Feature Extraction Through Transfer Learning Algorithms

Atul Tiwari 11800176

Atult2208@gmail.com , +91 9915065571

Abstract :- In this project we have used transfer learning models in Tensorflow to extract features of an image to classify them with the help of traditional machine learning algorithms like Svm, KNN etc. This will help understand the use of transfer learning for different kind of purposes. Then we perform multi-class classification on fruit 360 data set to evaluate our results and success rate of our models based upon different parameters we are also going to compare the time to train the algorithms.

1. Introduction

Deep convolutional neural network models may take days or even weeks to train on very large datasets.

In this experiment we are going to use various pre-trained models and compare them on different parameters because In practice, very few people train an entire Convolutional Network from scratch (with random initialization), because it is relatively rare to have a data-set of sufficient size. Instead, it is common to pretrain a ConvNet on a very large data-set (e.g. ImageNet, which contains 1.2 million images with 1000 categories), and then use the ConvNet either as an initialization or a fixed feature extractor for the task of interest.

1.1 Transfer learning

- Transfer learning involves using models trained on one problem as a starting point on a related problem.
- Transfer learning is flexible, allowing the use of pre-trained models directly, as feature extraction preprocessing, and integrated into entirely new models.
- Keras provides convenient access to many top performing models on the ImageNet image recognition tasks such as VGG, Inception, and ResNet.

1.2 Model's

In this experiment we are going to use 5 feature extraction models and 5 traditional machine learning models to evaluate our results

Feature Extraction models (Keras)	ML Algorithms (SK-learn)
VGG16	SVM
ResNet50V2	KNN
MobileNetV2	Random Forest
InceptionV3	Decision Tree
DenseNet121	Bagging

All features extraction models are using their default parameters and the only one specified are

- Input_shape = (100,100,3)
- Include_top = False
- weights = 'imagenet'
- Training = False

The Traditional models are using following parameters which are not mention are all default parameters provided by sklearn

- SVM
  - kernel ='rbf'
  - random_state =0
- KNN
  - n_neighbors=3
- Random Forest
  - max_depth=25
  - random_state=0
- Decision Tree
  - random_state=0
- Bagging
  - base_estimator=DecisionTreeClassifier()
  - n_estimators=5
  - random_state=0

2. Data Set

We are using a Fruits 360 dataset: A dataset of images containing images of various fruits the properties of dataset as follows

- Total number of images: 90483.
- Training set size: 67692 images (one fruit or vegetable per image).
- Test set size: 22688 images (one fruit or vegetable per image).
- Number of classes: 131 (fruits and vegetables).
- Image size: 100x100 pixels.

2.1 Modifications

The original data set contains two folders train and test I have combine the two and place all the data in a single folder name 'Dataset' for ease of use in my project and I am using all the 131 classes for this feature extraction.

2.2 Batch

Now after reading all the 90483 images then we have converted them into 19 batch of 5000 each for ease of computation after each batch is complete we store it in a csv file for respective model folder

3. Testing Methodology

For each traditional ML model we are storing following in a python object file which can be read through printresults.py file

- Accuracy
- recall
- precision
- f1 score
- Confusion matrix
- Classification report
- Training Time (70% of data)
- Testing Time (30% of data)

4 . Feature Extraction

4.1 VGG16

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”. The model achieves 92.7% top-5 test accuracy in ImageNet, which is a data-set of over 14 million images belonging to 1000 classes. It was one of the famous model submitted to ILSVRC-2014. It makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. VGG16 was trained for weeks and was using NVIDIA Titan Black GPU’s.

VGG16 Function Parameters

- input _Shape = (100,100,3)
- Include_top = False
- weights = 'imagenet'

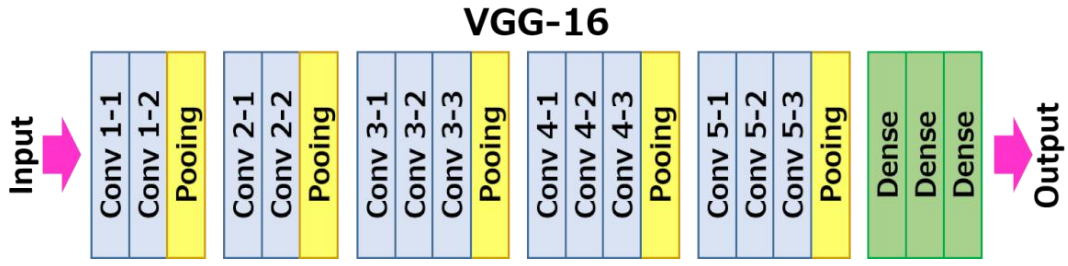
Input layer = 3,00,00
Model Parameters = 14714688
Output = 25

```
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 100, 100, 3)]     0
_________________________________________________________________
vgg16 (Functional)           (None, 3, 3, 512)         14714688
_________________________________________________________________
global_average_pooling2d (Gl (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 25)                12825
=================================================================
Total params: 14,727,513
Trainable params: 14,727,513
Non-trainable params: 0
_________________________________________________________________
```

VGG16 Results

Algorithms	Accuracy	Recall	Precision	F1 score	Train time (in Sec)	Pred time (in Sec)
SVM	99.78	99.78	99.78	99.78	13	114
KNN	99.98	99.99	99.98	99.99	1	5
Random Forest	99.94	99.93	99.94	99.94	81	2
Decision Tree	90.46	90.46	90.54	90.47	7	1
Bagging	96.34	96.34	96.42	96.35	21	1

Original VGG16



4.2 RESNET50

ResNet, short for Residual Networks is a classic neural network used as a backbone for many computer vision tasks. This model was the winner of ImageNet challenge in 2015. The fundamental breakthrough with ResNet was it allowed us to train extremely deep neural networks with 150+layers successfully. Prior to ResNet training very deep neural networks was difficult due to the problem of vanishing gradients.

ResNet-50 that is a smaller version of ResNet 152 and frequently used as a starting point for transfer learning.

ResNet 50 Function Parameters

- input _Shape = (100,100,3)
- Include_top = False
- weights = 'imagenet'

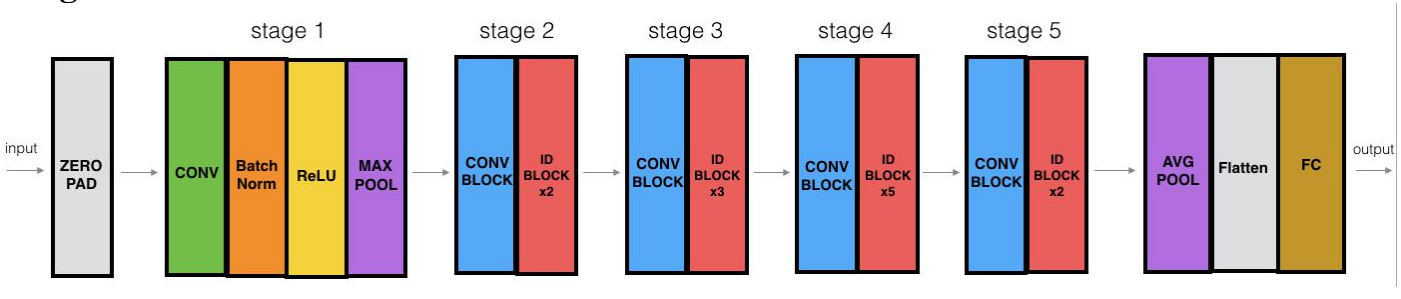
Input layer = 3,00,00
Model Parameters = 23564800
Output = 25

```
None
Model: "functional_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         [(None, 100, 100, 3)]     0
_________________________________________________________________
resnet50v2 (Functional)      (None, 4, 4, 2048)        23564800
_________________________________________________________________
global_average_pooling2d_1 ( (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 25)                51225
=================================================================
Total params: 23,616,025
Trainable params: 23,570,585
Non-trainable params: 45,440
_________________________________________________________________
```

ResNet 50 Results

Algorithms	Accuracy	Recall	Precision	F1 score	Train time (in Sec)	Pred time (in Sec)
SVM	73.40	73.40	74.19	72.51	50	220
KNN	99.52	99.52	99.53	99.52	1	4
Random Forest	99.35	99.35	99.36	99.35	83	3
Decision Tree	88.46	88.46	88.49	88.44	7	1
Bagging	94.90	94.90	95.07	94.90	21	1

Original ResNet 50



4.3 MoblieNetV2

Mobile-net is an family of mobile-first computer vision models for TensorFlow, designed to effectively maximize accuracy while being mindful of the restricted resources for an on-device or embedded application. MobileNets are small, low-latency, low-power models parameterized to meet the resource constraints of a variety of use cases. They can be built upon for classification, detection, embeddings and segmentation similar to how other popular large scale models, such as Inception, are used.

VGG16 Function Parameters

- input _Shape = (100,100,3)
- Include_top = False
- weights = 'imagenet'

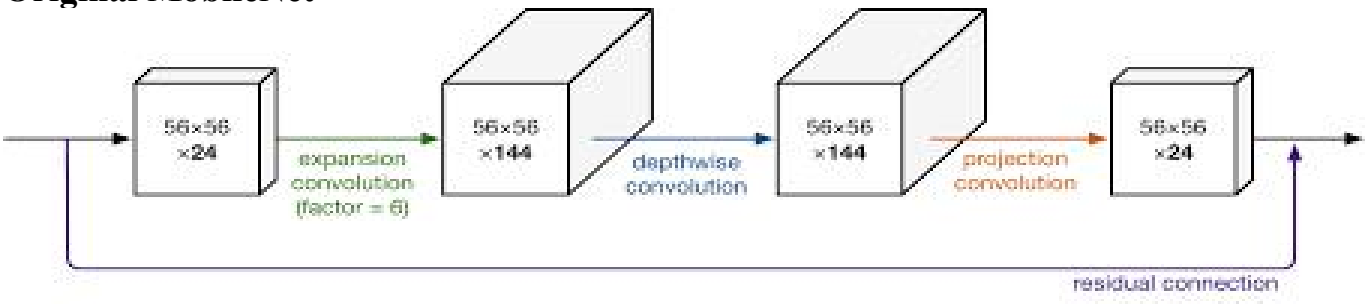
Input layer = 3,00,00
Model Parameters = 2257984
Output = 25

```
Layer (type)                 Output Shape              Param #
=================================================================
input_6 (InputLayer)         [(None, 100, 100, 3)]     0
_________________________________________________________________
mobilenetv2_1.00_224 (Functi (None, 4, 4, 1280)        2257984
_________________________________________________________________
global_average_pooling2d_2 ( (None, 1280)              0
_________________________________________________________________
dense_2 (Dense)              (None, 25)                32025
=================================================================
Total params: 2,290,009
Trainable params: 2,255,897
Non-trainable params: 34,112
_________________________________________________________________
```

MobileNetV2 Results

Algorithms	Accuracy	Recall	Precision	F1 score	Train time (in Sec)	Pred time (in Sec)
SVM	91.04	91.04	91.11	91.07	50	176
KNN	97.45	97.45	97.52	97.46	1	61
Random Forest	90.98	90.98	91.17	90.94	90	4
Decision Tree	52.18	52.18	52.35	52.17	7	1
Bagging	63.72	63.72	67.68	64.52	22	1

Original MobileNet



4.4 InceptionV3

Inception net achieved a milestone in CNN classifiers when previous models were just going deeper to improve the performance and accuracy but compromising the computational cost. The Inception network, on the other hand, is heavily engineered. It uses a lot of tricks to push performance, both in terms of speed and accuracy. It is the winner of the ImageNet Large Scale Visual Recognition Competition in 2014, an image classification competition, which has a significant improvement over ZFNet (The winner in 2013), AlexNet (The winner in 2012) and has relatively lower error rate compared with the VGGNet.

ResNet 50 Function Parameters

- input _Shape = (100,100,3)
- Include_top = False
- weights = 'imagenet'

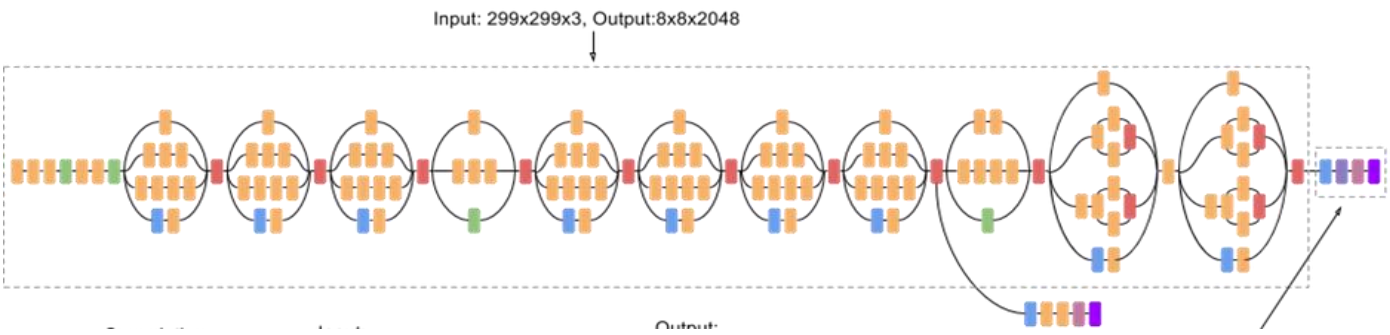
Input layer = 3,00,00
Model Parameters = 21802784
Output = 25

```
None
Model: "functional_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_8 (InputLayer)         [(None, 100, 100, 3)]     0
_________________________________________________________________
inception_v3 (Functional)    (None, 1, 1, 2048)        21802784
_________________________________________________________________
global_average_pooling2d_3 ( (None, 2048)              0
_________________________________________________________________
dense_3 (Dense)              (None, 25)                51225
=================================================================
Total params: 21,854,009
Trainable params: 21,819,577
Non-trainable params: 34,432
_________________________________________________________________
```

InceptioV3 Results

Algorithms	Accuracy	Recall	Precision	F1 score	Train time (in Sec)	Pred time (in Sec)
SVM	87.56	87.56	87.99	87.50	33	190
KNN	99.50	99.90	99.50	99.50	1	14
Random Forest	98.22	98.22	98.26	98.21	90	4
Decision Tree	74.44	74.44	74.51	74.45	8	1
Bagging	85.01	85.01	86.13	85.17	22	1

Original InceptioV3



4.5 DenseNet121

In DenseNet, each layer obtains additional inputs from all preceding layers and passes on its own feature-maps to all subsequent layers. Concatenation is used. Each layer is receiving a “collective knowledge” from all preceding layers.
Image for post
Since each layer receives feature maps from all preceding layers, network can be thinner and compact, i.e. number of channels can be fewer. The growth rate k is the additional number of channels for each layer.
So, it have higher computational efficiency and memory efficiency..

ResNet 50 Function Parameters

- input _Shape = (100,100,3)
- Include_top = False
- weights = 'imagenet'

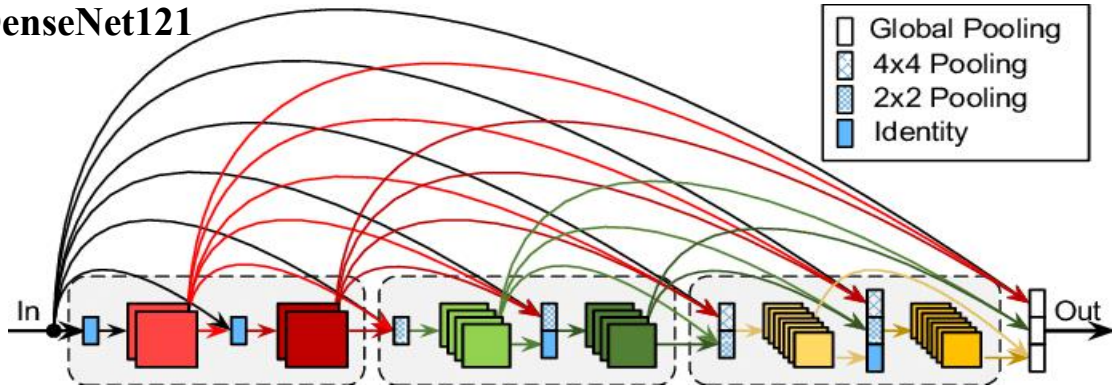
Input layer = 3,00,00
Model Parameters = 7037504
Output = 25

```
Layer (type)                 Output Shape              Param #
=================================================================
input_10 (InputLayer)        [(None, 100, 100, 3)]     0
_________________________________________________________________
densenet121 (Functional)     (None, 3, 3, 1024)        7037504
_________________________________________________________________
global_average_pooling2d_4 ( (None, 1024)              0
_________________________________________________________________
dense_4 (Dense)              (None, 25)                25625
=================================================================
Total params: 7,063,129
Trainable params: 6,979,481
Non-trainable params: 83,648
_________________________________________________________________
```

DenseNet121 Results

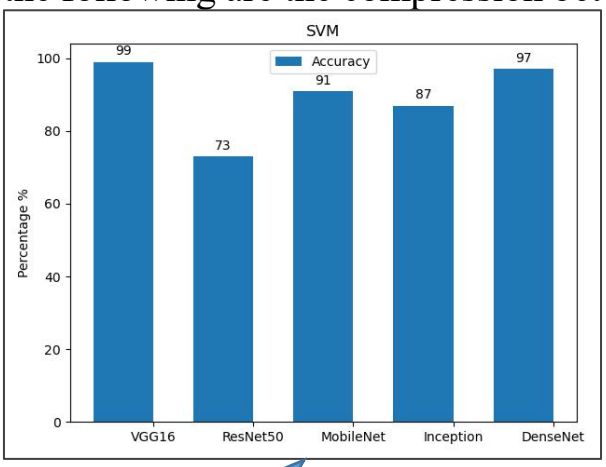
Algorithms	Accuracy	Recall	Precision	F1 score	Train time (in Sec)	Pred time (in Sec)
SVM	97.85	97.85	97.90	97.84	15	146
KNN	99.92	99.92	99.92	99.92	1	6
Random Forest	99.72	99.72	99.72	99.72	83	3
Decision Tree	89.47	89.47	89.54	89.48	7	1
Bagging	95.55	95.55	95.61	95.54	21	1

Original DenseNet121



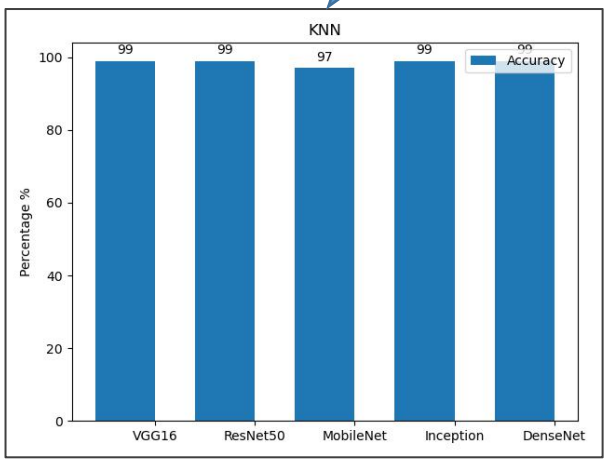
5. Result

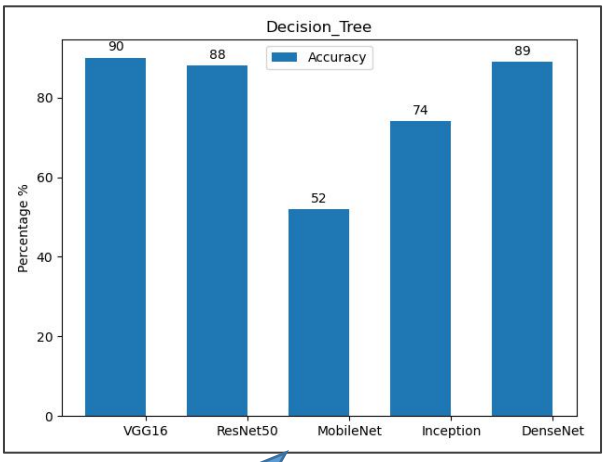
the following are the compression between the different models for different algorithms



1. **SVM** the highest is the VGG16

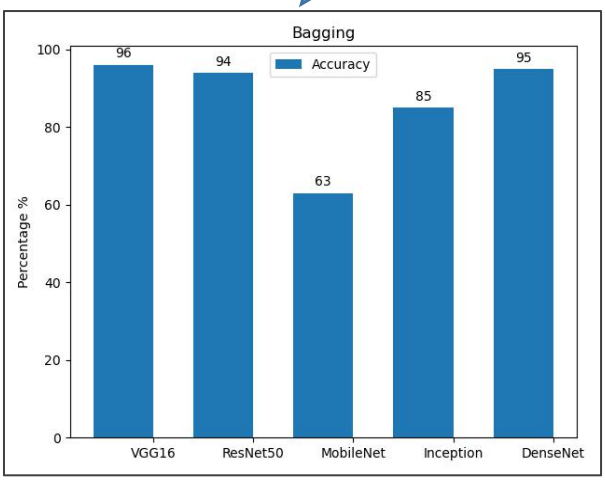
2. **KNN** the overall performance of this is the best among all algorithms

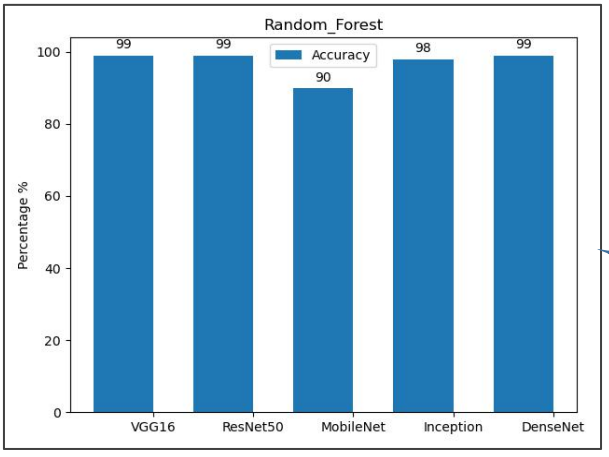




3. **Decision Tree** this is the worst performing algorithm overall

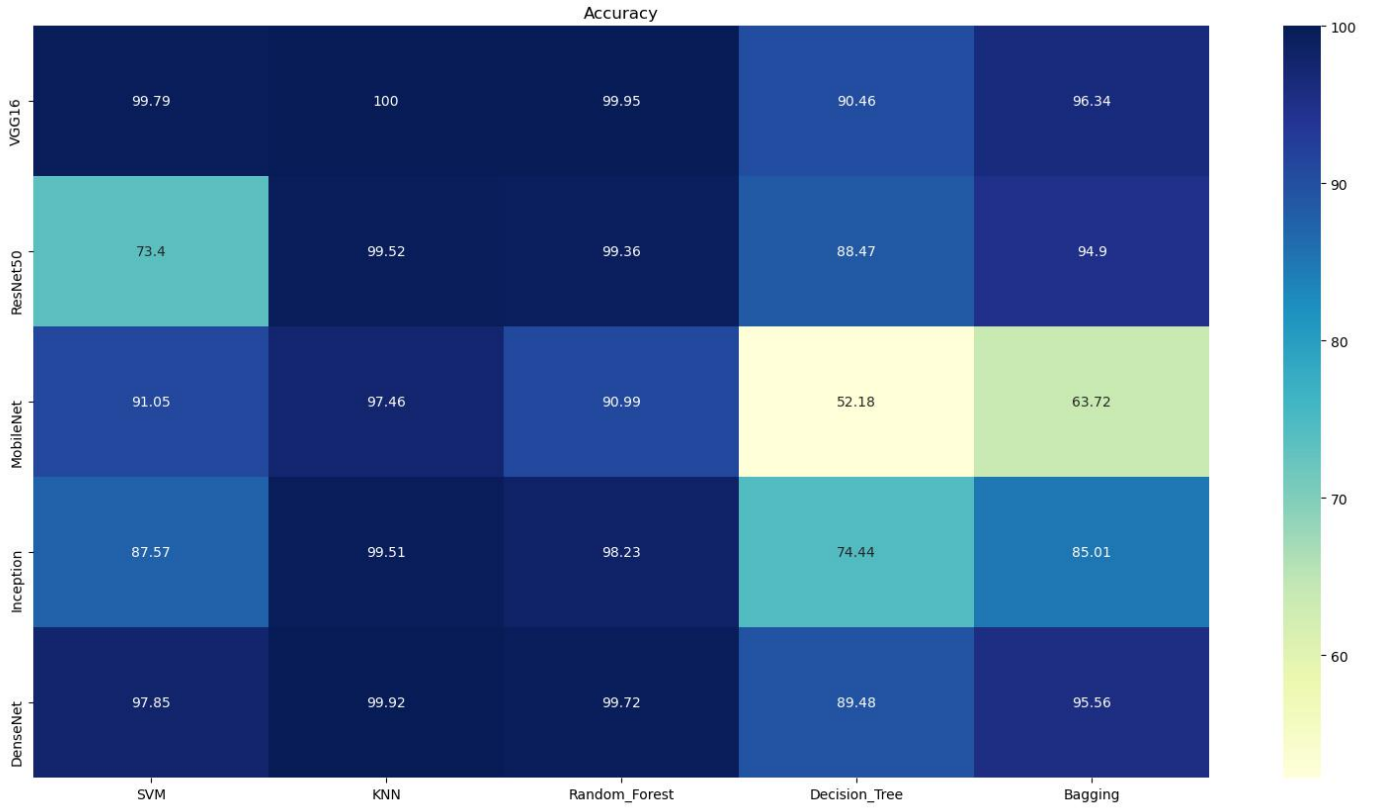
4. **Bagging** this is also not most reliable





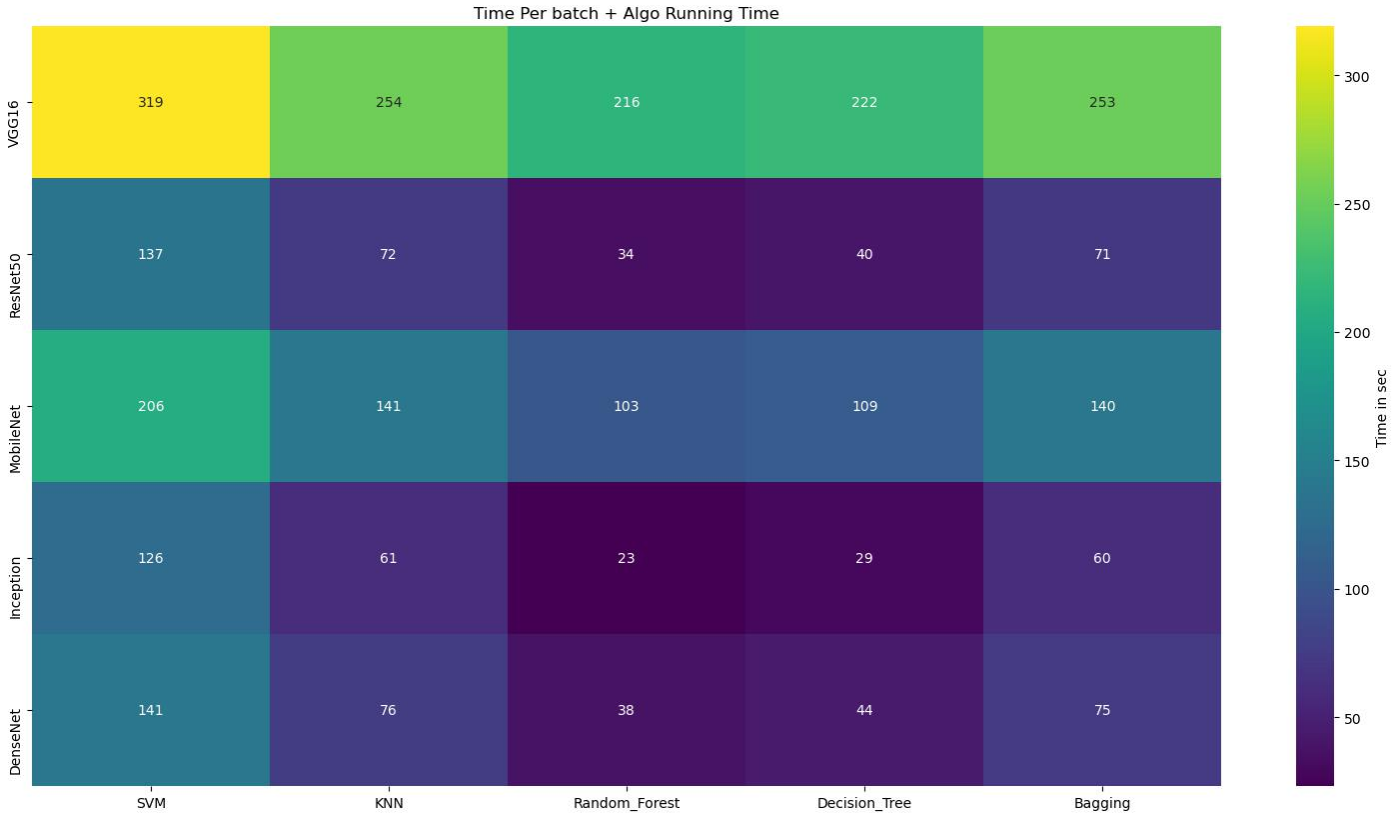
5. Random Forest is the second best and fastest algorithm among these

5.1 Compression between the Accuracy



- the above graph shows the a heat map between the algorithms and Models
- Best model is VGG16 followed by Densenet
- Best Algorithm is KNN followed by Random forest

5.2 Compression between the Running Time



- The above table shows the time taken by the model in one batch + algorithm training time + Prediction Time
- Random forest performs best in Algorithms
- VGG16 is most time consuming process to Extract features from images

5.3 Time Taken by Feature extraction Models

Data	VGG16	ResNet50	MobileNet	Inception	DenseNet121
Single Batch (5000)	118	53	15	21	52
Whole Dataset (90483)	2142	959	272	381	942

Note : -Time in second

6. Conclusion-

- If i want to go for the most accurate model the combination of VGG16 and KNN works best.
- If you want to go for fastest model the combination of Inceptionv3 and random Forest Work's best.
- If You want to go for best overall model I would suggest combination of KNN and InceptionV3. as it is more accurate then the fastest combination and faster then most accurate one.
- Through algorithms and all model they all are have more precision then recall.

Data Set Link :- https://www.kaggle.com/moltean/fruits
Git Repo Link : - https://github.com/atul-tiwari/ML-Image-Feature-Extraction