learning aims to learn multiple levels of representation and abstraction that help infer knowledge from data such as images, videos, audio, and text, is making astonishing gains in computer vision, speech recognition, multimedia Analysis. Here we will deep learning model to extract features from images of foods and automatically classify it. The first goal is to be able to automatically classify food item images previously unseen by our model. Beyond this, there are a number of possibilities for looking at what regions/image components are important for making classifications, identify new types of food as combinations of existing tags, build object detectors which can find similar objects in a full scene.Here supervised learning will be used so, the data needs to be labeled properly before using for training our deep learning model. Pytorch framework will be used for the task.

Analysis:

As a step of solving the problem we are to going to use supervised learning so we need labeled data to train our deep learning model. For this task we are going to use Food-101 dataset which contains huge 101000 images divided into 101 classes. Each class contains 1000 images each. The images of the dataset are like these:

From the images we can see that the image contains the particular food with background. So our model need to identify the food from its background at first before classification.

When it comes to image we need a model which is very sensitive to spatial information and very good in feature extraction. For this specific purpose we are going to use convolutional neural network or CNN which performs greatly when it comes to feature representation of visual imagery. It mainly contains two main part:

Convolution layer for feature extraction and fully connected layer for either classification or regression. In this scenario we need a fc layer which is suited for classification, so we will build our model according to our purpose.

After choosing the kind of network, we are going to use transfer learning and pretrained model to solve our task. Transfer learning will reduce the training time and help us to achieve better accuracy.

We have many pretrained model which were trained on the ImagNet dataset at our disposal. They contains state of the architecture for feature extraction and we can also

access the saved weights from its model. The only change we need to perform is that we need to change its fully connected layers to support our specific task.

Another thing we have to take care of is image pre-processing. These model accepts images of certain size(224*224), we have to resize the images during preprocessing and also normalize it.

Then we will split the dataset in train-valid-test set. The ratio is not fixed and subject to experimentation.

After that we will train multiple model and choose the model with highest accuracy. Then it will be tested on test set. After evaluating the performance of the model on the unseen data we will choose our best model.

After to use this model in real life , a flask based web app will be created. The web application will take image as input and tell the category of food from the image. After it we are going to host our web app in live server to be accessed in real time.

Implementation:

We are going to use pytorch as framework as it provides more flexibility and control on the whole pipeline.

We spilt the whole dataset into the following ratio:

Train:Valid:Test=85:10:10

Image Preprocessing:

Pytorch provides the API for loading and preprocessing raw images from the user. However, the dataset and the images in the raw state as obtained aren't suitable for further processing. Consequently, successive transformations are used to preprocess the training dataset. In our implementation, these transformations include: Random

rotation ,Random resized crop, Random horizontal flip, Imagenet policy and at the end Normalization.

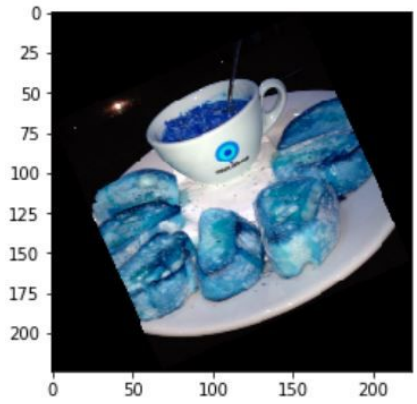
These preprocessing transforms are used to mitigate the disparities in image properties due to different image backgrounds; to help the model learn faster; and to improve the output accuracy.

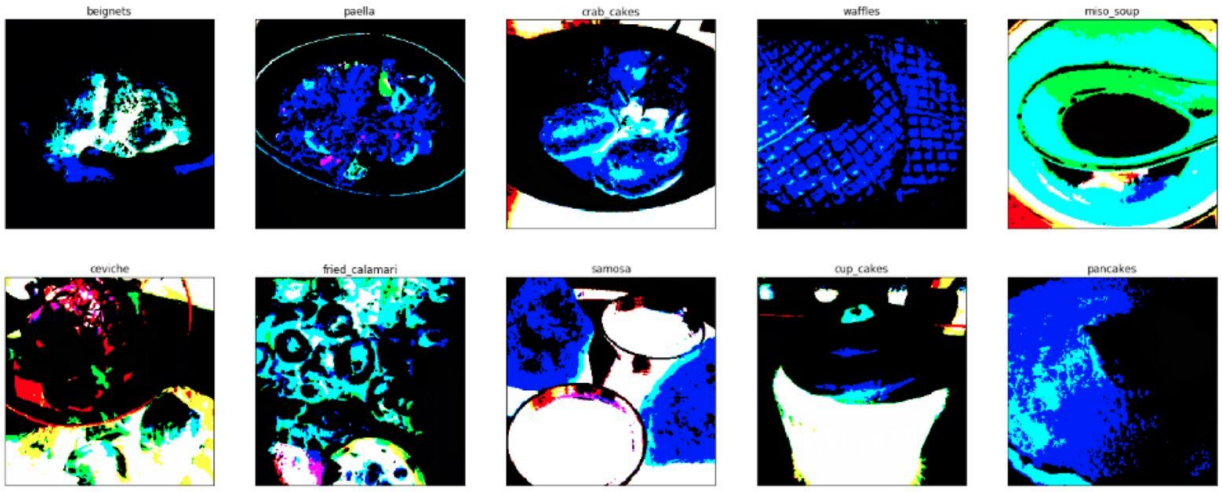
Our transforms for both the training and test data are defined as follows:

```
train_transforms = transforms.Compose([transforms.RandomRotation(30),
                                       transforms.RandomResizedCrop(224),
                                       transforms.RandomHorizontalFlip(),ImageNetPolicy(),
                                       transforms.ToTensor(),
                                       transforms.Normalize([0.485, 0.456, 0.406],
                                                            [0.229, 0.224, 0.225])])

test_transforms = transforms.Compose([transforms.Resize(255),
                                      transforms.CenterCrop(224),
                                      transforms.ToTensor(),
                                      transforms.Normalize([0.485, 0.456, 0.406],
                                                           [0.229, 0.224, 0.225])])
```

We are using `transforms.RandomResizedCrop(224)` as the pretrained model accepts images of 224*224 resolution. And we are heavy augmentation like RandomRotation, RandomHorizontalFlip,ImageNetPolicy(). Here ImageNetPolicy is custom augmentation pipeline which you can find in the underlying link before. After that we are converting the image into tensor and normalizing it. The mean and std values are chosen as the same values on which these pretrained models were originally trained. These are some examples of transformed images:

```
torch.Size([3, 224, 224])
bruschetta


<matplotlib.image.AxesImage at 0x7f36044f1940>
```





Model Architecture & Training:

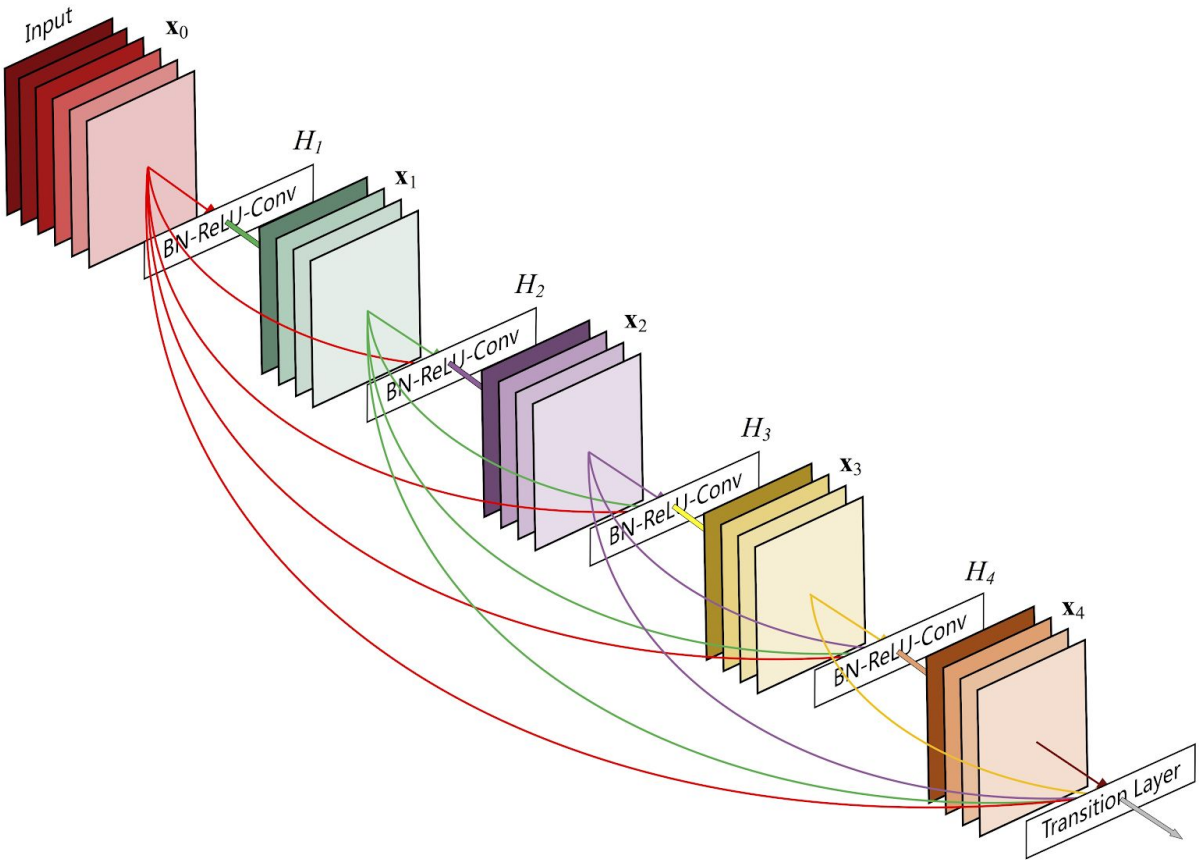
Now we are going to download a pretrained model from torchvision library:

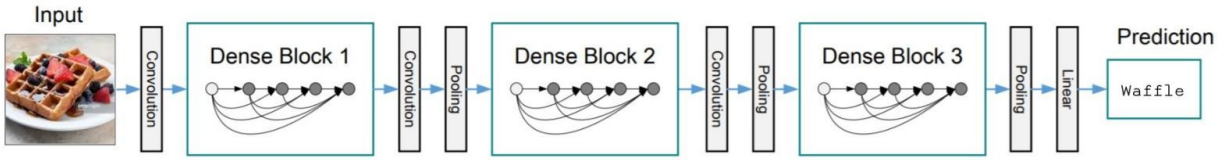
```
model =models.densenet161(pretrained=True)
```

So we downloaded densenet161 model which is based on is another state-of-the-art CNN architecture inspired by the cascade-correlation learning architecture proposed in

NIPS, 1989. The architecture connects each layer to every other layer in a feed-forward fashion. Whereas traditional convolutional networks with L layers have L connections—one between each layer and its subsequent layer—our network has L(L+1) 2 direct connections. For each layer, the feature-maps of all preceding layers are used as inputs, and its own feature-maps are used as inputs into all subsequent layers.

The advantages of Densenet include its ability to alleviate the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters.





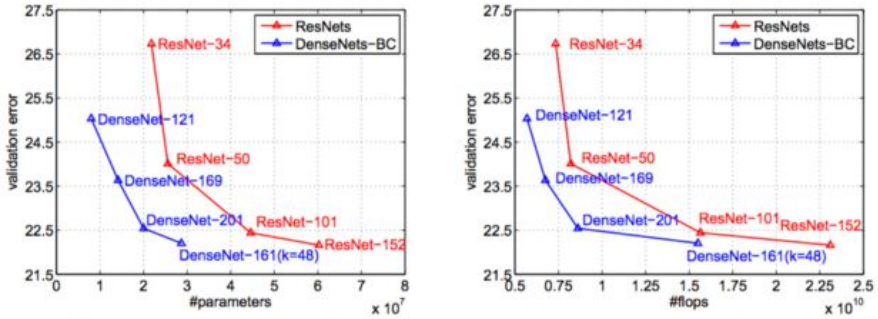


Figure 3. Comparison of the DenseNet and ResNet Top-1 (single model and single-crop) error rates on the ImageNet classification dataset as a function of learned parameters (*left*) and flops during test-time (*right*).

We set pretrained=True so that we can access the saved weights for better performance. This is another advantage of using pretrained model.

After it we are going to change architecture of last layers to suit our particular task:

```
classifier = nn.Sequential(OrderedDict([
                          ('fc1', nn.Linear(25088, 1000)),
                          ('relu', nn.ReLU()),
                          ('dropout', nn.Dropout(p=0.3)),
                          ('fc2', nn.Linear(1000, 101)),
                          ('logsoftmax', nn.LogSoftmax(dim=1))]))
model.classifier=classifier
```

The number of neurons in the last layers is 101 equal to our number of categories of food. As activation RELU is used and for fighting overfitting dropout is also used.

Next is loss function and optimizer:

```
criterion = nn.NLLLoss()
optimizer = optim.Adam(model.classifier.parameters(), lr=0.001, betas=[0.9, 0.999])
```

For loss function negative likelihood loss and for optimizer Adam is used.

As now for training the model we divided the training into two phases:

1. First we only train the classifier part so that we don't change the weights of the features part.
2. Then we train the whole model to achieve highest accuracy possible.

To ensure the model generalises well during training, model's respective performance on validation set will be tracked and we will choose the best model from that performance.

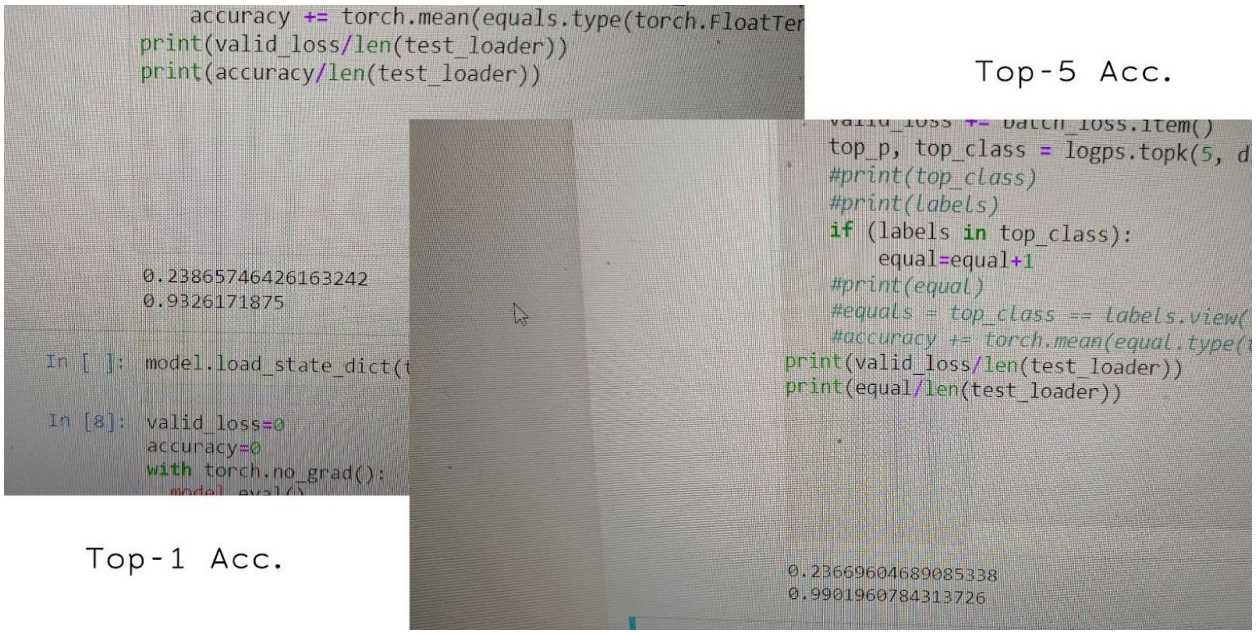
After that finally the performance on test set will be checked.

Then we will test the model with a picture of food and see the what it classifies.

I have also included the code to create a flask based web app which will accept image as input and tell us the which food it is.

Result:

After completion of training, model's performance on test set is as follows:

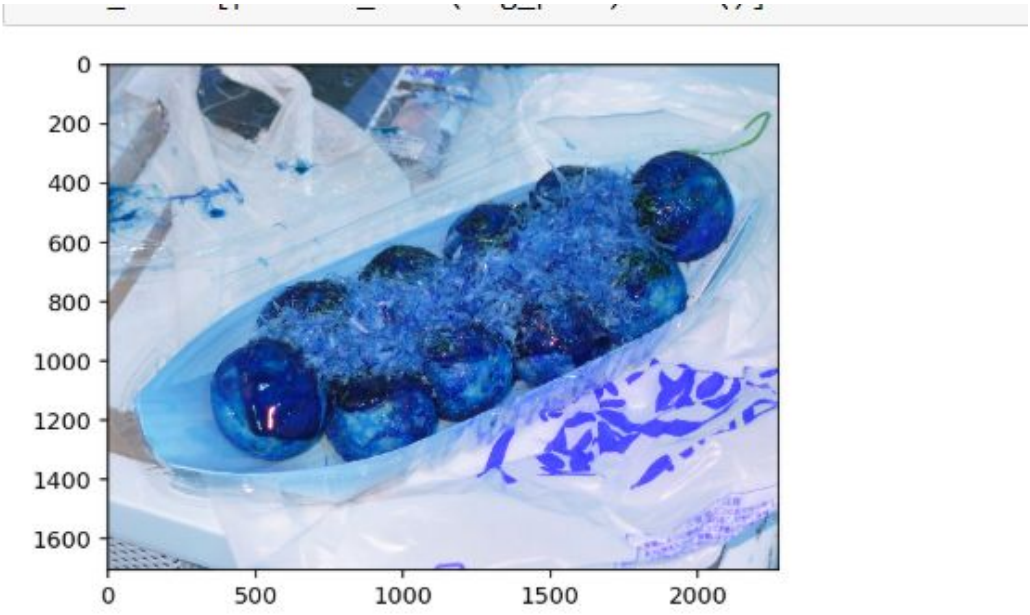


As can be seen , top-1 accuracy is 93.26% and top-5 accuracy is 99.02% which clearly beats the current best benchmark.

Method	Top - 1	Top - 5	Publication
HoG	8.85	-	ECCV2014
SURF BoW-1024	33.47	-	ECCV2014
SURF IFV-64	44.79	-	ECCV2014
SURF IFV-64 + Color Bow-64	49.40	-	ECCV2014
IFV	38.88	-	ECCV2014

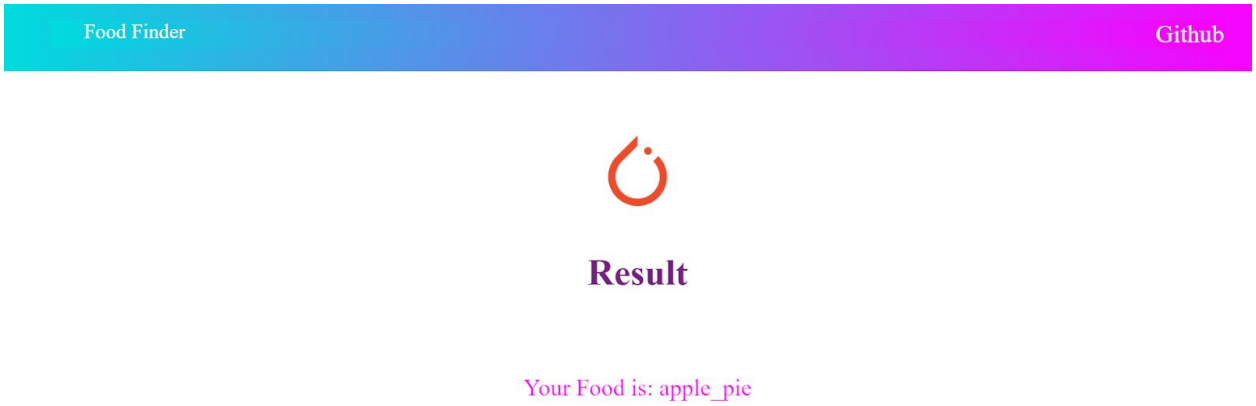
RF	37.72	-	ECCV2014
RCF	28.46	-	ECCV2014
MLDS	42.63	-	ECCV2014
RFDC	50.76	-	ECCV2014
SELC	55.89	-	CVIU2016
AlexNet-CNN	56.40	-	ECCV2014
DCNN-FOOD	70.41	-	ICME2015
DeepFood	77.4	93.7	COST2016
Inception V3	88.28	96.88	ECCVW2016
ResNet-200	88.38	97.85	CVPR2016
WRN	88.72	97.92	BMVC2016
WISeR	90.27	98.71	UNIUD2016
DenseNet - 161	93.26	99.01	Proposed

These is the result of live images given as input to the model:




```
Out[60]: 'takoyaki'
```

This is the image of the flask based web app:



The another variation of densenet model(densenet201) which was created during training with various model is hosted at :

https://food-finder-arka.herokuapp.com/

Conclusion:

There is room for further improvement:

- Mobile application for automatic food identification.
- More augmentation techniques can be used like five crops and ten crops etc.
- Inclusion of more food classes in the model classification capability. This will be done by collecting and annotating more data in varying food categories across different parts of the world.