FLORIDA ATLANTIC UNIVERSITY

Final Course Project
Foundation of Vision
Fall 2017

IS IT A CAT OR A DOG ?

Submitted to,
Oge Marques, Ph.D.

Submitted by,
Debarshi Dutta
Satarupa Khamaru

Date
12/12/2017

Table of Contents

1. Introduction

The final course project was to design and build classifiers that would effectively classify images of cats and dogs using MATLAB. We took twenty-five thousand of images from the Kaggle dataset. Among them, 12,500 were cats, and rest 12,500 were dogs. The motivation behind this project came from Dogs vs. Cats competition from Kaggle which is trying to solve the CAPTCHA (Elson et al., 2007) challenge, which relies on the problem of distinguishing images of dogs and cats. It is easy for humans, but the evidence (Elson et al., 2007) suggests that cats and dogs are particularly difficult to tell apart automatically. A human can accomplish it quickly and easily, but to train an artificial intelligence network is not so easy.

The aim of this project is to effectively use the machine learning and deep learning algorithms in the field of image classification problem by using MATLAB. We choose a set of images around 60% of the total dataset as a training set (20,000 images) to train the classifier and used 20% of the total dataset as a validation set (5000 images) to validate the accuracy of the classifier. Finally, rest 20% of the total data used as a test-set (5000 images) to validate the testing accuracy of the classifier using a "confusion matrix."

2. Background

2.1 Visual categorization: From the Human Vision aspects

Previous studies showed that both prefrontal cortex (PFC) and inferior temporal cortex (ITC) are responsible for the higher level of visual processing (Freedman et al., 2003, Freedman et al., 2006, Minamimoto et al., 2010, Cromer et al., 2010, Roy et al., 2010, Segar et al., 2010). However, the respective role of these brain areas is barely known. ITC neuron seems to have properties of visual categorization (Freedman et al., 2003). Comparing the two areas: ITC and PFC, PFC seems to have the stronger categorization ability than ITC but, PFC might play a more active role in categorization (Freedman et al., 2003).

Freedman et al., (2003) trained monkeys and record neural activity from the PFC and the ITC areas. They found that after the training, ITC neurons showed enhanced activity during the feature categorization comparable to extracting the unrelated features (Freedman et al., 2003).

Previous studies also had shown that damage ITC in monkeys has deficits of categorization ability of objects (Minamimoto et al., 2010). Monkeys experienced visual agnosia for ITC neuron damage, but not damage in the PFC (dorsal lateral). (Minamimoto et al., 2010).

In 2006, Freedman did the similar kind of experiment, where the stimuli during the training and the test were the same, but orientation is different. They noticed that neuronal selectivity was stronger for stimuli presented in the same orientation as the training and test period than the different orientations.

The prefrontal cortex (PFC) is the highest order cognitive area. Cromer et al., (2010) tested whether PFC neurons could encode more than one category scheme since other models showed that specialist neurons could encode only a single category. They recorded PFC neural activity while monkeys switched task between two different individual stimuli: cats vs. dogs. The results showed that PFC neurons are "multitaskers," they showed more categorical distinctions. PFC function more independently on top-down behavioral demands than bottom-up since these neurons have abilities in distinct two categories (Roy et al., 2010).

2.2 Advances in Machine Learning/Deep Learning

2.2.1 Machine Learning: Machine learning is the process where a machine is "trained" by using large amounts of data and algorithms, that could give the ability to the machine to interact with the outside world by performing tasks. This learning came from the idea how human minds work; the artificial intelligence is a result of decision tree learning, inductive logic programming, clustering, reinforcement learning, and Bayesian networks among others. One of the best application of machine learning is computer vision still it needs a great deal of handheld coding to run the machine its way. The examples of machine learning in computer vision are edge detection by using classifier for filtering the image, so the program could classify where the object started and stopped. Another example of machine learning in computer vision is shaped detection, which could determine if the objects have many sides (https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/).

2.2.2 Deep Learning: Deep learning is an application of machine learning; it breaks down into the tasks the way where machines can be assisted. Deep learning can be supervised, semi-supervised or unsupervised. Deep learning architectures such as deep neural networks, deep belief networks and recurrent neural networks, which could be applied to the areas of computer vision, speech recognition, natural language processing, Bioinformatics, and drug design. For example, driverless cars are the results of deep learning. Figure 1 shows the traditional Machine Learning flow and Deep Learning flow in computer vision.

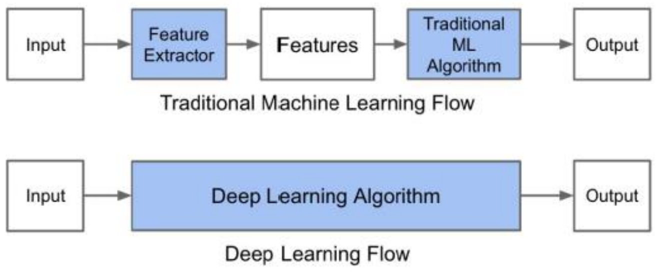


Figure 1: Traditional Machine Learning flow and Deep Learning flow in computer vision

2.3 Dog vs. Cat: An overview of CNN, Image Net, and Alexnet

2.3.1 Convolutional Neural Network (CNN)

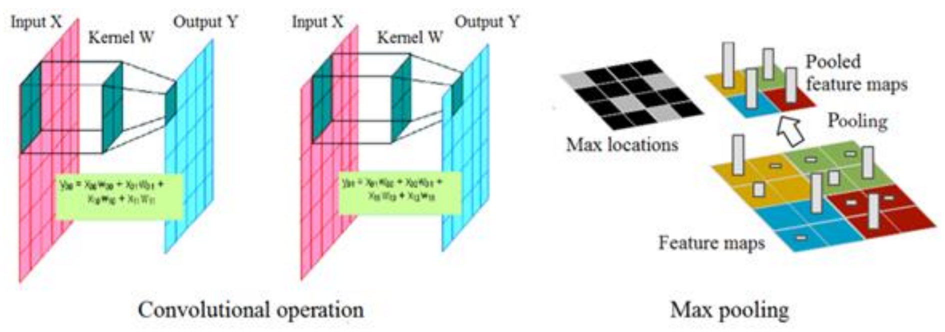


Figure 2: Convolutional Operation and Max Pooling in CNN

(Figure Source**:** https://sites.ualberta.ca/~bang3/files/DogCat_report.pdf)

The CNN is a kind of deep architecture which has achieved great performance in tasks like document recognition (LeCun et al., 1998) and image recognition (Krizhevsky et al., 2012). Different from traditional BP Neural Networks, which contains an input layer, hidden layers, and output layer, the CNN also contains Convolutional layers and Max Pooling layers (Figure 2). Convolutional Neural Networks are built up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores on the other. And they still have a loss function (e.g., SVM/Softmax) on the last (fully-connected) layer.

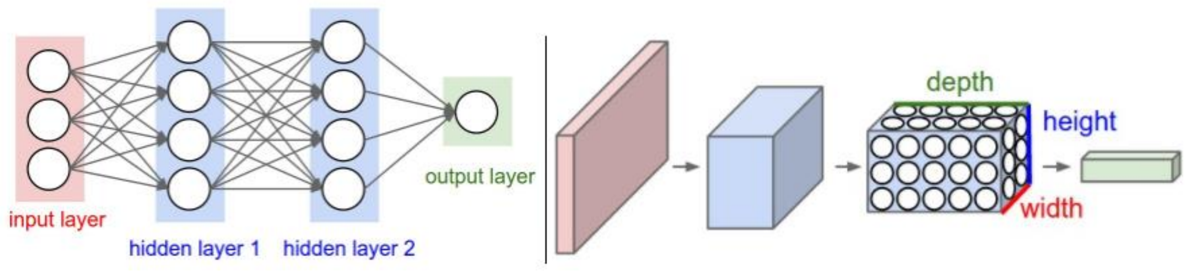


Figure 3: The left image is: a regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels). (Image Source: http://cs231n.github.io/convolutional-networks/)

2.3.2 ImageNet

ImageNet is a hierarchical model which is developed by the influence of WordNet (hierarchical model of only nouns). Each node of the ImageNet is portrayed by millions of images. Each node has an average of five hundred images. ImageNet has gained popularity as a useful resource for researchers, educators, and student all over the world (http://image-net.org/index). We took our training and test data from the Kaggle database (https://www.kaggle.com/datasets).

2.3.3 AlexNet

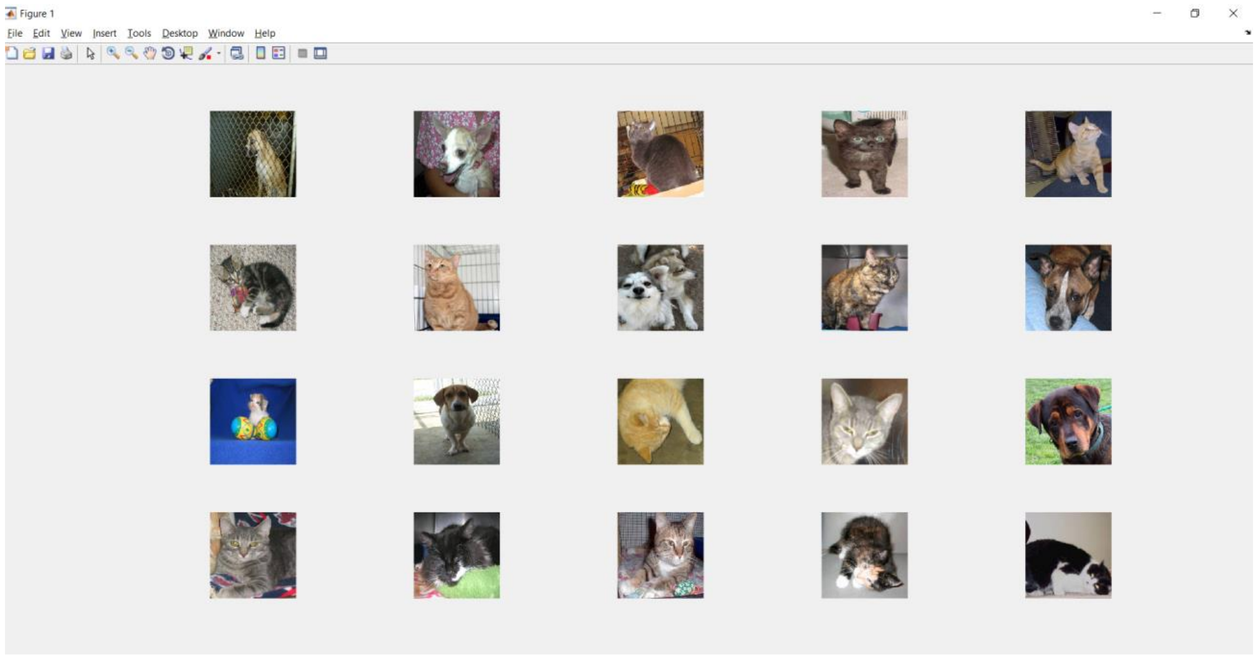
AlexNet is a pretend Convolutional Neural Network (CNN). It is trained on approximately 1.2 million images from the ImageNet Dataset. The model contains 23 layers which can classify images into 1000 object categories (e.g., Keyboard, mouse, coffee mug, pencil, cats, dogs, etc.) (https://www.mathworks.com/matlabcentral/fileexchange/59133-neural-network-toolbox-tm--model-for-alexnet-network).

2.3.4 Cats Vs. Dogs Classification Task with MATLAB

The goal of this project is to learn how to implement a complete, fully functional, machine learning solution for a contemporary computer vision challenge (Cats Vs. Dogs Classification) using MATLAB and understanding how to optimize neural network in either accuracy or efficiency side.

Our basic task was to create an algorithm to classify whether an image contains a dog or a cat. The input for this task is images of dogs or cats from the training dataset, while the output is the classification accuracy of test dataset. The given dataset for this competition is the Kaggle dataset.

The below image shows a small part of the whole dataset.



Our training set contains 20,000 images (out of which 15,000 are training set, and 5,000 are validation set partitioned by the Classification Learner App in MATLAB), while the test dataset contains 5,000 images. Our learning task is to learn a classification model to determine the decision boundary for the training data set. The whole process is illustrated in Figure 4, from which we can see the input for the learning task is images from the training dataset, while the output is the learned classification model.

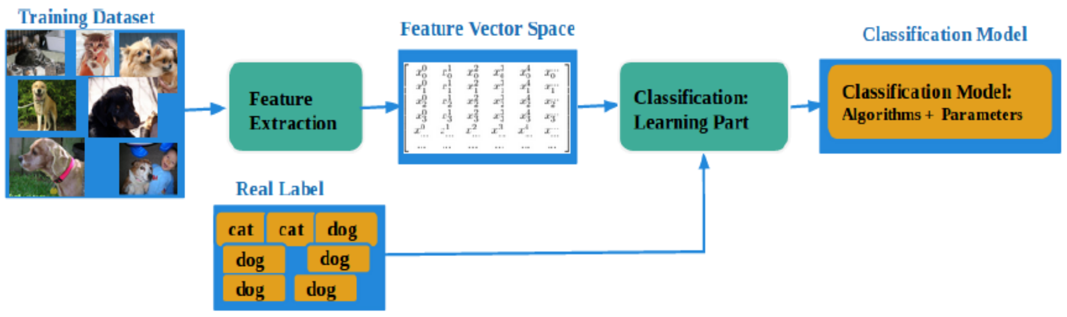


Figure 4: Architecture for learning task

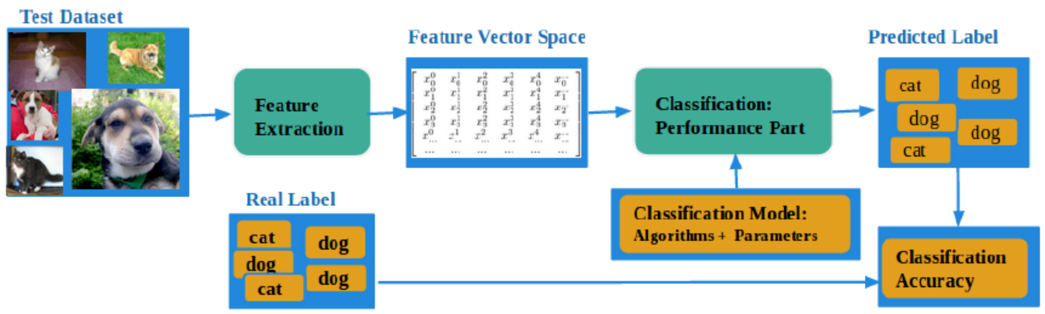


Figure 5: Architecture for Performance task

(Figure Source: https://sites.ualberta.ca/~bang3/files/DogCat_report.pdf)

Our performance task is to apply the learned classification model to classify images from the test dataset and then evaluate the classification accuracy. As seen from Figure 5, the input is images from the test dataset, and the output is the classification accuracy.

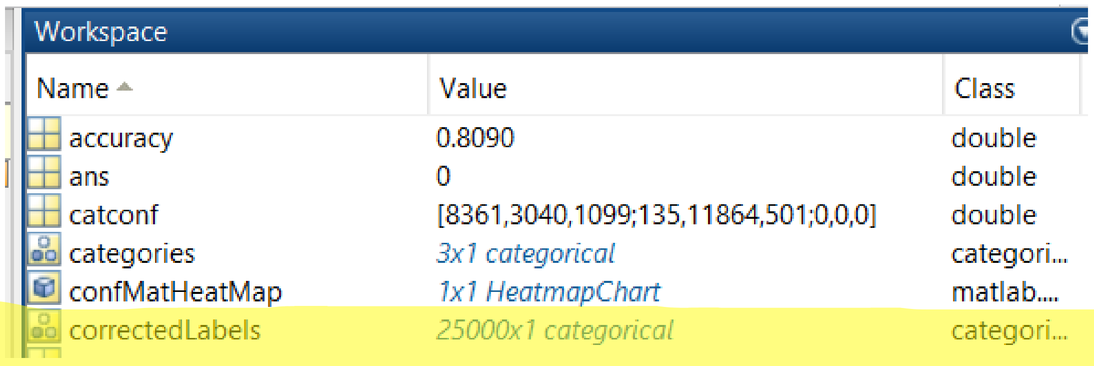
3. Solutions, Experiments, and Results

In our solution, we mainly tried three different approaches:

- A pre-trained CNN model used "as is."
- Transfer Learning using a pre-trained CNN
- A pre-trained CNN as a feature extraction block and then Support Vector Machine as a classifier trained by the Classification Learner App

3.1 Solution 1: A pre-trained CNN model, e.g., AlexNet used "as is."

As a first way of solutions, we used pre-trained CNN model AlexNet "as is" i.e. just by loading the AlexNet in MATLAB and providing 25,000 images to the ALexNet to classify cats and dogs. But, the ALexNet is not only classifying the images as a cat or dog but also it is even more specific to find the subcategories of cats and dogs, for example it is classified as tabby cat or wild dog. When the classification is done by Alexnet, we then converted specific kinds of subcategorization to the generalized cats or dogs. The below image shows the 25,000 corrected labels (either 'cat' or 'dog'). As a result, we have some images which are neither classified as 'cats' nor as 'dogs'. If we analyze the confusion matrix, we could see that out of total 25,000 images, 20225 (11864 + 8361) images are classified correctly by ALexNet. So, the accuracy is approximately 80%. Total, 3175 (3040 + 135) images are misclassified by the network that means cats are misclassified as dogs, and dogs are misclassified as cats. The remaining 1600 (1099 + 501) images are not falling to any of the category i.e. they are classified neither as a 'cat' nor as a 'dog'. Figure 6 is the graphical representation of the confusion matrix and it shows one example of misclassification by the AlexNet where a cat is classified as a dog.



Workspace

Name	Value	Class
accuracy	0.8090	double
ans	0	double
catconf	[8361,3040,1099;135,11864,501;0,0,0]	double
categories	*3x1 categorical*	categori...
confMatHeatMap	*1x1 HeatmapChart*	matlab....
correctedLabels	*25000x1 categorical*	categori...

Accuracy = 0.8090

catconf =

```
    8361      3040      1099

     135     11864       501

       0         0         0
```

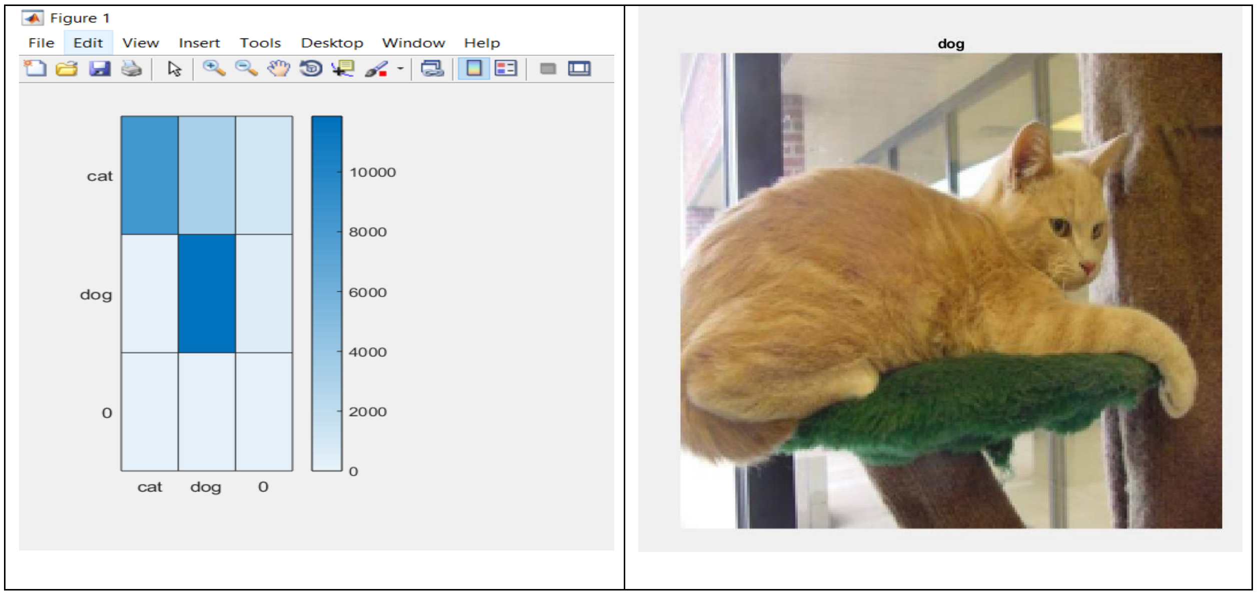


Figure 6: (left) confusion matrix; (right) incorrect classification by AlexNet

3.2 Solution 2: Transfer learning using a pre-trained CNN, e.g., Alexnet

A pre-trained CNN model (e.g., AlexNet) used under the transfer learning paradigm, i.e., with the proper modifications to one or more layers. Transfer learning is commonly used in deep learning applications. We can take a pre-trained network and use it as a starting point to learn a new task. Fine-tuning a network with transfer learning is usually much faster and easier than training a network with randomly initialized weights from scratch. We can quickly transfer learned features to a new task using a smaller number of training images.

3.2.1 Transfer Layers to New Network

The last three layers of the pre-trained network net are configured for 1000 classes. These three layers must be fine-tuned for the new classification problem. Extract all layers, except the last three, from the pre-trained network.

```
layersTransfer = net. Layers (1: end-3);
```

In our case, we transferred the layers to the new classification task by replacing the last three layers with a fully connected layer, a softmax layer, and a classification output layer. Specify the options of the new fully connected layer according to the new data. Set the fully connected layer to have the same size as the number of classes in the new data. To learn faster in the new layers than in the transferred layers, increase the WeightLearnRateFactor and BiasLearnRateFactor values of the fully connected layer.

```
Layers = [

   layersTransfer

   fullyConnectedLayer (numClasses,'WeightLearnRateFactor', 20,'BiasLearnRateFactor', 20)

   softmaxLayer

   classificationLayer];
```

3.2.2 Training the network

While specifying the training options for transfer learning, we kept the features from earliest layers of the pre-trained network (the transferred layer weights). We set ‘InitialLearnRate’ to a small value to slow down learning in the transferred layers. In the previous step, we increased the learning rate factors for the fully connected layer to speed up learning in the new final layers. This combination of learning rate settings results in faster learning only in the new layers and slower learning in the other layers. An epoch is a full training cycle on the entire training data set. Specifying the mini-batch size and validation data. The software validates the network every ValidationFrequency iteration during training and automatically stops training if the validation loss stops improving. Validate the network once per epoch.

```
miniBatchSize = 10;

numIterationsPerEpoch = floor (Numel (trainingImages. Labels) /miniBatchSize);

Options = trainingOptions ('sgdm',...

    'MiniBatchSize', miniBatchSize,...

    'MaxEpochs,' 4,...

    'InitialLearnRate', 1e-4,...

    'Verbose,' false,...

    'Plots','training-progress',...

    'Validation Data', validationImages,...

    'ValidationFrequency', numIterationsPerEpoch);
```

3.2.3 Result

```
Training on a single CPU.

Initializing image normalization.

|=====================================================================================
=====|

|   Epoch   |  Iteration  | Time Elapsed |  Mini-batch  |  Mini-batch  | Base Learning|

|           |             |  (Seconds)   |     Loss     |   Accuracy   |     Rate     |

|=====================================================================================
=====|

|         1 |           1 |       579.22 |       0.6818 |       57.00% |     1.00e-04 |

|         5 |           5 |      2875.54 |       0.6860 |       64.00% |     1.00e-04 |
|==========================================================================
=====|
ans =
   0.6430
```

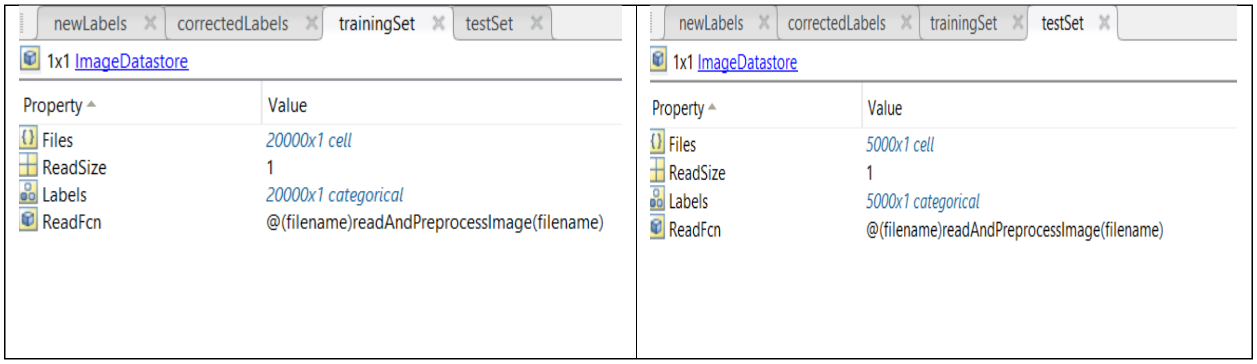
We run the above Transfer Learning solution only for 1000 images among them 500 images are run as a training set, and the remaining 500 images are run as a test set. We got low accuracy (64%) because we used only a small part of the whole dataset.

3.3 Solution 3: Use CNN as feature extraction block

A pre-trained CNN model (e.g., AlexNet) in which an appropriate (fully connected, in our case, it is 'fc7' layer) layer activations are used as features, which are then used by a conventional machine learning classifier (e.g., SVM, logistic regression, or decision tree, among others). In this project work, we used Quadratic Support Vector Machine (SVM) to classify the images. The detailed explanation of the workflow has been described in the later section.

3.3.1 Partitioning Dataset

We used 60% of total data as Training Dataset (15,000 images), 20% as Cross Validation (5,000 images) and the rest 20% as Testing Dataset (5,000) (Figure 7). The whole dataset was partitioned 80/20 ratio, to begin with, and 80% were used by the Classification Learner App. The Classification Learner App's Holdout Validation method (Figure 8) then partitioned the 80% of data into two sets -> 75% Training and 25% Holdout Validation. This has produced the desired partitions on 60/20/20 for Training/Validation/Testing.



1x1 ImageDatastore

Property	Value
Files	20000x1 cell
ReadSize	1
Labels	20000x1 categorical
ReadFcn	@(filename)readAndPreprocessImage(filename)

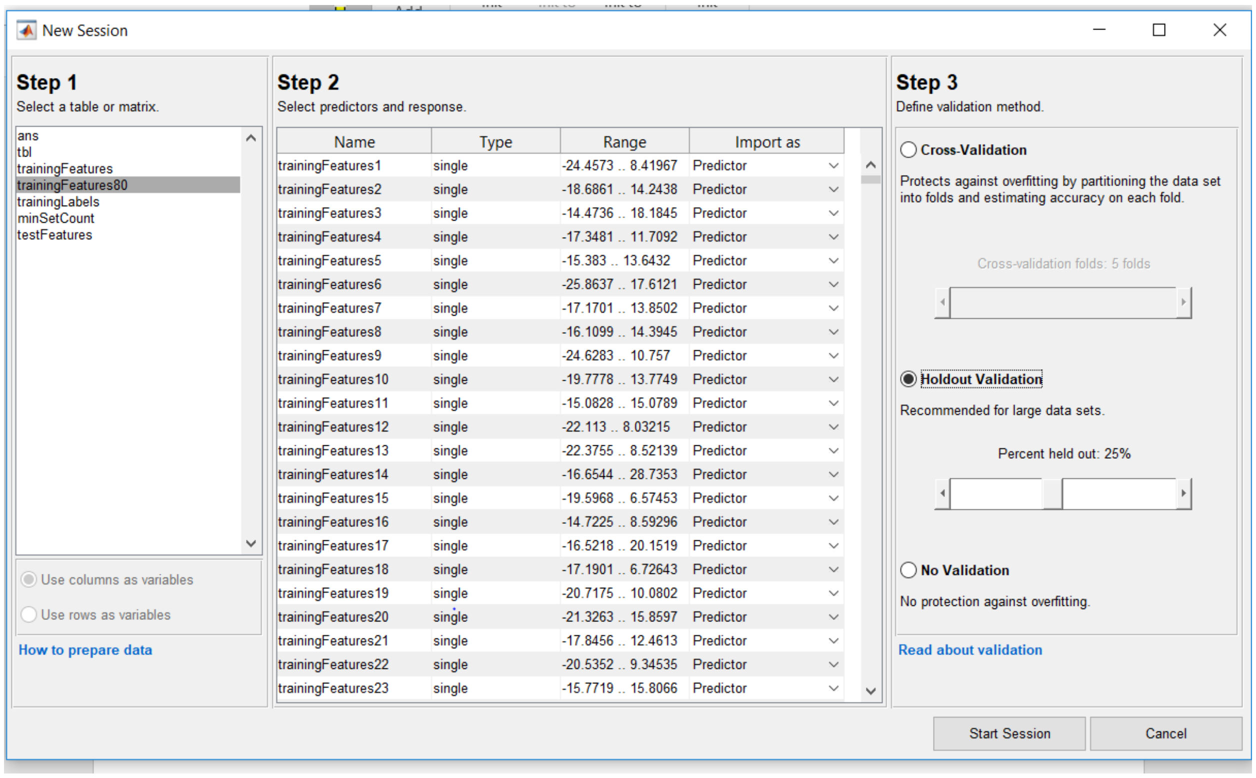
1x1 ImageDatastore

Property	Value
Files	5000x1 cell
ReadSize	1
Labels	5000x1 categorical
ReadFcn	@(filename)readAndPreprocessImage(filename)

Figure 7: partitioning training and testing dataset

3.3.1.1 Holdout Validation

Holdout Validation is a validation method in the Classification Learner App in MATLAB which allow us to select a percentage of the data to use as a test set using the slider control. In our case, 25% of the training data set as Holdout Validation (Figure 8) by which we can eventually achieve 20% of the total dataset as a validation set (5,000 images). The app trains a model on the training set and assesses its performance with the test set. The model used for validation is based on only a portion of the data, so Holdout Validation is recommended only for large data sets. The final model is trained with the full data set.


New Session

Step 1
Select a table or matrix.

ans
tbl
trainingFeatures
trainingFeatures80
trainingLabels
minSetCount
testFeatures

Use columns as variables
Use rows as variables

How to prepare data

Step 2
Select predictors and response.

Name	Type	Range	Import as
trainingFeatures1	single	-24.4573 .. 8.41967	Predictor
trainingFeatures2	single	-18.6861 .. 14.2438	Predictor
trainingFeatures3	single	-14.4736 .. 18.1845	Predictor
trainingFeatures4	single	-17.3481 .. 11.7092	Predictor
trainingFeatures5	single	-15.383 .. 13.6432	Predictor
trainingFeatures6	single	-25.8637 .. 17.6121	Predictor
trainingFeatures7	single	-17.1701 .. 13.8502	Predictor
trainingFeatures8	single	-16.1099 .. 14.3945	Predictor
trainingFeatures9	single	-24.6283 .. 10.757	Predictor
trainingFeatures10	single	-19.7778 .. 13.7749	Predictor
trainingFeatures11	single	-15.0828 .. 15.0789	Predictor
trainingFeatures12	single	-22.113 .. 8.03215	Predictor
trainingFeatures13	single	-22.3755 .. 8.52139	Predictor
trainingFeatures14	single	-16.6544 .. 28.7353	Predictor
trainingFeatures15	single	-19.5968 .. 6.57453	Predictor
trainingFeatures16	single	-14.7225 .. 8.59296	Predictor
trainingFeatures17	single	-16.5218 .. 20.1519	Predictor
trainingFeatures18	single	-17.1901 .. 6.72643	Predictor
trainingFeatures19	single	-20.7175 .. 10.0802	Predictor
trainingFeatures20	single	-21.3263 .. 15.8597	Predictor
trainingFeatures21	single	-17.8456 .. 12.4613	Predictor
trainingFeatures22	single	-20.5352 .. 9.34535	Predictor
trainingFeatures23	single	-15.7719 .. 15.8066	Predictor

Step 3
Define validation method.

Cross-Validation
Protects against overfitting by partitioning the data set into folds and estimating accuracy on each fold.
Cross-validation folds: 5 folds

[✓ **Holdout Validation**]
Recommended for large data sets.
Percent held out: 25%

No Validation
No protection against overfitting.

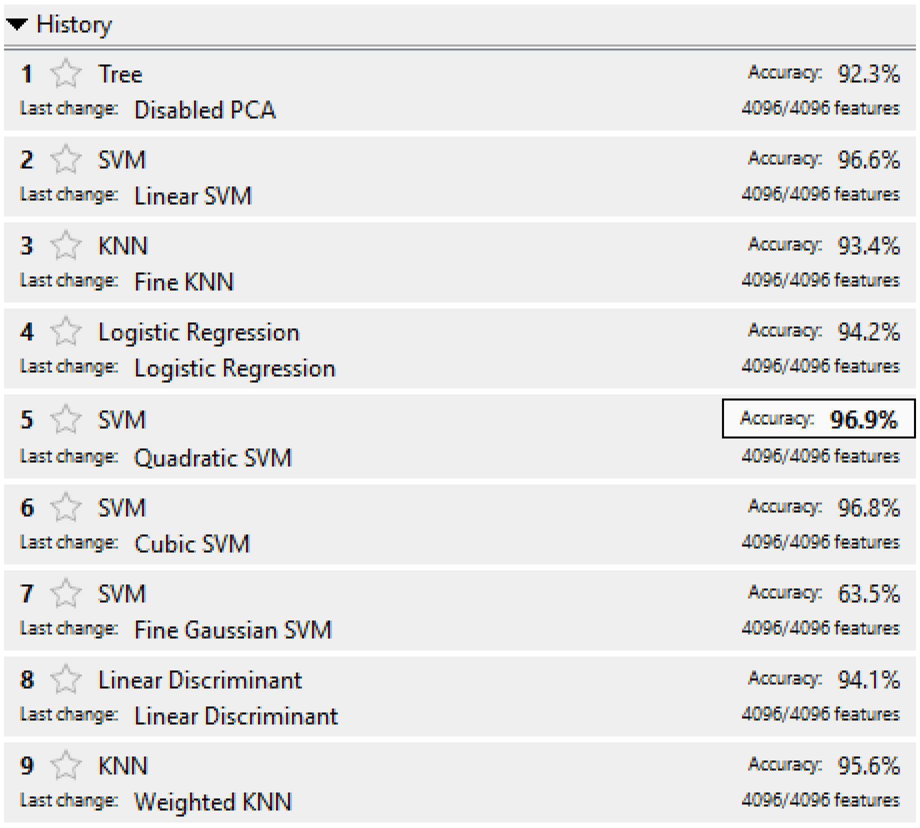
Read about validation

Start Session   Cancel

Figure 8: Holdout Validation in Classification Learner App

3.3.2 Train the classifier in Classification Learner App

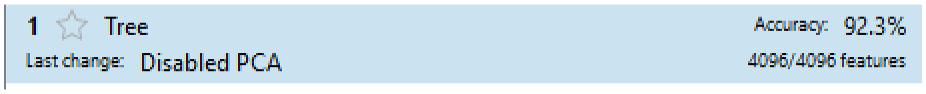
We used a Pre-Trained CNN to extract features from image data: AlexNet. Then we used Classification Learner App in MATLAB to train different models (classifier, e.g., SVM, logistic regression, decision tree, KNN, a linear discriminant, etc.) with the 4096 features provided by AlexNet. After training each model with 4096 features, the Classification Learner App's 'history' section generates validation accuracy for each trained model. In our case, we found Quadratic SVM classifier with the highest accuracy 96.9%. All the models with their accuracy have been listed in the below figure.



History

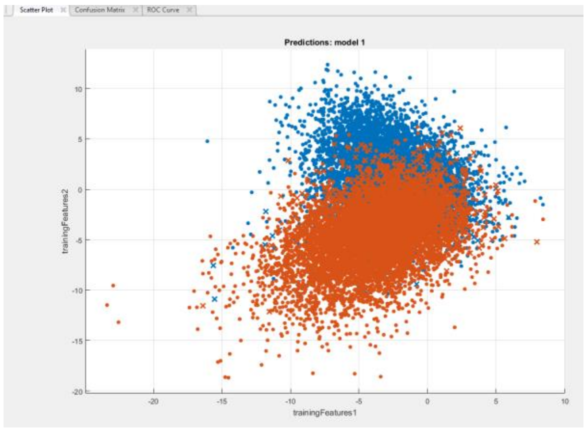
	Model	Last change	Accuracy	Features
1	Tree	Disabled PCA	92.3%	4096/4096 features
2	SVM	Linear SVM	96.6%	4096/4096 features
3	KNN	Fine KNN	93.4%	4096/4096 features
4	Logistic Regression	Logistic Regression	94.2%	4096/4096 features
5	SVM	Quadratic SVM	**96.9%**	4096/4096 features
6	SVM	Cubic SVM	96.8%	4096/4096 features
7	SVM	Fine Gaussian SVM	63.5%	4096/4096 features
8	Linear Discriminant	Linear Discriminant	94.1%	4096/4096 features
9	KNN	Weighted KNN	95.6%	4096/4096 features

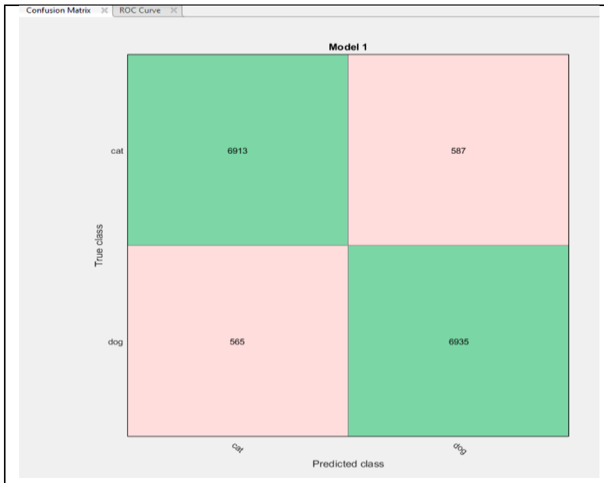
Here we attached the screenshots of the confusion matrix and ROC curve for each classifier trained with the 4096 features provided by AlexNet.

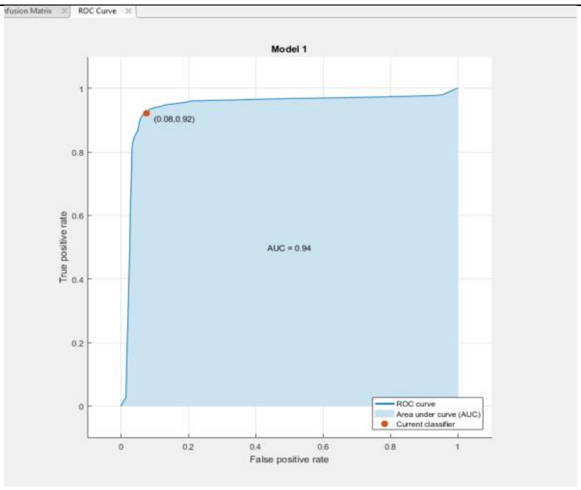


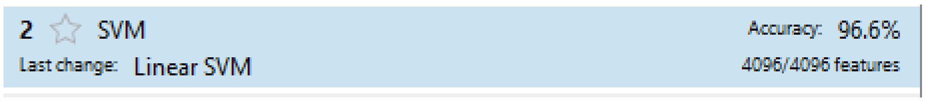
1 Tree — Last change: Disabled PCA — Accuracy: 92.3% — 4096/4096 features

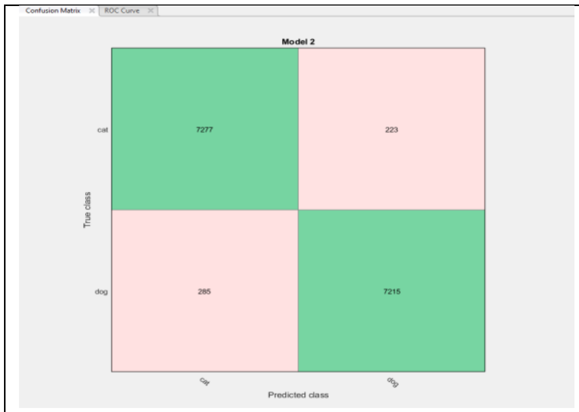
Below are the scatter plot, confusion matrix and ROC curve for decision tree.

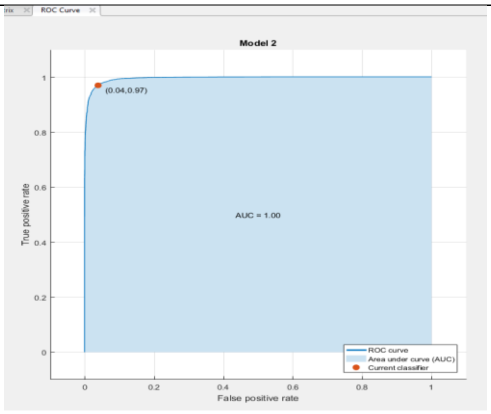


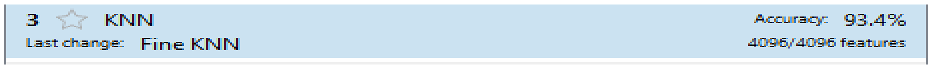


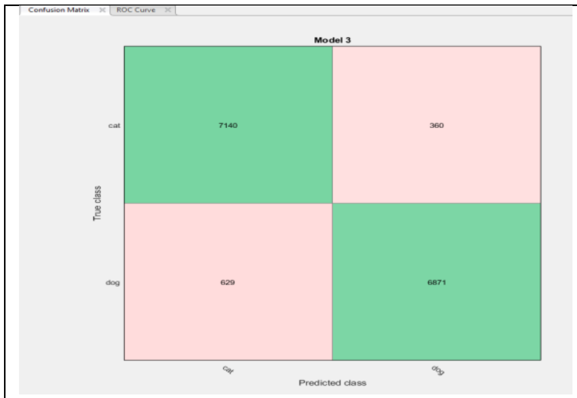


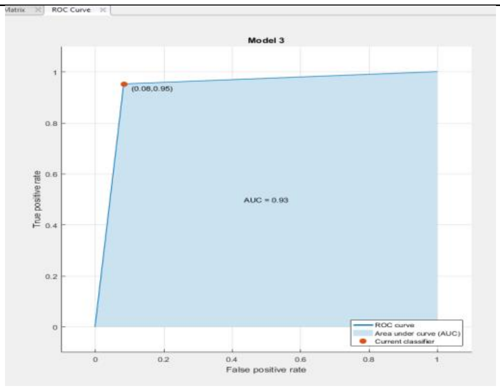

2 SVM
Last change: Linear SVM
Accuracy: 96.6%
4096/4096 features

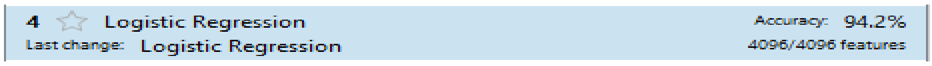


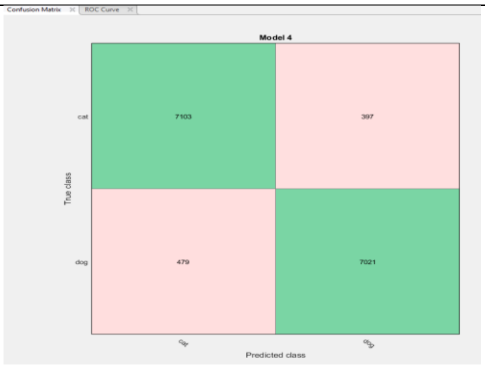


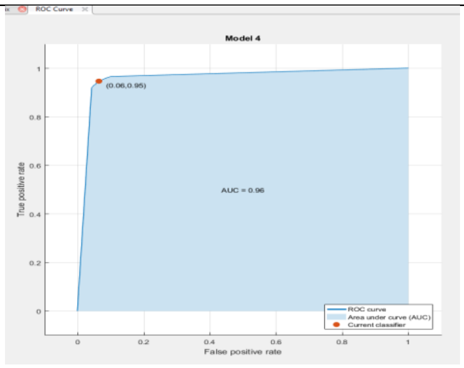

3 KNN Accuracy: 93.4%
Last change: Fine KNN 4096/4096 features

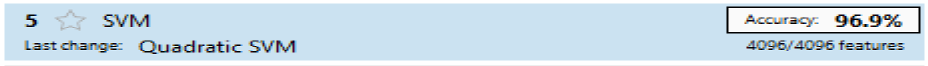


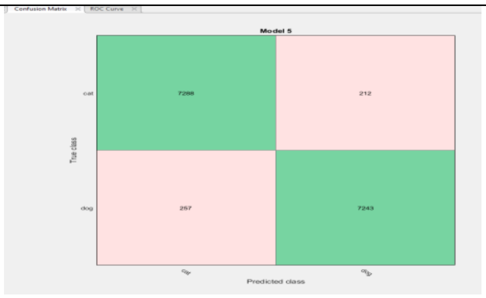


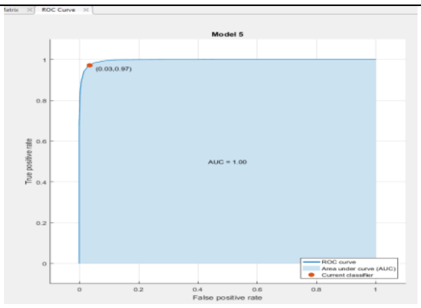

4 Logistic Regression Accuracy: 94.2%
Last change: Logistic Regression 4096/4096 features

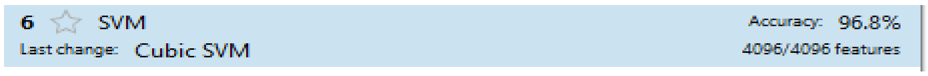


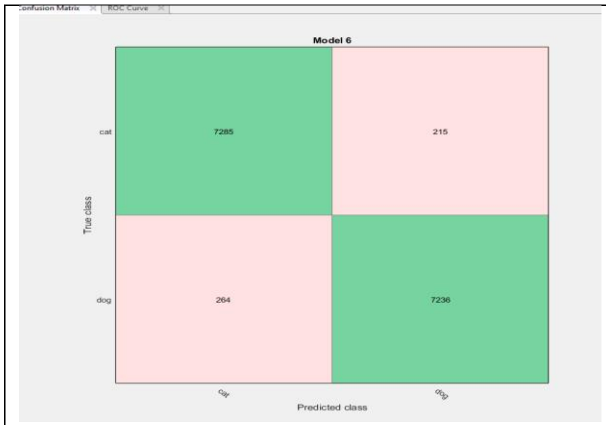


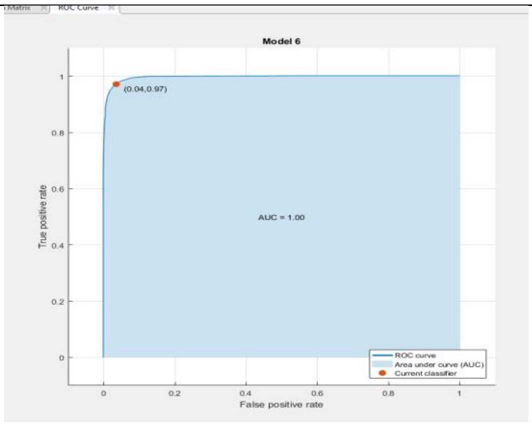

5 SVM Accuracy: 96.9%
Last change: Quadratic SVM 4096/4096 features

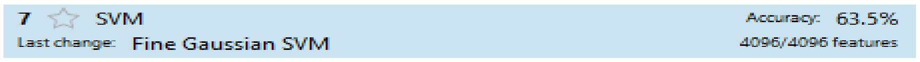


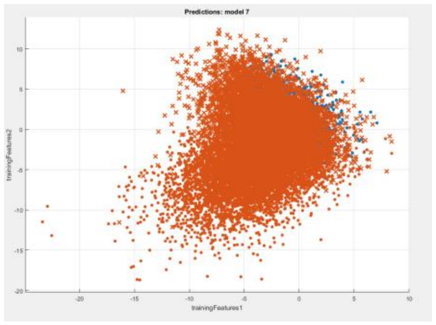


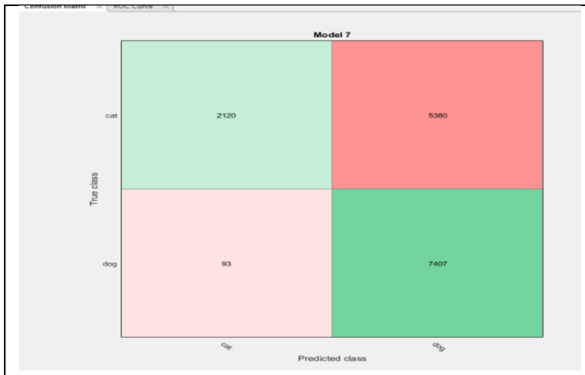

6 SVM
Last change: Cubic SVM
Accuracy: 96.8%
4096/4096 features

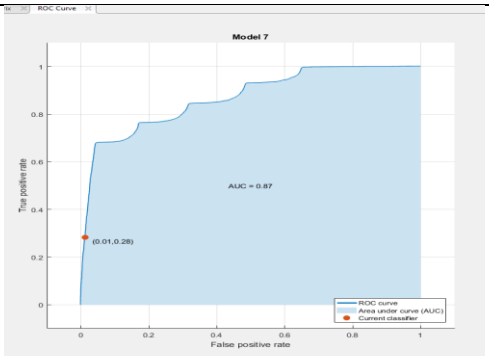


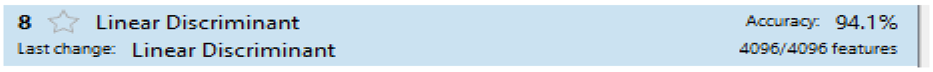


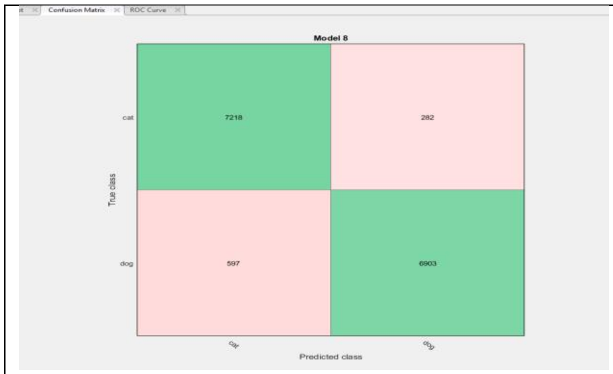

7 SVM
Last change: Fine Gaussian SVM
Accuracy: 63.5%
4096/4096 features

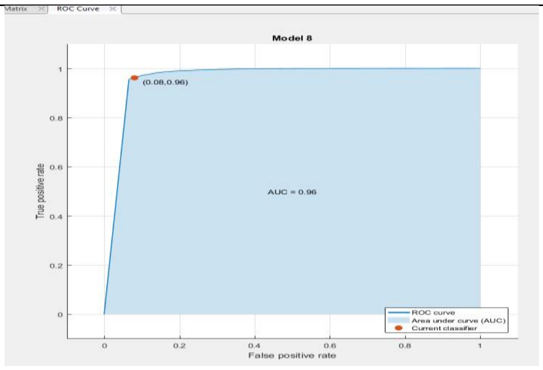


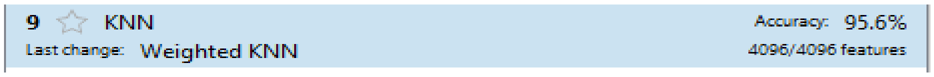


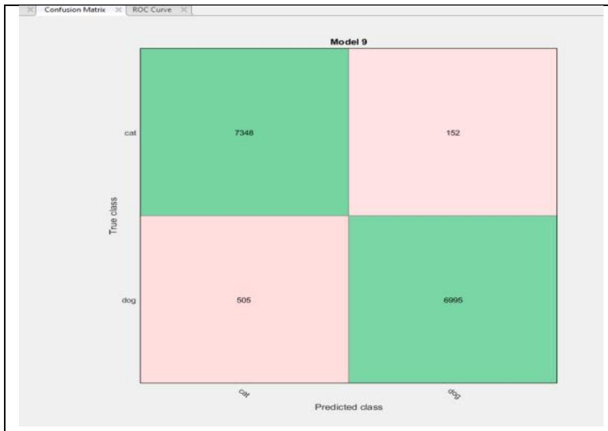


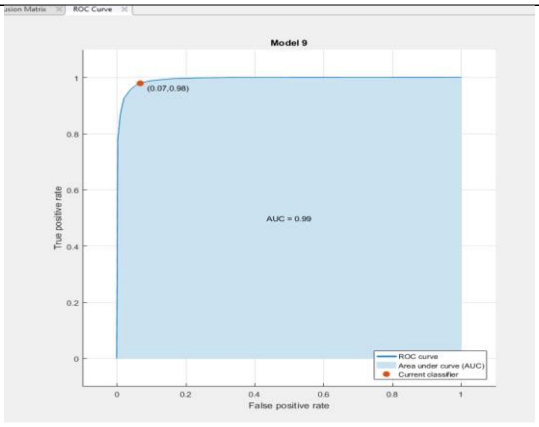

8 Linear Discriminant
Last change: Linear Discriminant
Accuracy: 94.1%
4096/4096 features



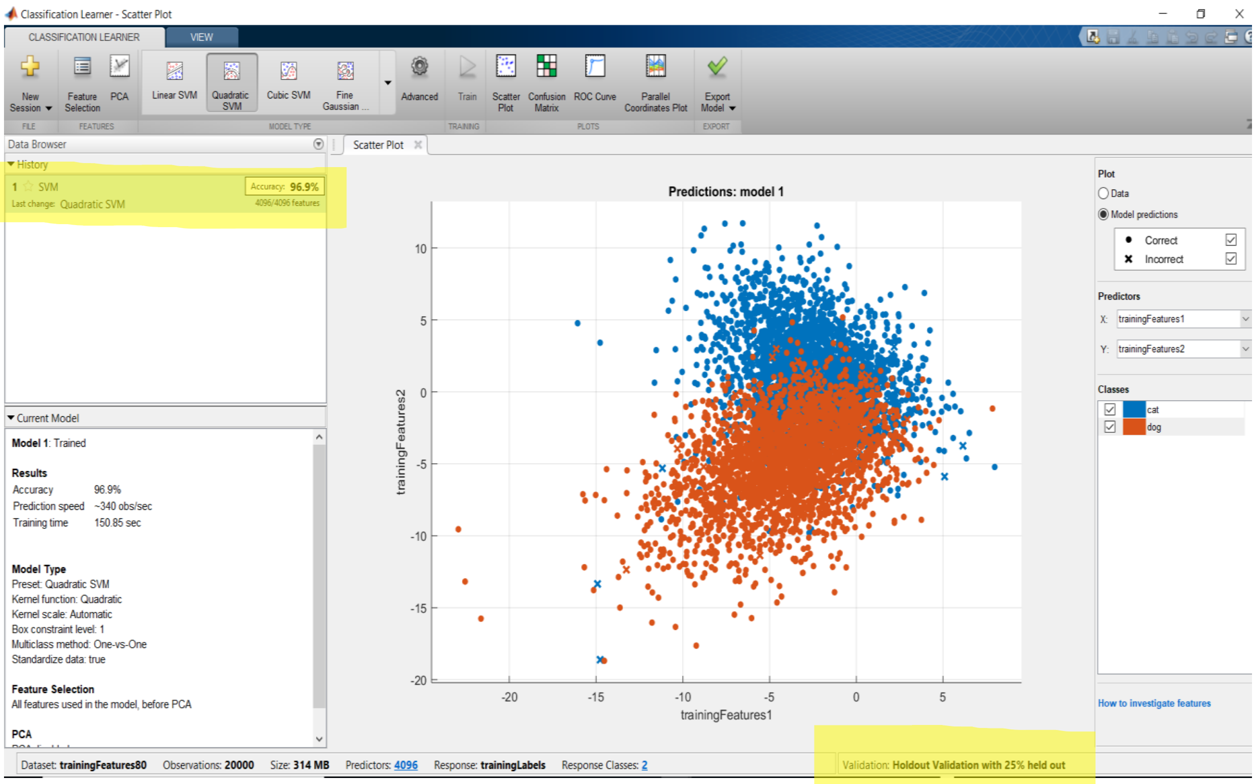



9 KNN
Last change: Weighted KNN
Accuracy: 95.6%
4096/4096 features





We selected Quadratic SVM as our final classier to validate on the testing dataset as it is the highest accuracy classifier with validation accuracy 96.9%. The below images show the scatter plot, confusion matrix and ROC curve from Classification Learner App for Quadratic SVM classifier.



History

1 SVM Accuracy: 96.9%

Last change: Quadratic SVM 4096/4096 features

Current Model

Model 1: Trained

Results
Accuracy 96.9%
Prediction speed ~340 obs/sec
Training time 150.85 sec

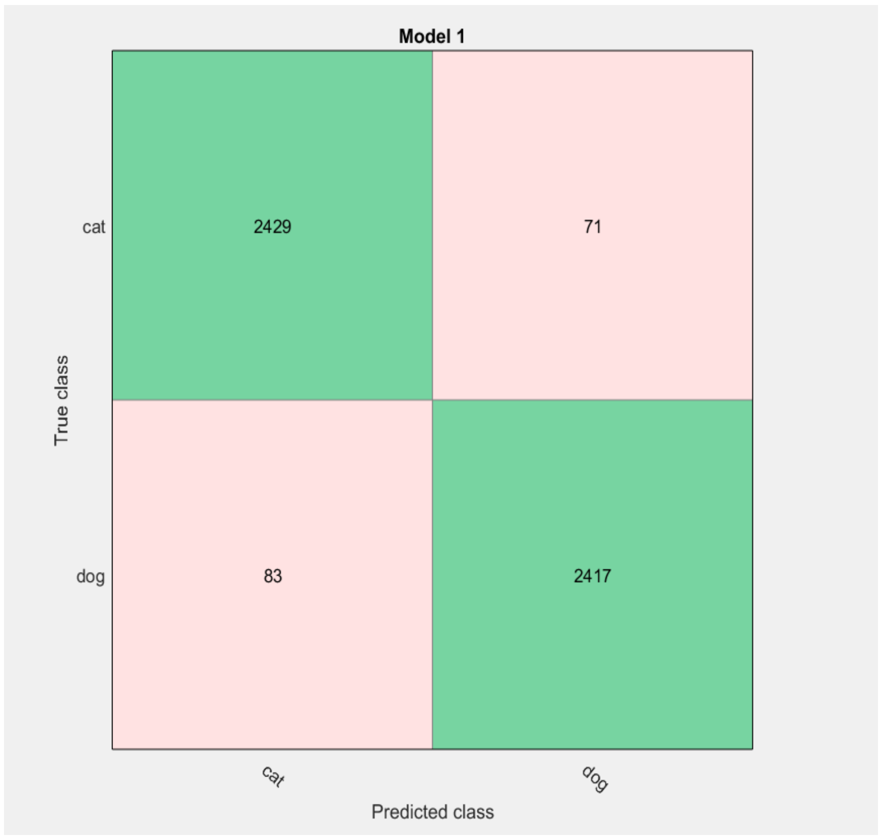
Model Type
Preset: Quadratic SVM
Kernel function: Quadratic
Kernel scale: Automatic
Box constraint level: 1
Multiclass method: One-vs-One
Standardize data: true

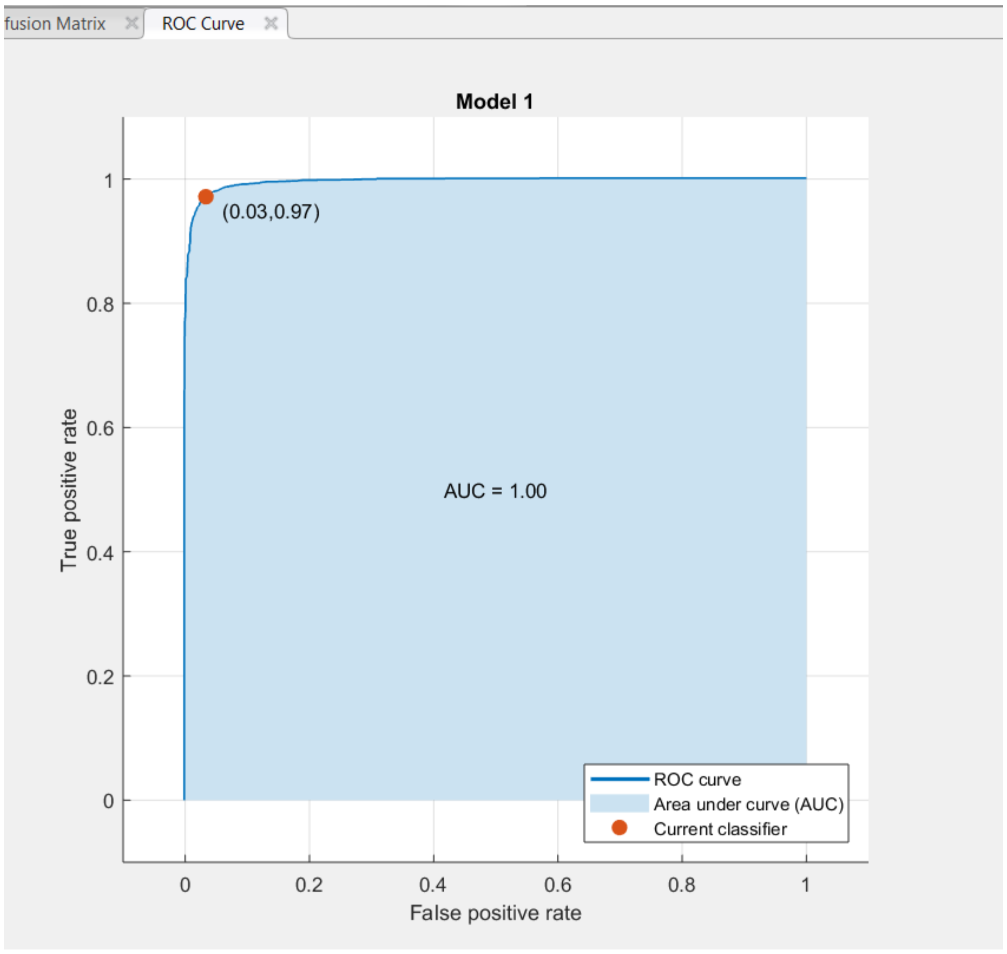
Feature Selection
All features used in the model, before PCA

PCA

Dataset: **trainingFeatures80** Observations: **20000** Size: **314 MB** Predictors: **4096** Response: **trainingLabels** Response Classes: **2**

Validation: **Holdout Validation with 25% held out**





3.3.3 Perform Cross-Validation and check accuracy

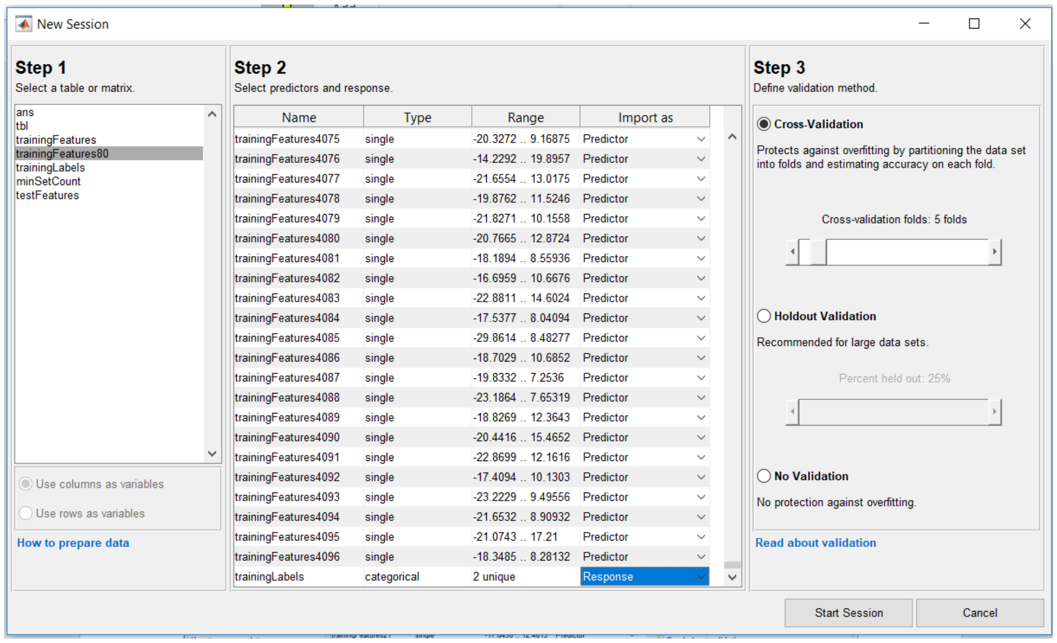
We used both Holdout Validation and Cross-Validation as a validation method in Classification Learner App to check the validation accuracy of the trained model. Here we used 10-fold cross validation on the classification object of our trained SVM model 'trainedSVMModel'. By-default Classification Learner App use 5-fold cross validation (Figure 9).

```
>> cvmdl = crossval (trainedSVMModel. ClassificationSVM,'KFold', 10);

>> fprintf ('fold CV accuracy: %2.2f\n',1-cvmdl.kfoldLoss);

Fold CV accuracy: 0.97
```

We got 97% accuracy for 10-fold cross-validation.


New Session

Step 1
Select a table or matrix.

ans
tbl
trainingFeatures
trainingFeatures80
trainingLabels
minSetCount
testFeatures

Use columns as variables
Use rows as variables

How to prepare data

Step 2
Select predictors and response.

Name	Type	Range	Import as
trainingFeatures4075	single	-20.3272 .. 9.16875	Predictor
trainingFeatures4076	single	-14.2292 .. 19.8957	Predictor
trainingFeatures4077	single	-21.6554 .. 13.0175	Predictor
trainingFeatures4078	single	-19.8762 .. 11.5246	Predictor
trainingFeatures4079	single	-21.8271 .. 10.1558	Predictor
trainingFeatures4080	single	-20.7665 .. 12.8724	Predictor
trainingFeatures4081	single	-18.1894 .. 8.55936	Predictor
trainingFeatures4082	single	-16.6959 .. 10.6676	Predictor
trainingFeatures4083	single	-22.8811 .. 14.6024	Predictor
trainingFeatures4084	single	-17.5377 .. 8.04094	Predictor
trainingFeatures4085	single	-29.8614 .. 8.48277	Predictor
trainingFeatures4086	single	-18.7029 .. 10.6852	Predictor
trainingFeatures4087	single	-19.8332 .. 7.2536	Predictor
trainingFeatures4088	single	-23.1864 .. 7.65319	Predictor
trainingFeatures4089	single	-18.8269 .. 12.3643	Predictor
trainingFeatures4090	single	-20.4416 .. 15.4652	Predictor
trainingFeatures4091	single	-22.8699 .. 12.1616	Predictor
trainingFeatures4092	single	-17.4094 .. 10.1303	Predictor
trainingFeatures4093	single	-23.2229 .. 9.49556	Predictor
trainingFeatures4094	single	-21.6532 .. 8.90932	Predictor
trainingFeatures4095	single	-21.0743 .. 17.21	Predictor
trainingFeatures4096	single	-18.3485 .. 8.28132	Predictor
trainingLabels	categorical	2 unique	Response

Step 3
Define validation method.

● Cross-Validation
Protects against overfitting by partitioning the data set into folds and estimating accuracy on each fold.
Cross-validation folds: 5 folds

○ Holdout Validation
Recommended for large data sets.
Percent held out: 25%

○ No Validation
No protection against overfitting.

Read about validation

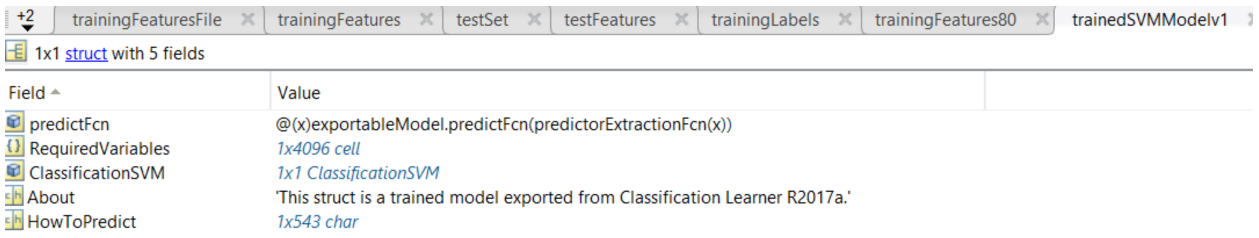
Start Session
Cancel

Figure 9: Cross-Validation in Classification Learners App

3.3.4 Exporting the trained model into the workspace

Next, we picked the classifier with the highest accuracy, i.e., Quadratic SVM classifier (i.e., Our trained model) and export the model into our workspace as ‘trainedSVMModel. mat’ to evaluate on our test dataset (remaining 20% of the data).

‘trainedClassifier’ is a structure containing the trained classifier. The struct contains various fields with information about the trained classifier. The below is the structure of the trained SVM model exported from Classification Learner App- ‘trainedSVMModel. mat’.

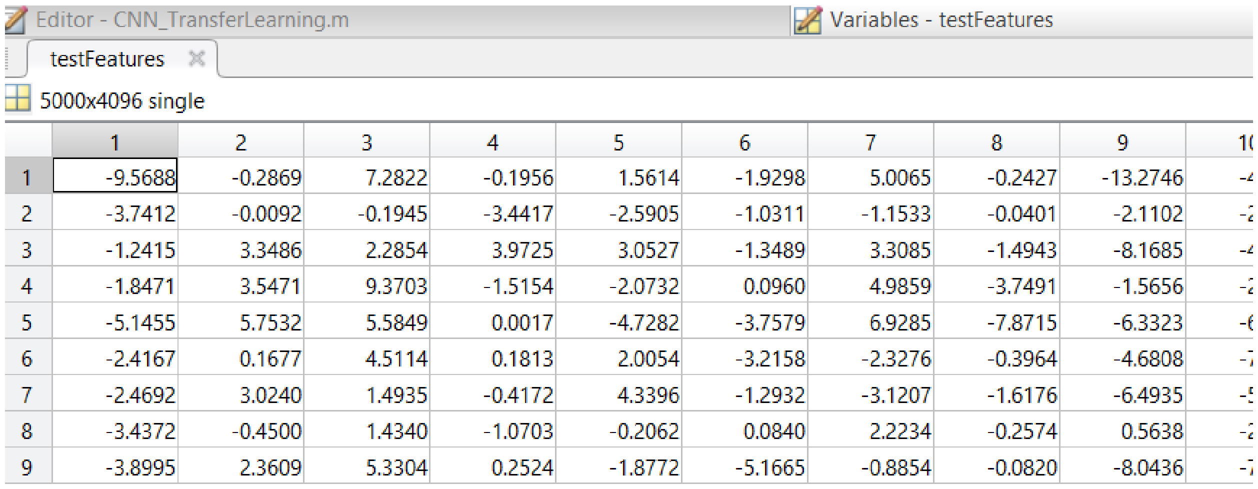

trainingFeaturesFile | trainingFeatures | testSet | testFeatures | trainingLabels | trainingFeatures80 | trainedSVMModelv1

1x1 struct with 5 fields

Field	Value
predictFcn	@(x)exportableModel.predictFcn(predictorExtractionFcn(x))
RequiredVariables	1x4096 cell
ClassificationSVM	1x1 ClassificationSVM
About	'This struct is a trained model exported from Classification Learner R2017a.'
HowToPredict	1x543 char

3.3.5 Evaluate classifier with test dataset

In this step, first we have extracted features from the test dataset using pre-trained CNN Alexnet, and 4096 feature had been extracted in total, and we saved the features in a mat file called 'testFeatures. mat'. It is a 5000-by-4096 array matrix.



Editor - CNN_TransferLearning.m | Variables - testFeatures

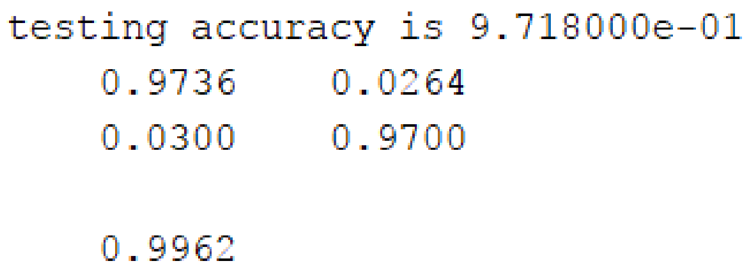
testFeatures

5000x4096 single

	1	2	3	4	5	6	7	8	9	10
1	-9.5688	-0.2869	7.2822	-0.1956	1.5614	-1.9298	5.0065	-0.2427	-13.2746	-4
2	-3.7412	-0.0092	-0.1945	-3.4417	-2.5905	-1.0311	-1.1533	-0.0401	-2.1102	-2
3	-1.2415	3.3486	2.2854	3.9725	3.0527	-1.3489	3.3085	-1.4943	-8.1685	-4
4	-1.8471	3.5471	9.3703	-1.5154	-2.0732	0.0960	4.9859	-3.7491	-1.5656	-2
5	-5.1455	5.7532	5.5849	0.0017	-4.7282	-3.7579	6.9285	-7.8715	-6.3323	-6
6	-2.4167	0.1677	4.5114	0.1813	2.0054	-3.2158	-2.3276	-0.3964	-4.6808	-7
7	-2.4692	3.0240	1.4935	-0.4172	4.3396	-1.2932	-3.1207	-1.6176	-6.4935	-5
8	-3.4372	-0.4500	1.4340	-1.0703	-0.2062	0.0840	2.2234	-0.2574	0.5638	-2
9	-3.8995	2.3609	5.3304	0.2524	-1.8772	-5.1665	-0.8854	-0.0820	-8.0436	-7

3.3.5.1 Computing test accuracy, confusion matrix, and computing AUC

Below is the screenshot of result from MATLAB after running the trained classifier on the test dataset.



```
testing accuracy is 9.718000e-01
    0.9736    0.0264
    0.0300    0.9700

    0.9962
```

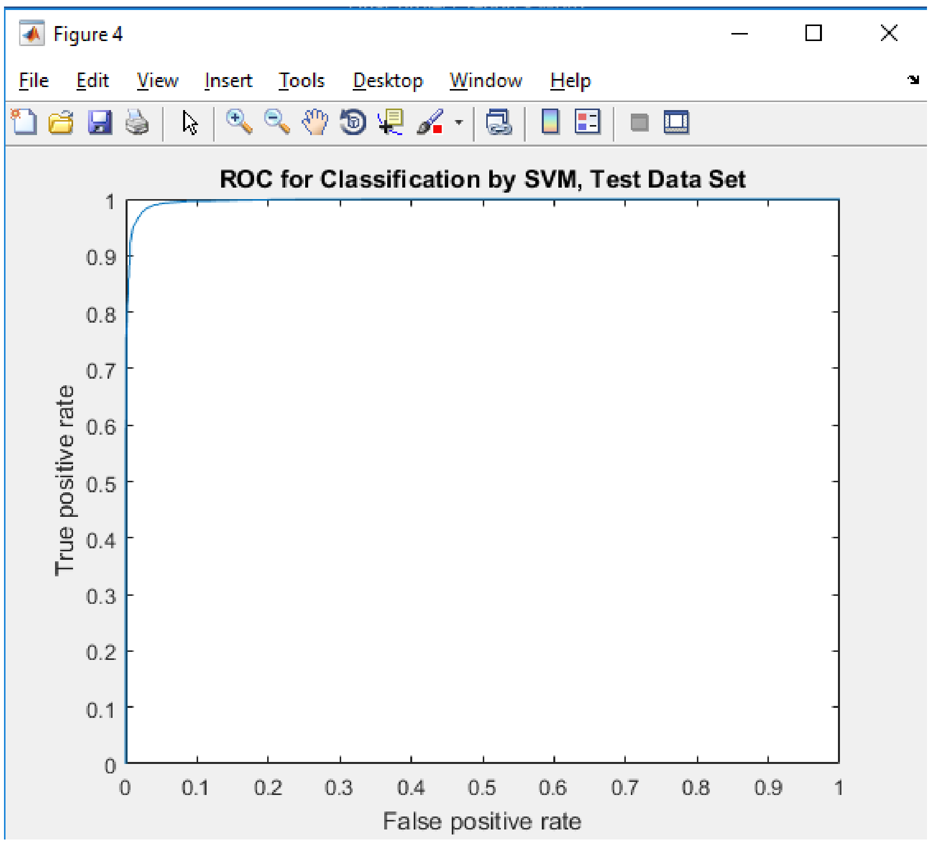
We got 97.18% test accuracy with our trained model on the test dataset.

Confusion matrix in percentage form:

$$\begin{bmatrix} 0.9736 & 0.0264 \\ 0.0300 & 0.9700 \end{bmatrix}$$

AUC value is 0.9962.

The ROC curve for classification by SVM on a test dataset has shown below.



4. Tabulation of Final Results

This table illustrates the performance rates of three different solutions we used to solve the cats vs dogs classification problem.

Solutions	Number of images	Accuracy
1) Pre-trained CNN AlexNet used "as is"	25,000	**~80%**
2) Transfer Learning using CNN	500 as training set, 500 as test set	**~ 64%**
3) CNN as feature extraction block	60% as a training set, 20% as a validation set and 20% as a test set of 25,000 images.	**~ 97%**

5. Conclusions

It has been observed that the highest accuracy of the cats Vs. dogs' classification problem solved by a classifier using machine learning algorithms is 97%. The rest 3% is classified wrongly. To examine this data, we could say that AI is unable to resolve the overall problems with 100% accuracy. It has also been noted that the "duration" which machine has needed to classify the images based on their features is larger than as expected.

If we compare the image classification accuracy of machine with a human by extracting the features, human could be more superior than machine until this date. In addition to that, we could say that human takes lesser time to classify images. We envision that our findings could give a motivation to neuroscientists to find mechanisms happened in the brain, which could facilitate this process more accurately.

References:

J. Elson, J. Douceur, J. Howell and J. Saul. Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. Proc. of ACM CCS 2007, pp. 366-374.

Freedman, D. J., Riesenhuber, M., Poggio, T., & Miller, E. K. (2003). A comparison of primate prefrontal and inferior temporal cortices during visual categorization. *Journal of Neuroscience*, *23* (12), 5235-5246.

Freedman, D. J., Riesenhuber, M., Poggio, T., & Miller, E. K. (2005). Experience-dependent sharpening of visual shape selectivity in the inferior temporal cortex. *Cerebral Cortex*, *16* (11), 1631-1644.

Cromer, J. A., Roy, J. E., & Miller, E. K. (2010). Representation of multiple, independent categories in the primate prefrontal cortex. *Neuron*, *66* (5), 796-807.

Roy, J. E., Riesenhuber, M., Poggio, T., & Miller, E. K. (2010). Prefrontal cortex activity during flexible categorization. *Journal of Neuroscience*, *30* (25), 8519-8528. excitation of multiple, independent categories in the primate prefrontal cortex. *Neuron*, *66* (5), 796-807.

Minamimoto, T., Saunders, R. C., & Richmond, B. J. (2010). The monkeys quickly learn and generalize visual categories without lateral prefrontal cortex. *Neuron*, *66* (4), 501-507.

Minamimoto, T., Saunders, R. C., & Richmond, B. J. (2010). The monkeys quickly learn and generalize visual categories without lateral prefrontal cortex. *Neuron*, *66* (4), 501-507.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86 (11), 2278-2324.

Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). Imagined classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems 25 (pp. 1106-1114).

Source URLs:

https://www.pyimagesearch.com/2016/06/27/my-top-9-favorite-python-deep-learning-libraries/

http://image-net.org/index

https://www.kaggle.com/datasets

https://sites.ualberta.ca/~bang3/files/DogCat_report.pdf https://www.mathworks.com/matlabcentral/fileexchange/59133-neural-network-toolbox-tm--model-for-alexnet-network

http://cs231n.github.io/convolutional-networks/

https://sites.ualberta.ca/~bang3/files/DogCat_report.pdf

https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/