Team Indiscriminant Analysis Final Project: Cats vs Dogs

Featuring:
Safwat Rahman
Matt Rublee
Ángel Silvar
Aaron Barel

Overview

- Safwat - Principal Component Analysis & Kohonen’s Novelty
- Matt - Linear Discriminant Analysis
- Ángel - Deep Neural Network
- Aaron - Convolutional Neural Network, Support Vector Machine

Principal Component Analysis

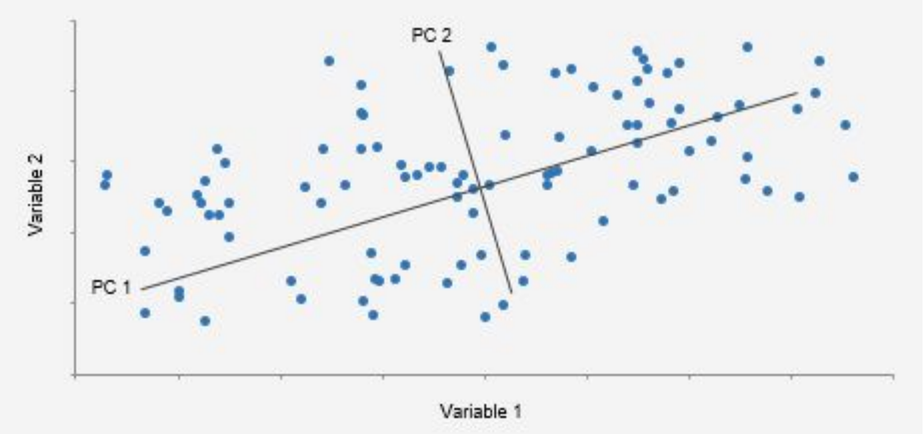
Goal: Reduce Dimensionality

Steps:

Center Data (mean subtract)

Project to new coordinates/dimensions

Select Dimensions to maximize variance



Overview

Mean subtract raw training data

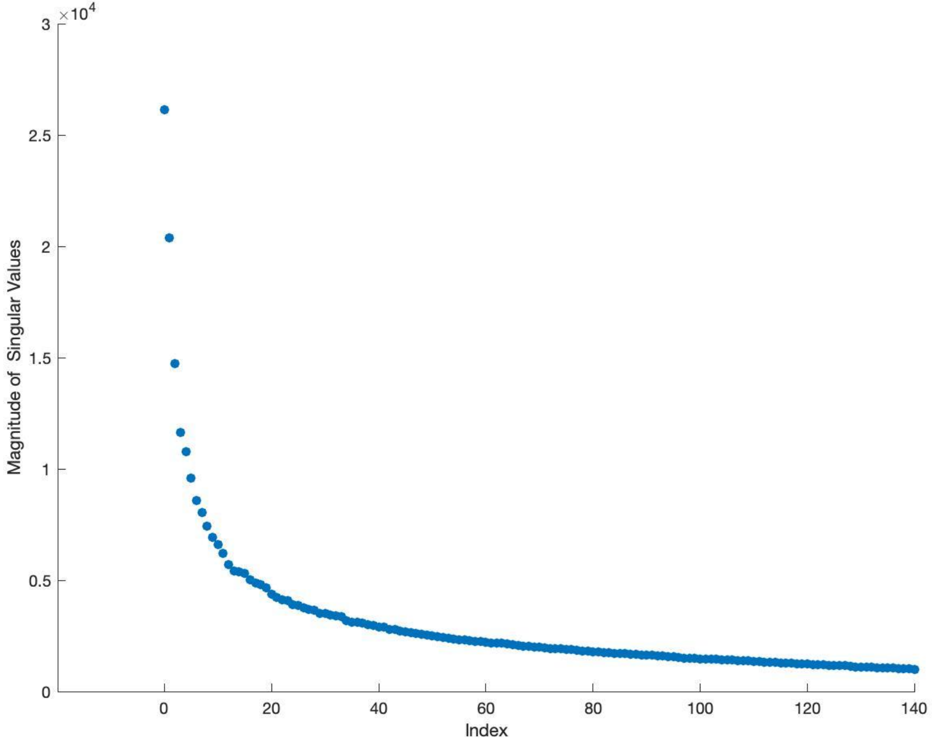
Run SVD

Select value for Dimension reduction

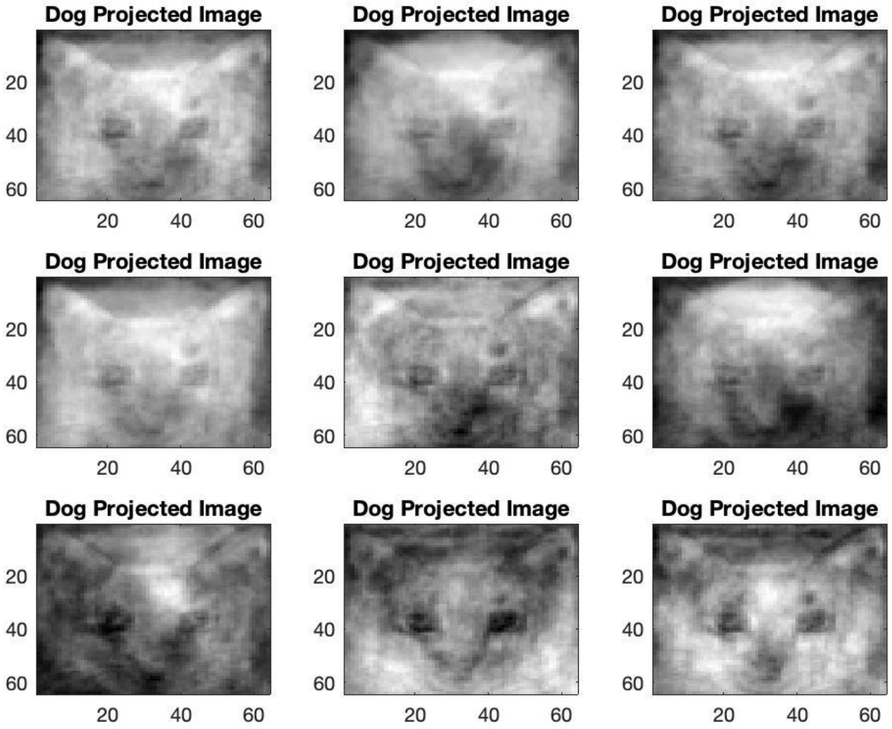
Test Data

Dog & Cat Coefficients and Principal Components

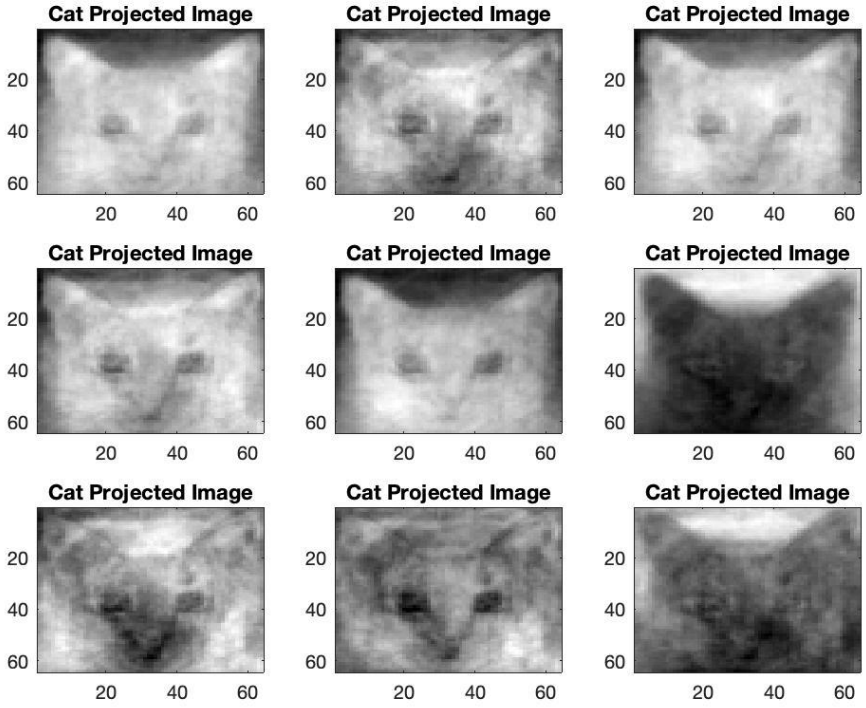
Reconstruct Images



Dog Reconstruction

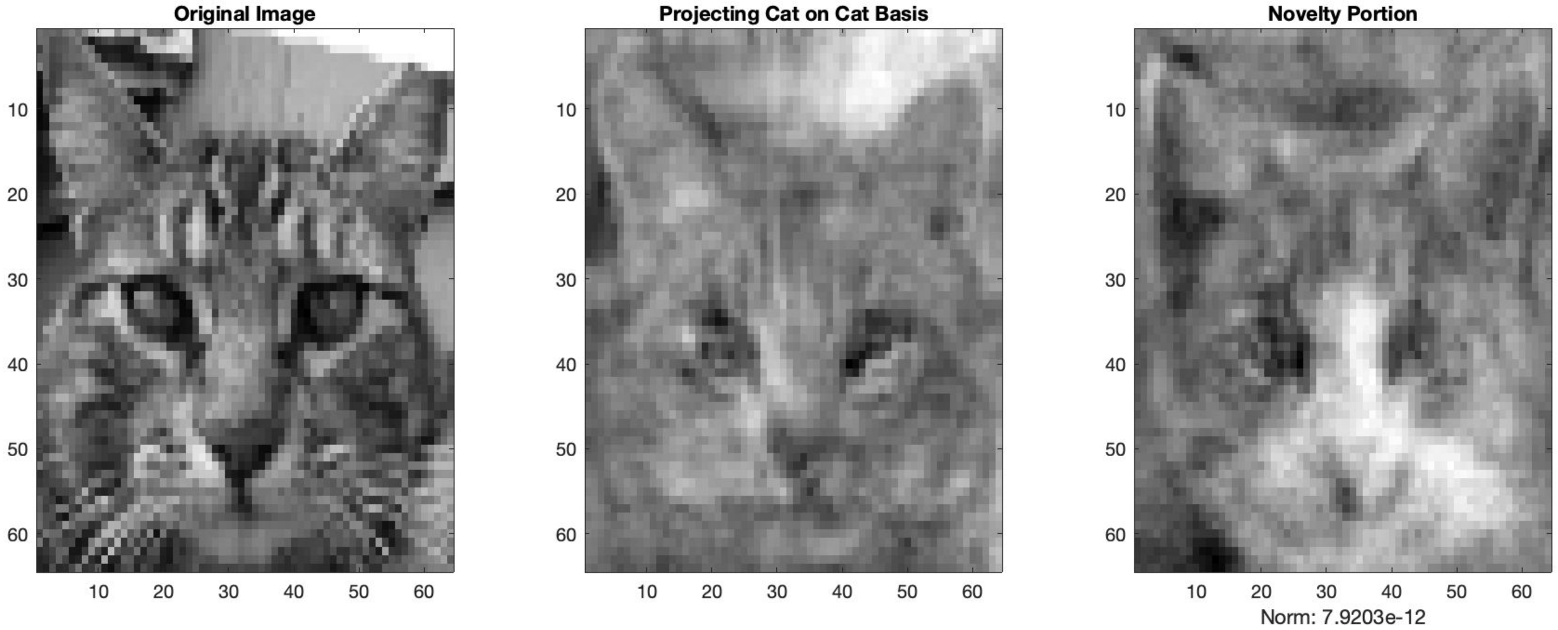


Cat Reconstruction

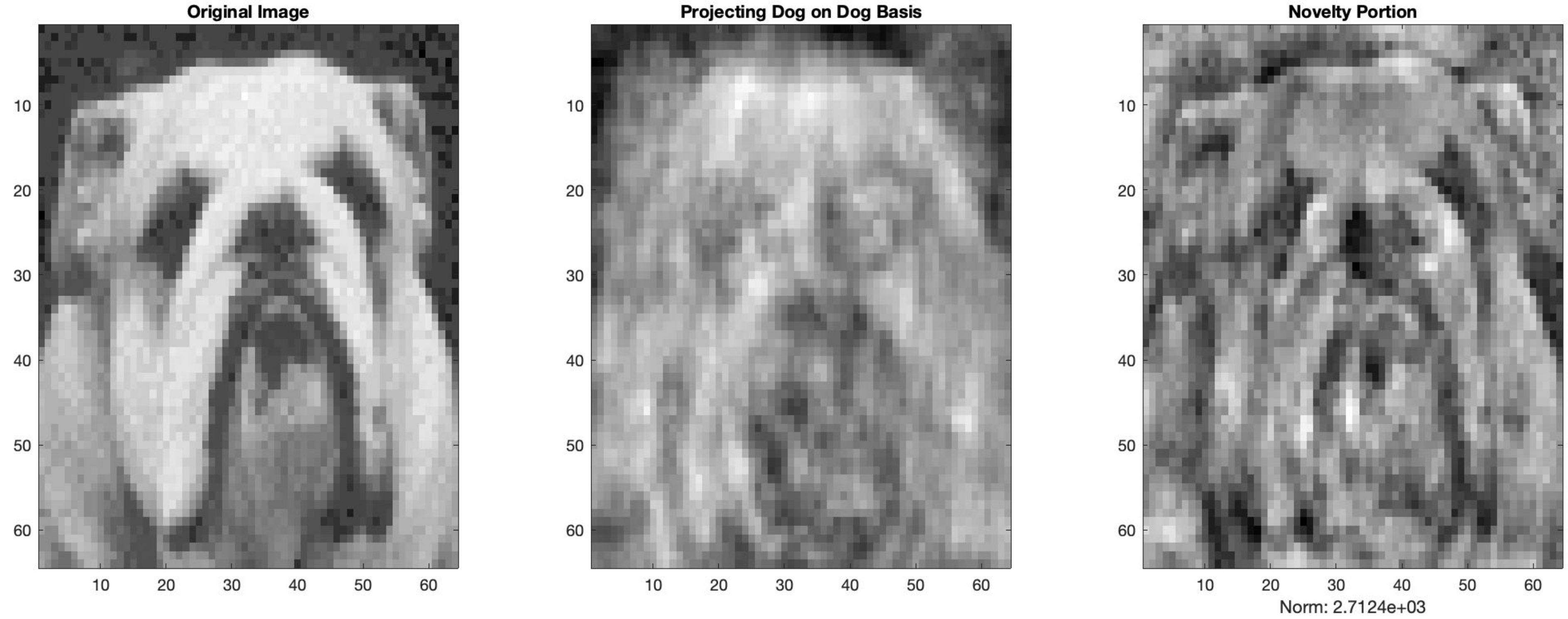


Kohonen Novelty Filter

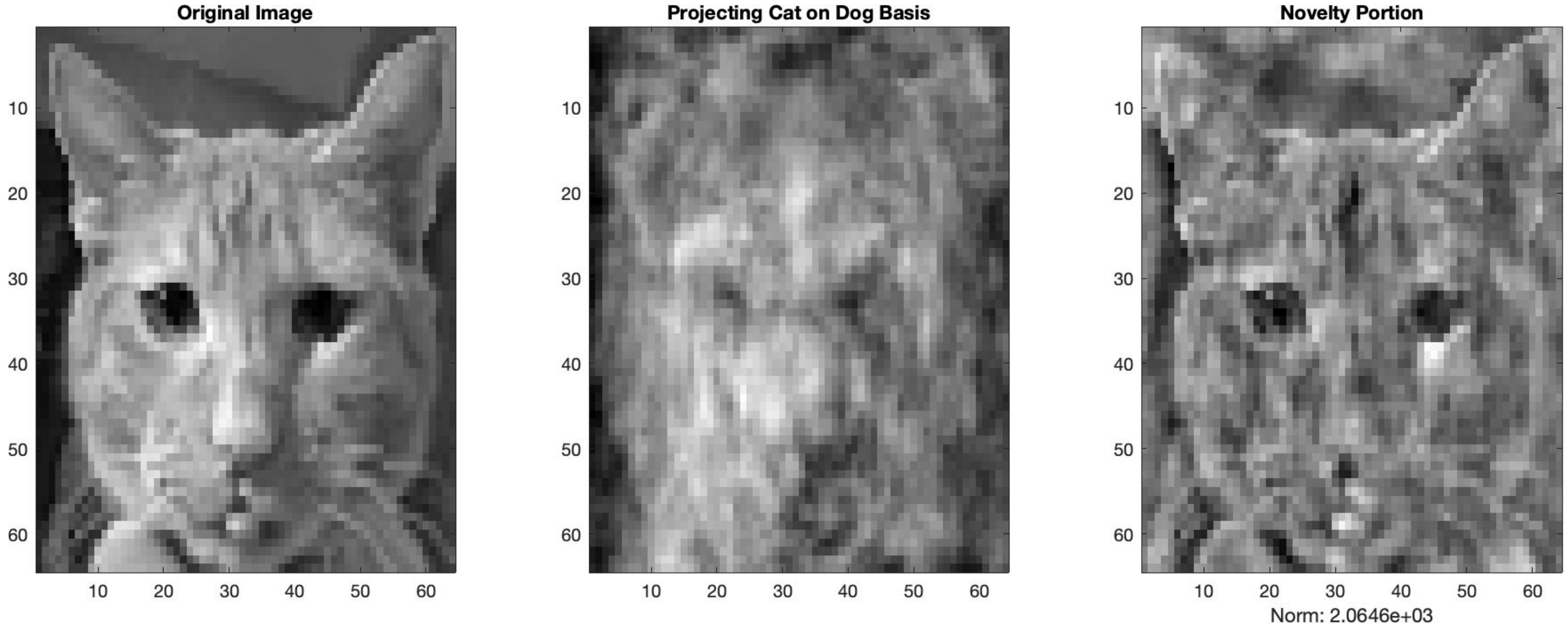
Cats Projected on Cats Basis



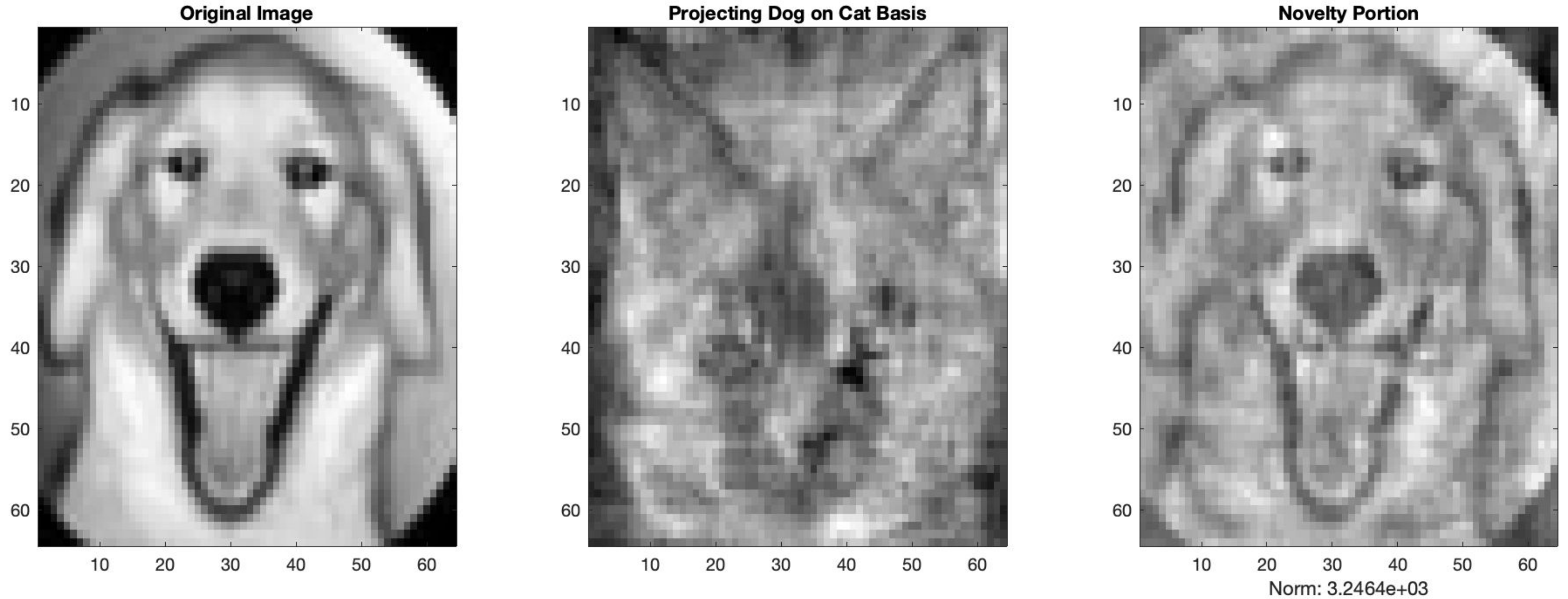
Dog Projected on Dog Basis



Projecting Cat on Dog Basis



Dogs Projected on Cat Basis



Confusion Matrix & Accuracy

```
C =

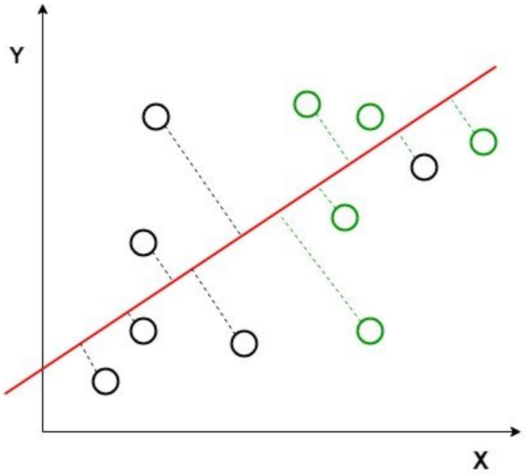
    15     4
     2    17

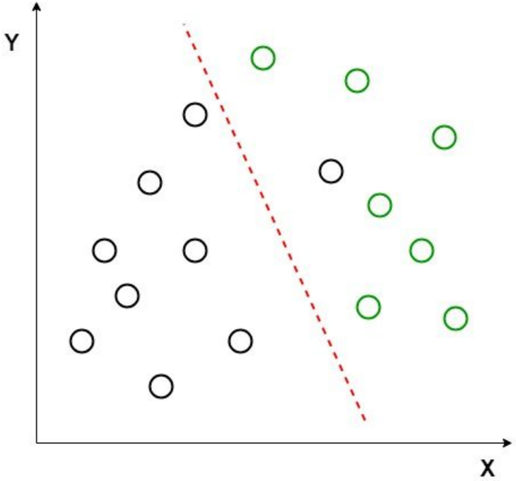
>> Accuracy=(15+17)/(15+17+2+4)

Accuracy =

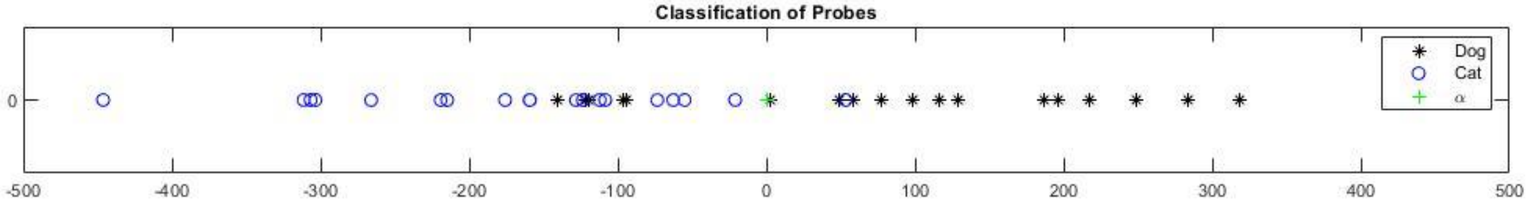
    0.8421
```

Linear Discriminant Analysis





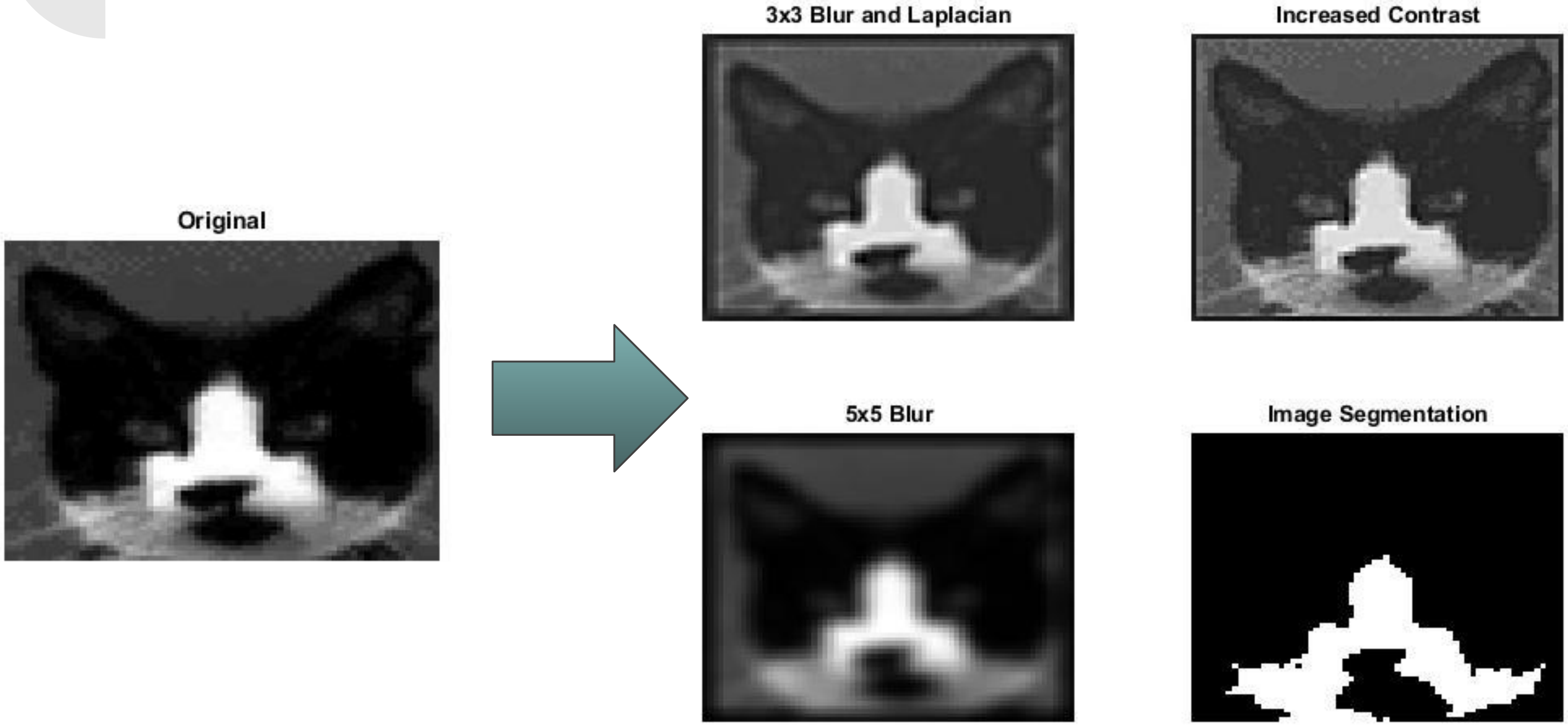
Initial Results



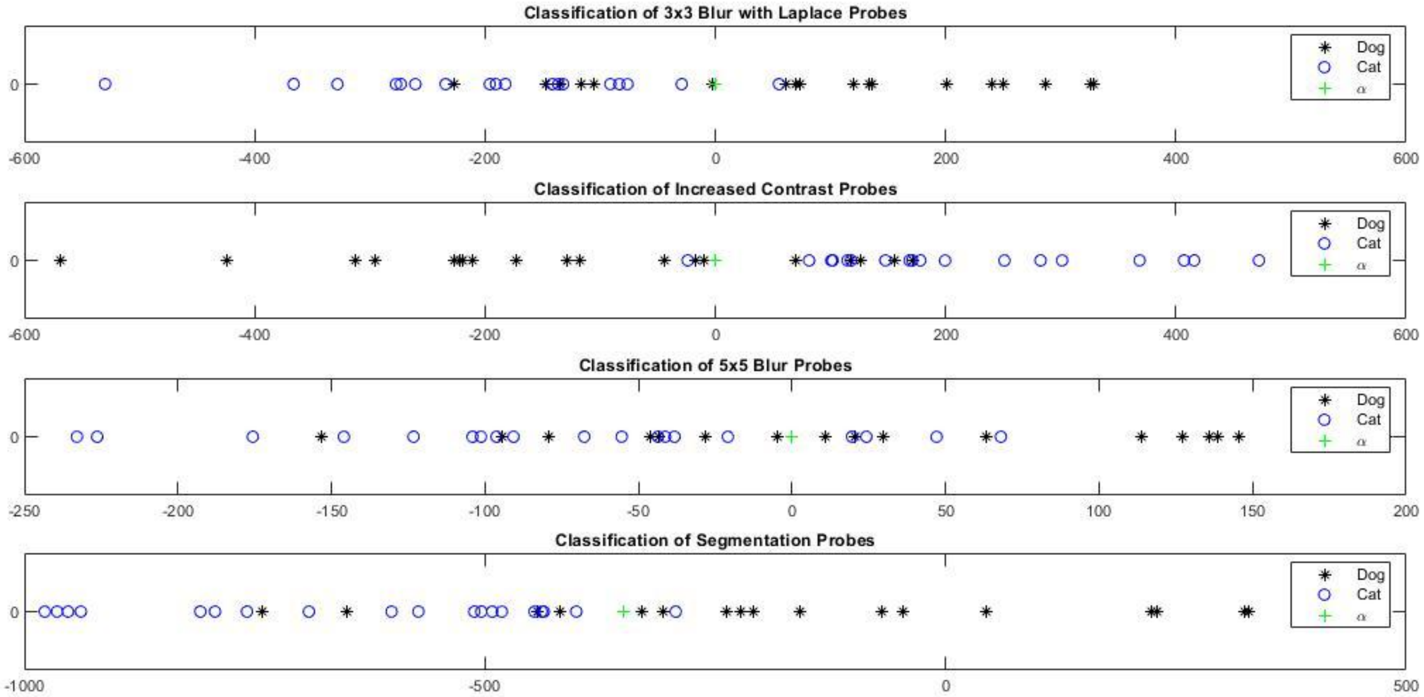
Confusion Matrix		Projected	
		Dog	Cat
Actual	Dog	14	5
	Cat	1	18

- Accuracy: 84.2%
- Better at classifying cats as cats

Filters


Original
3x3 Blur and Laplacian
Increased Contrast
5x5 Blur
Image Segmentation

Filtered Results



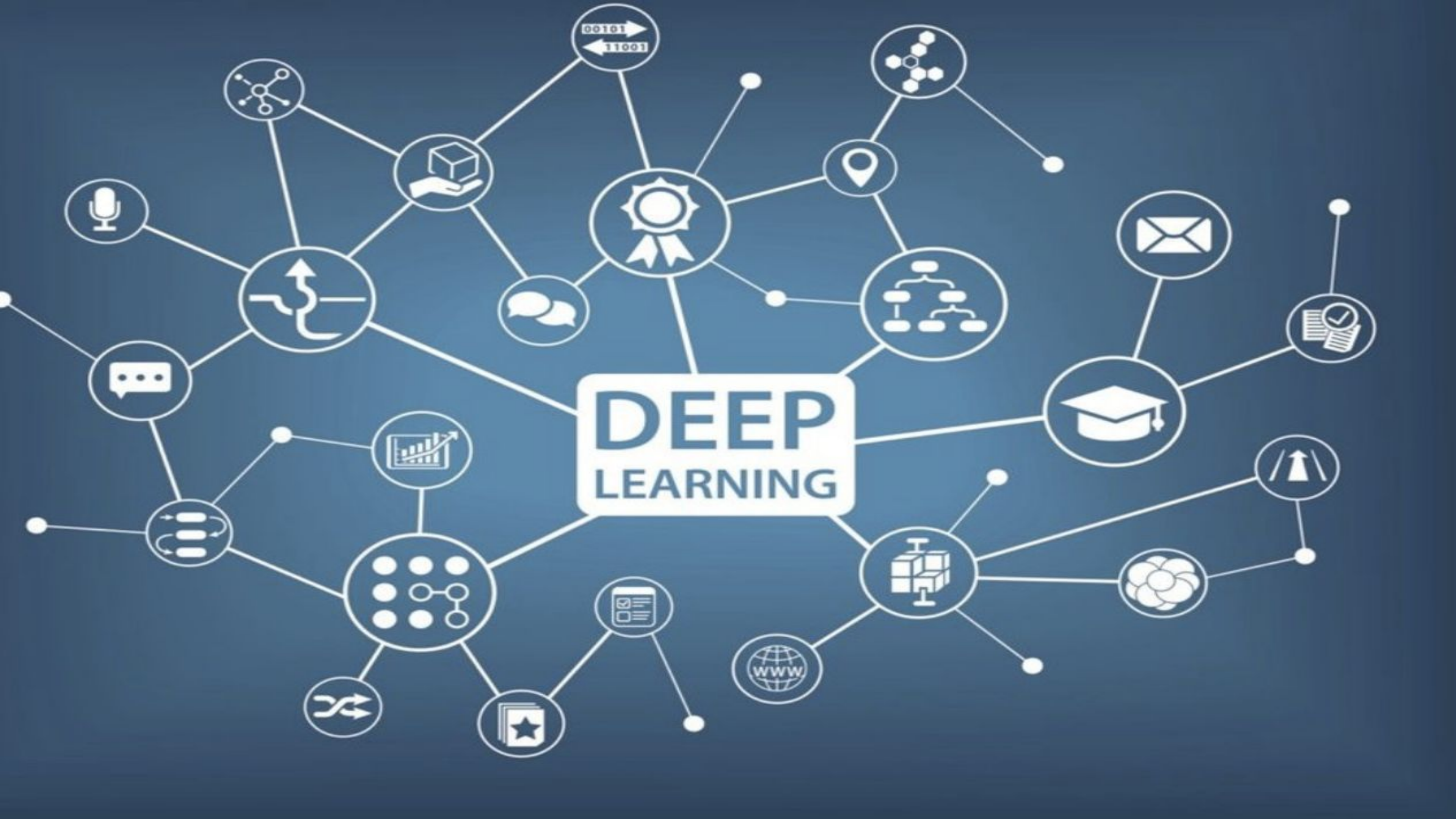
Filtered Results - Cont.

3x3 Blur + Laplacian Acc: 81.6%		Projected	
		Dog	Cat
Actual	Dog	13	6
	Cat	1	18

Increased Contrast Acc: 84.2%		Projected	
		Dog	Cat
Actual	Dog	14	5
	Cat	1	18

5x5 Blur Acc: 71.1%		Projected	
		Dog	Cat
Actual	Dog	12	7
	Cat	4	15

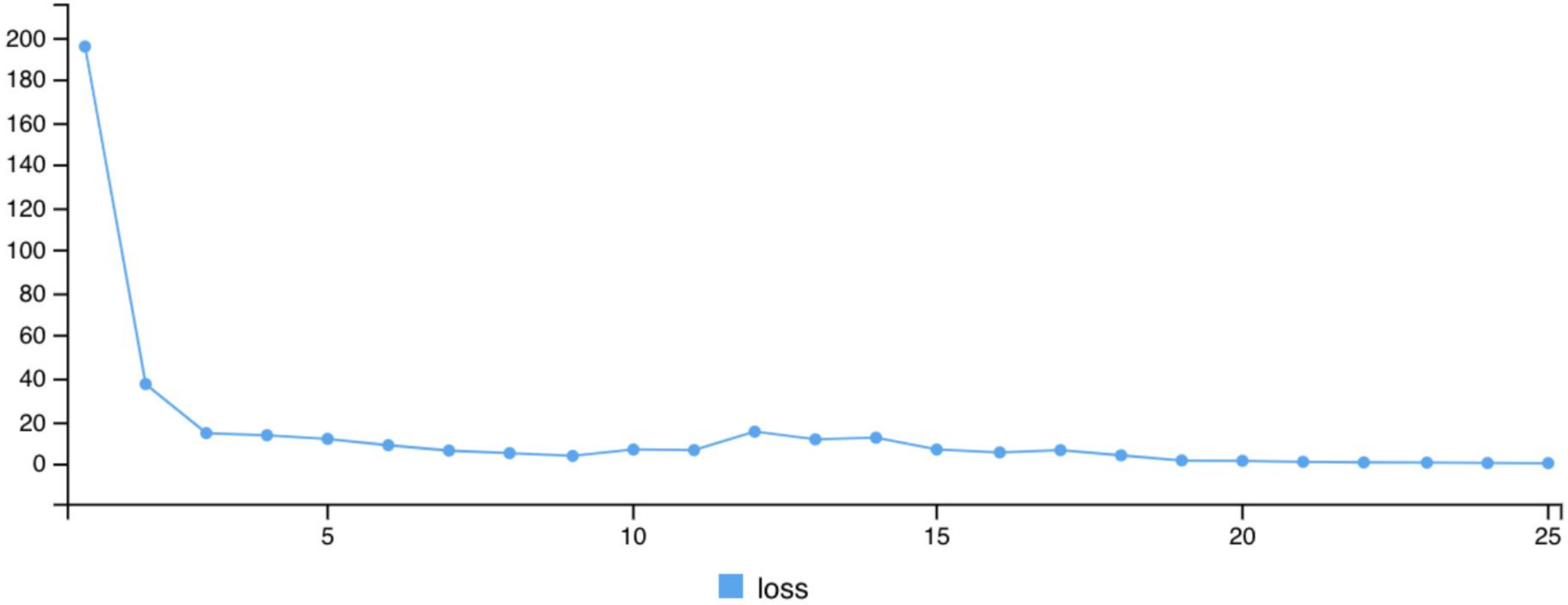
Segmentation Acc: 86.8%		Projected	
		Dog	Cat
Actual	Dog	15	4
	Cat	1	18


DEEP
LEARNING
00101
11001
www

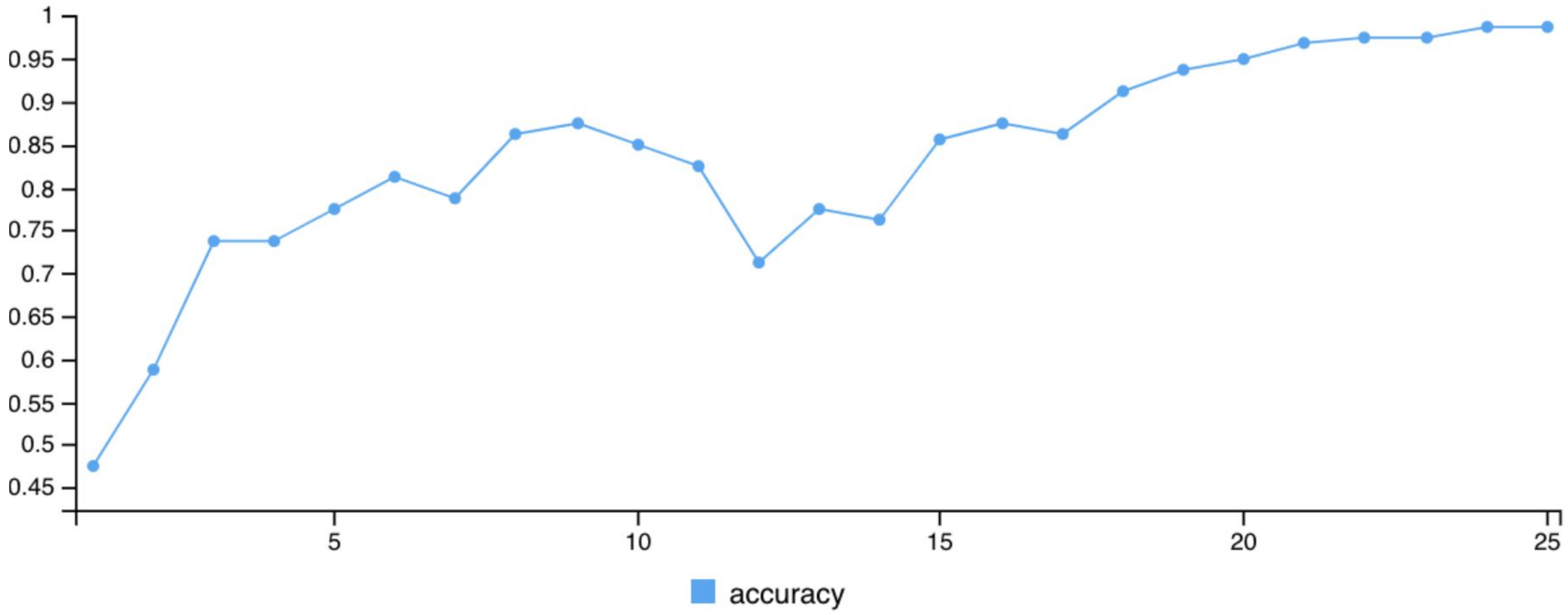
Data Preparation:

```
```{r }
train <-readMat("Desktop/MATH521/Final Project/tiffstudentdata_raw_vectorized.mat")$X
test  <-readMat("Desktop/MATH521/Final Project/PatternRecAns.mat")$TestSet

y_test  <- readMat("Desktop/MATH521/Final Project/PatternRecAns.mat")$hiddenlabels

n <- 80 # sample size
x = sample(1, replace=TRUE, size=n)
y = sample(0, replace=TRUE, size=n)
y_train = c(x,y)



training = data.matrix(train, rownames.force = NA)
test     = data.matrix(test, rownames.force = NA)


x_training = t(training)
x_test     = t(test)



```
```

R & Python: Keras Package

```
```{r }
model <- keras_model_sequential()
model %>%
  # Directly outputs positive values. Zeros negatives values.
  # 32 nodes.
  layer_dense(units = 32, activation = 'relu') %>%

  layer_dense(units = 1, activation = 'sigmoid') #
```
```

- Layer_Dense
  - Nodes = 32
  - Relu = Rectified Linear Activation Function
- Layer_Dense
  - Level = 1
  - Output: Sigmoid (Logistic/Binary)

Compile: Learning Rate, Probabilistic Losses, and Accuracy

- Stochastic Optimizer (Learning Rate):
  - Adam Algorithm: Stochastic Gradient Descent Method that estimates 1st and 2nd moments.
- Loss:
  - $\text{Cross Entropy} = E_p[q], \qquad q = 1 - p, \qquad p \sim \beta(\alpha, \beta)$
  - Loss between predicted and actual
- Metrics (Accuracy)
  - Outputs a variable that counts true and false frequency

```
```{r }
model %>% compile(
  optimizer = 'adam', loss = 'binary_crossentropy', metrics = c('accuracy')
)
```
```

Cross Entropy: Error between True Labels & Predicted Labels

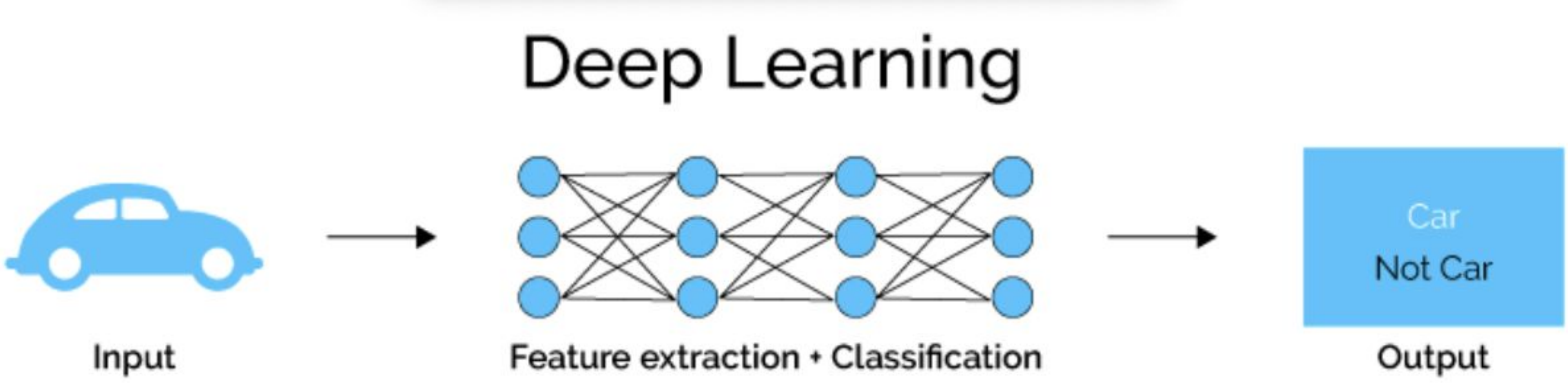


Accuracy



Fit the Model

Epochs: Defined by an arbitrary cutoff. An interaction through the entire dataset.
Verbose: “How do you want to see the process?”




```
model %>% fit(x_training ,y_train , epochs = 25, verbose = 2)
```

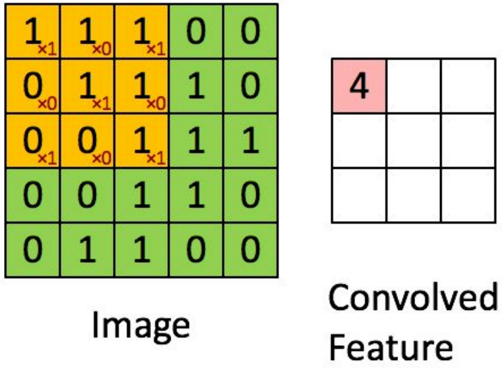
Prediction:

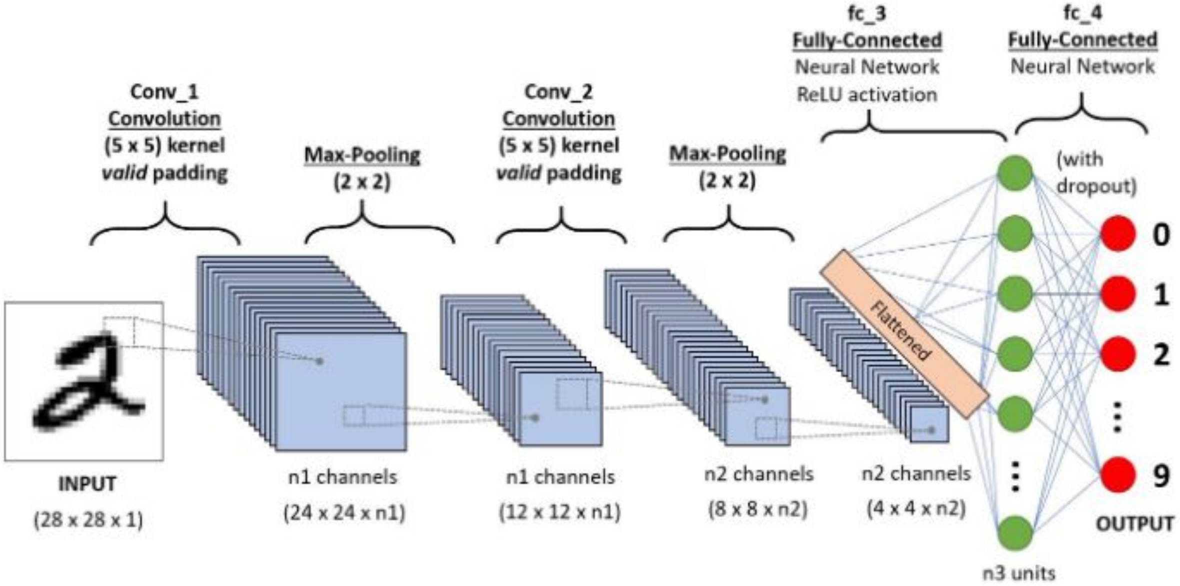
```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 10  9
         1  0 19

               Accuracy : 0.7632
                 95% CI : (0.5976, 0.8856)
```

Convolutional Neural Network

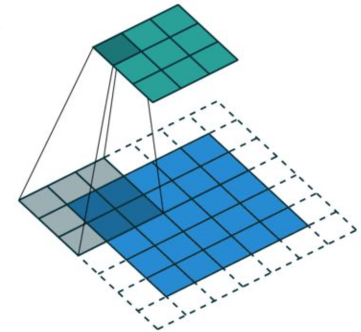




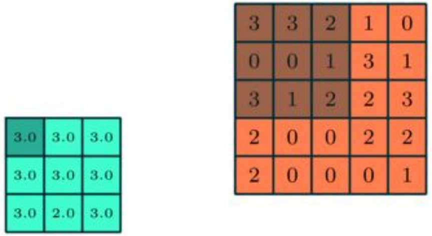
A CNN sequence to classify handwritten digits

Convolutional Neural Network

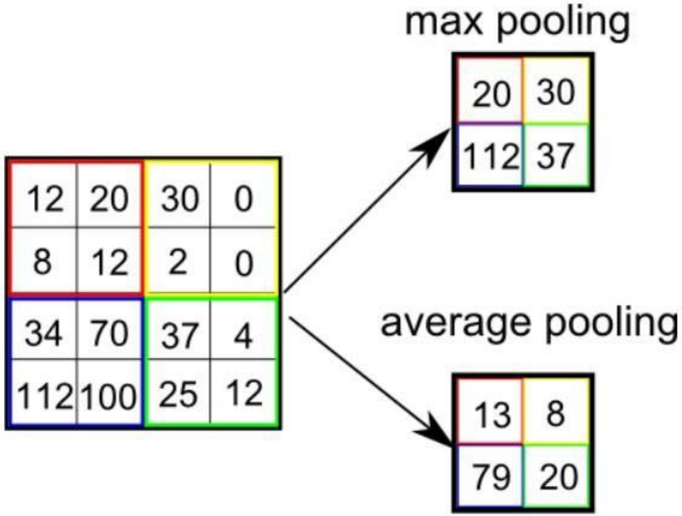
Padding (Same)



Pooling Layer

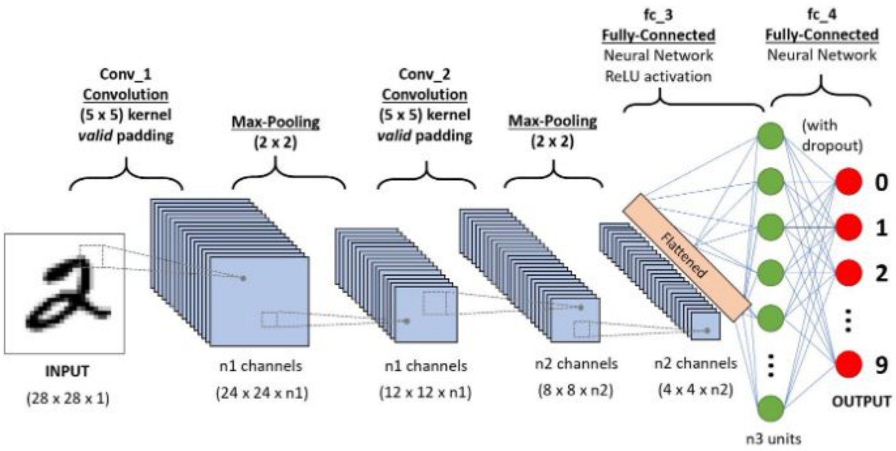


Max vs Average Pooling



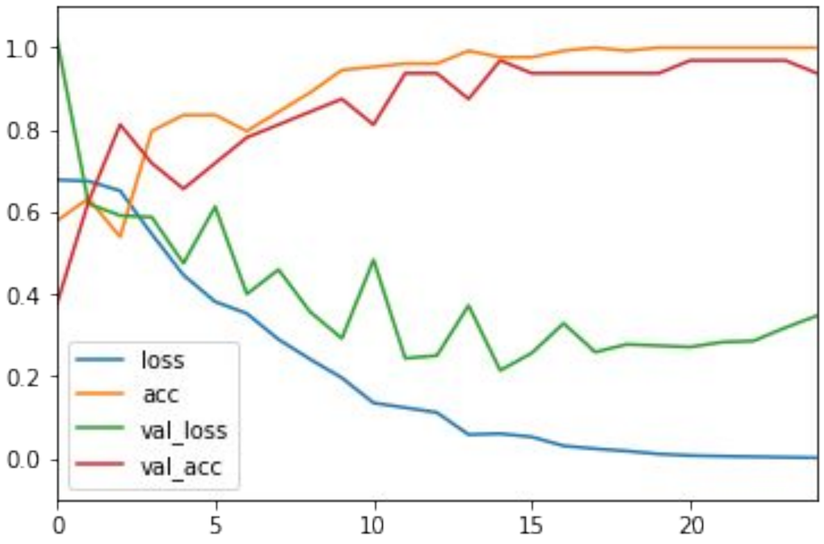
Convolutional Neural Network

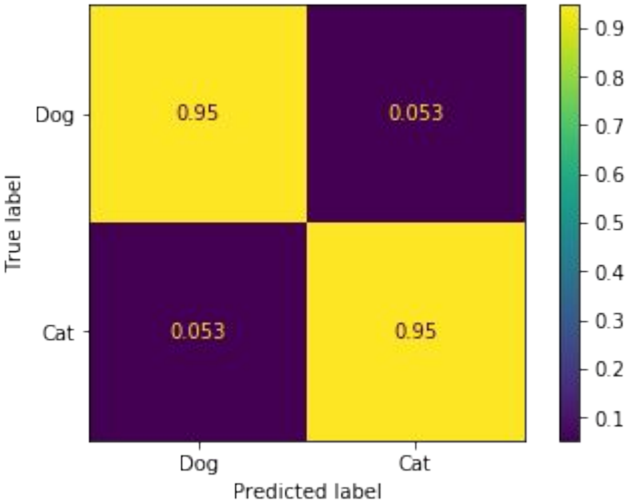
```
model = keras.Sequential()
model.add(keras.layers.Conv2D(32, (3,3),input_shape = (64,64,1), activation='relu', padding='same'))
model.add(keras.layers.MaxPooling2D(2,2))
model.add(keras.layers.Conv2D(64, 3, activation='relu', padding='same'))
model.add(keras.layers.MaxPooling2D(2))
model.add(keras.layers.Conv2D(128, 3, activation='relu', padding='same'))
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(32, activation='relu'))
model.add(keras.layers.Dense(1, activation='sigmoid'))
```



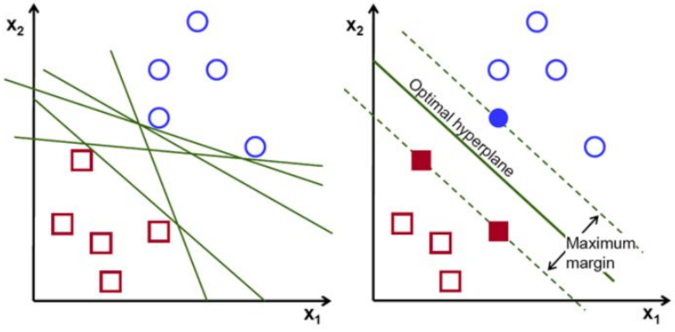
A CNN sequence to classify handwritten digits

Convolutional Neural Network





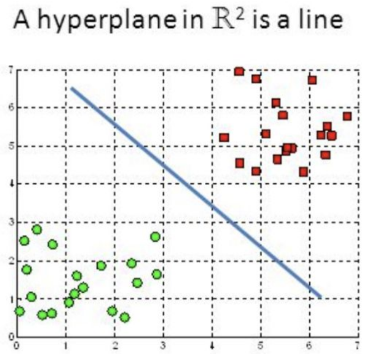
Support Vector Machine

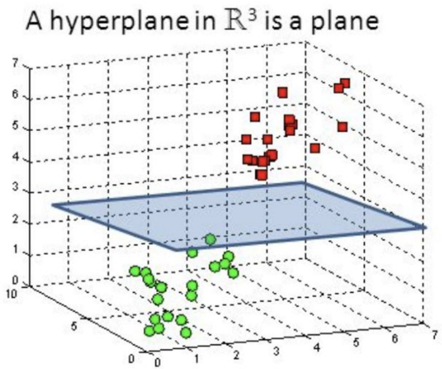


Possible hyperplanes

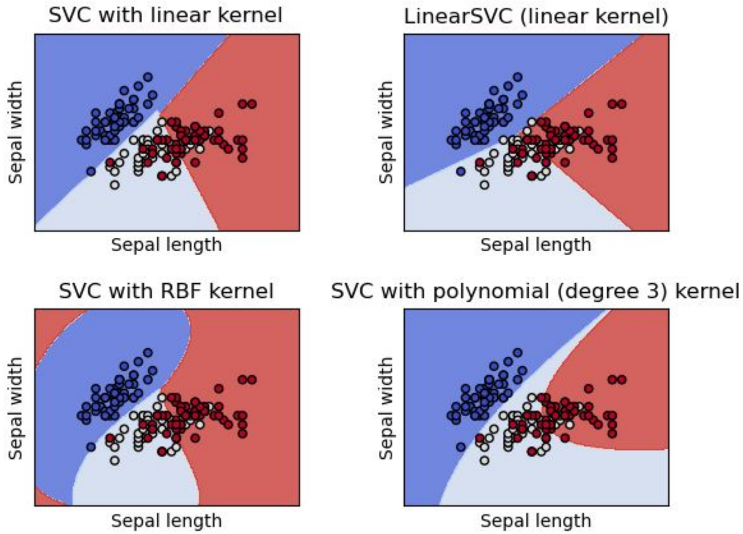
The objective of the support vector machine algorithm is to find a **hyperplane** in an N-dimensional space(N — the number of features) **that distinctly classifies the data points.**

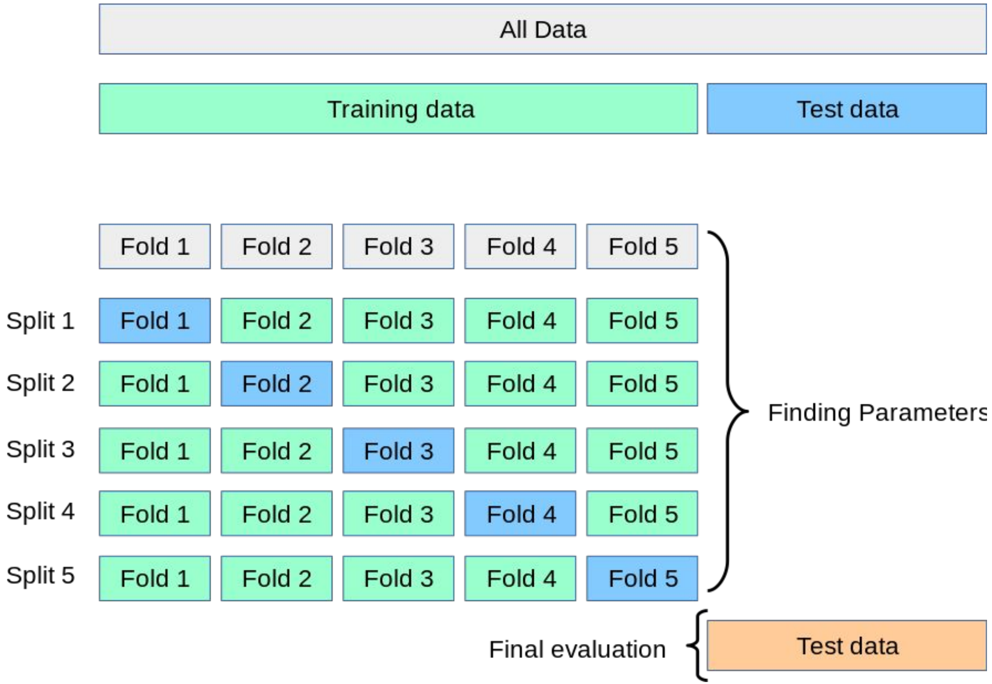
Our objective is to **find a plane that has the maximum margin**, i.e the maximum distance between data points of both classes.



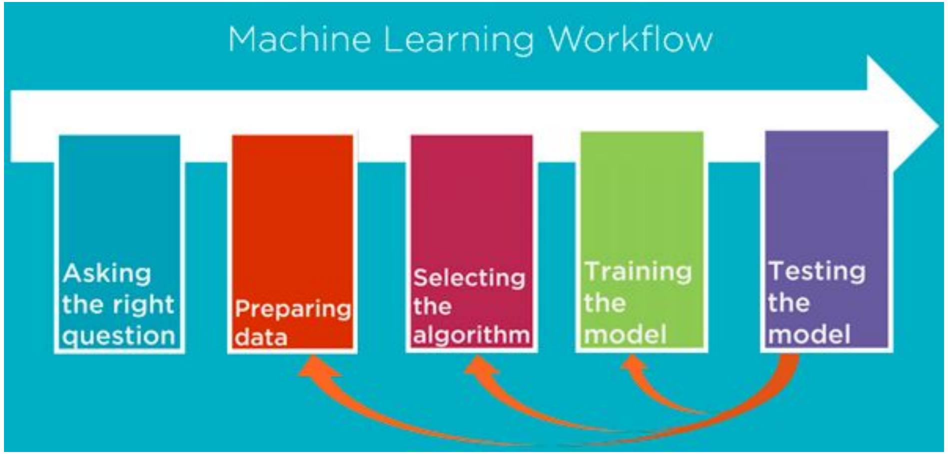


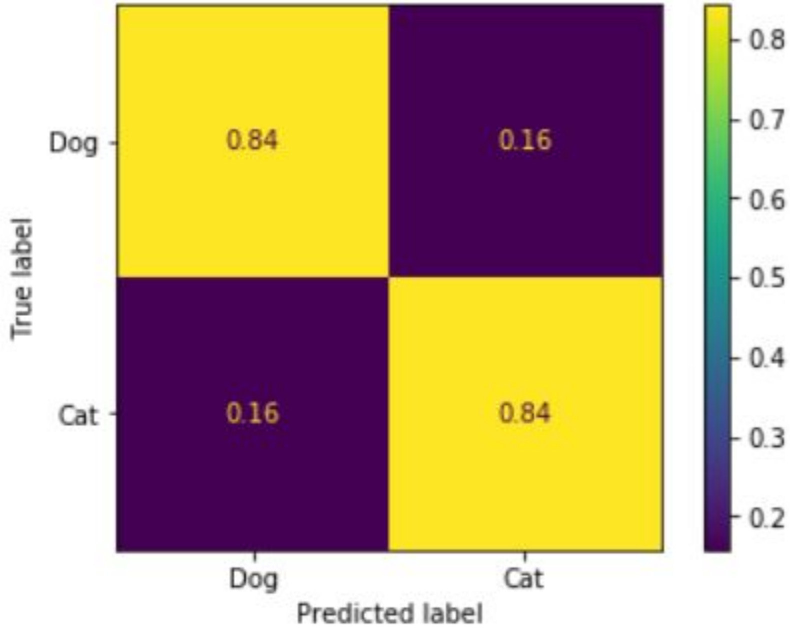
Support Vector Machine



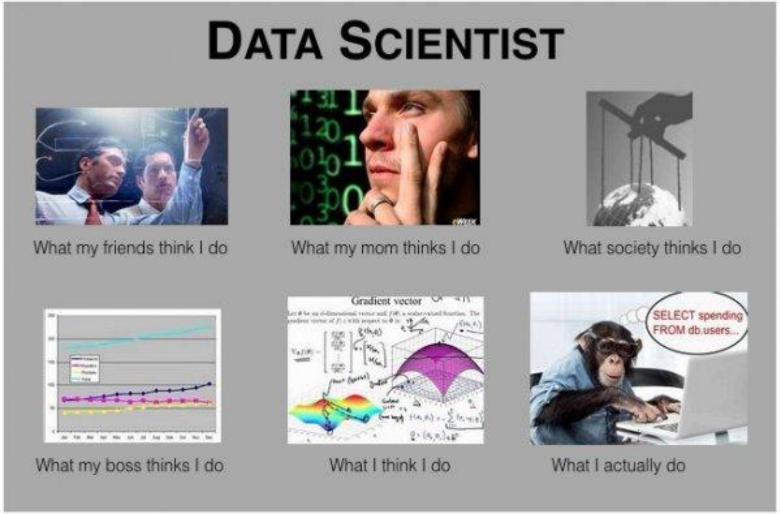


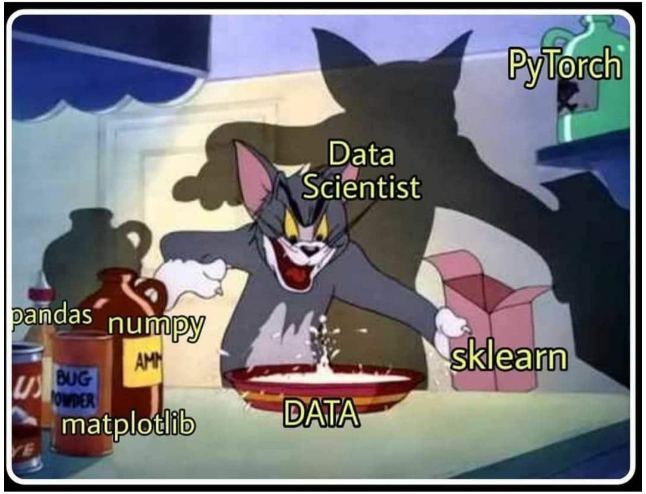
Support Vector Machine





Q&A and Feedback





Thanks!