**Summary:**
Machine learning (ML) is a sub-section of artificial intelligence (AI) that implements statistical algorithms that learn from data without being explicitly programmed. Machine learning's predictive analytic solutions apply in many fields including natural language processing, computer vision, speech recognition, agriculture and medicine. This report is an effort to promote a hands-on understanding of many different algorithm in the machine learning domain. Custom designed data sets have been obtained from www.roboflow.ai, www.udemy.com (machine learning A to Z), www.kaggle.com to simulate data collection. Many different algorithm have been implemented in python's sklearn library and keras library. These algorithms are deployed to C/C++ using computer vision libraries (OpenCV) and Open Neural Network Exchange (ONNX) run time library. These ML algorithms are Linear Regression, Polynomial Regression, Support Vector Regression (SVM), Decision Tree Regression, Random Forest Regression, K-Nearest-Neighbor (KNN), Kernel SVM, Naive-Bays, Association Rule Learning, Reinforcement Learning, Natural Language Processing. The computer vision algorithms are Haar Cascade, Histogram of Oriented Gradient (HOG), Convolution Neutral Network (CNN), AlexNet, Visual Geometry Group (VGG-16), You Only Look Once (YOLOv5). Examination of computer vision's algorithms against an identical data set showed the deep learning's predictive solutions are 90 percent accurate and traditional predictive models are 75 percent accurate. Overall, the python and CplusPlus implementation of the algorithms described here provided a controlled environment for the experimental and detailed understanding of when a particular mathematical models to be used.

**Introduction**
Machine learning computer algorithms are trained with real world data to build predictive models. As an example, data is collected on the height and weight of 100 people. This is called the training data. The data is graphed with the measured heights on the X-axis and weights on the Y-axis as seen in Figure 1. A simple machine learning algorithm could fit a line through this data. This line is utilized to make prediction about the weight of new people given their heights.

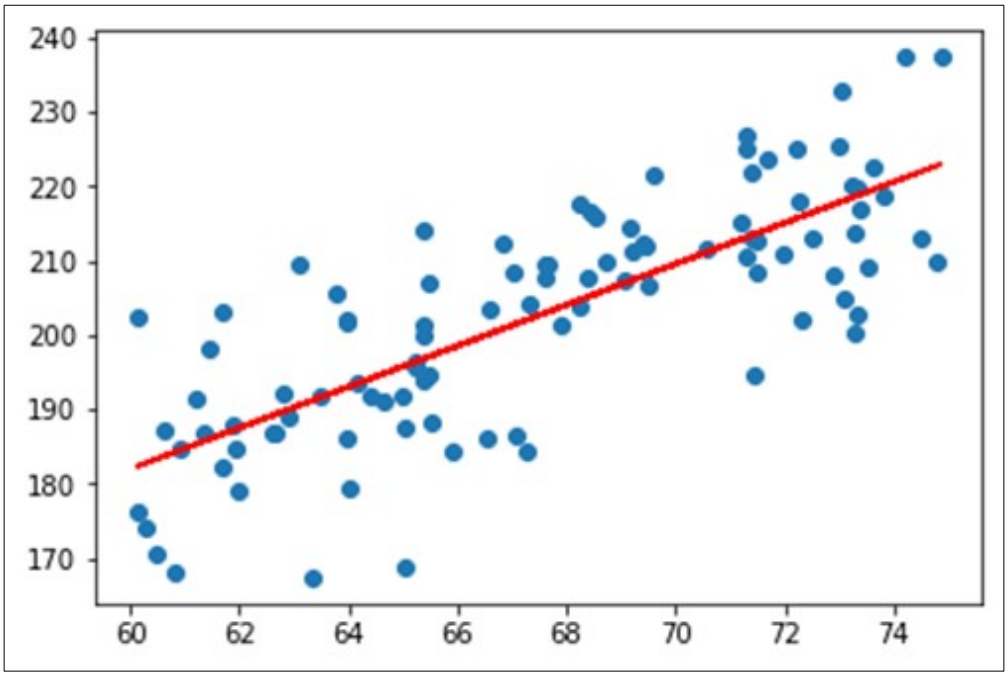

Figure 1: Data Collected from 100 people (machine learning A-Z)[1]

**Linear Regression**

A python's computer program for a simple linear regression is presented in the Table 1. This model is used to predict the value of a variable based on the value of another variable. The predicted values are dependent variable. The variable used to predict the other variable's value is called the independent variable. A template python code is provided in Table 1.

The first section of the code is importing the libraries. The second section of the code is to import the dataset to the python code. The third section of the code uses sklearn functions to split the input data to training and testing partitions. The forth section of the code will train the simple regression model to fit the provided data. The fifth section of the code predicts future data and plots them. This is template for all the python code utilized for all the ML algorithms.

*Table 1: Simple Linear Regression [1]*

```
# Simple Linear Regression

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
```

```
dataset = pd.read_csv('Salary_Data.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3,
random_state = 0)

# Training the Simple Linear Regression model on the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Predicting the Test set results
y_pred = regressor.predict(X_test)

# Visualising the Training set results
plt.scatter( X_train, y_train, color = 'red')
plt.plot( X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Training set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()

# Visualising the Test set results
plt.scatter( X_test, y_test, color = 'red')
plt.plot( X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Test set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

**Polynomial Regression**

This type of regression is useful when there is a non-linear relationship between the value of independent variable and the corresponding conditional mean of the predicted value.  A working template for this type of regression is provided in the regression directory [2]

**Support Vector Regression (SVM)**
SVMs can efficiently perform linear classification and non-linear classification using some kernel adjustments. The data is represented through a set of pairwise similarity comparison using a kernel function. The kernel function transforms the pairwise data into coordinates in a higher-dimensional space. Consequently, SVM uses a kernel trick to implicitly map it's inputs into high-dimensional feature spaces where linear classification is performed. This property makes SVM resilient to noisy data. A working template for this type of regression is provided in the regression directory [2]. Kernel SVM is a class mathematical algorithms for pattern analysis. They help organize data points and

**Decision Tree Regression**
In this algorithm a tree is built by splitting the source dataset into a root node and a pair of subset nodes. The splitting is based on a set of rules dictated by classification features. This process is repeated on each derived subset in recursive manner called recursive partitioning. This process of top-down induction of decision trees is an example of greedy algorithm. A working template for this type of regression is provided in the regression directory [2]

**Random Forest Regression**
This machine learning algorithm combines the predictions of multiple decision trees to reduce over-fitting and improve accuracy. A working template for this type of regression is provided in the regression directory [2]

**K-Nearest-Neighbor (KNN)**
The kNN algorithm is a technique that uses a set of data point's neighbors to predict its class or value. A working template for this type of regression is provided in the regression directory [2]

**Naive-Bays**
This algorithm uses Baye's theorem to calculate the conditional probability distribution of a label given an observation. It then assignes the observation to the class with the highest probability. A working template for this type of regression is provided in the regression directory [2]

**Association Rule Learning**
This algorithm discovers relationships between variable in large dataset. It's used to find patterns and association in data, such as which items customers buy together. Association rule learning uses "if-then" statements to indicate the

probability of relationship between data items. A working template for this type of regression is provided in the regression directory [2].

**Reinforcement Learning**
This algorithm uses an "agent" to learns to make decisions in an environment by receiving feedback in the form of rewards or punishments for its actions. It is essentially learning through trial and error to maximize it's long term reward by choosing the best actions in each situation. A working template for this type of regression is provided in the regression directory [2].

**Natural Language Processing**
This algorithm abbreviated by NLP helps computers understand human language. It is used to perform tasks like machine translation, spell check and sentiment analysis.

**Object Detection and Image classification**
Advances in the field of deep learning has allowed neutral network to surpass many previous approaches in performance. Computer vision historically used algorithms such as haar cascade classifier and Histogram of Oriented Gradients (HOG) to perform object detection. These were traditional machine learning object detection program that identified objects in an image and video. A detailed understanding of Harr classification can be observed in Paul Viola and Michel Jone's paper "Rapid Object Detection using a Boosted Cascade of Simple Features" [3]. This publication is mathematical and understanding of the mathematical models is out scope for this report. However the algorithm can be explained to follow a few steps.

- Calculating Haar Features.
- Creating Integral Images.
- Using Adaboost
- Implementing Cascading Classifiers.

It is important to remember that Haar Cascading algorithm requires a large number of positive images and negative images of non-target to train the model. Code is developed in Python and C++ using OpenCV libraries. The process of training a cascade classifier model and utilizing that model to perform object detection is described in youtube article. The youtube article is deciphered in an example and presented in 'opencv_training' directory [2].

Deep learning family of machine learning algorithms have surpassed the traditional computer vision algorithms. The Artificial Neural Network, Convolution Neural Network (AlexNet), Visual Geometry Group (VGG16), ResNet50, YOLOV3, YOLOV5 are image classification and object detection algorithms that utilize neural network or deep neural network. In this effort multiple data sets such as cats and dogs, hand written digits, masked and unmasked faces, automobiles, elephants have been downloaded to represent custom data-sets. Custom data-sets are labeled and annotated manually to train the models in python using 'tensorflow.keras' libraries and Open Neural Network Exchange run time libraries. Once the keras formatted model is trained and saved to the hard drive, the ONNX is used to convert the model to ONNX format. A second python code and ANSI C/C++ code loads the ONNX formatted model and uses openCV libraries to perform image classification and object detection. A few of the results are show in the figures below. With custom data set of 90 images and an epoch of 100 iteration the object detection has an accuracy of 64 percent to 85 percent.

*Figure 2: Object Detection performed by YOLOv5 model.*

YOLOv5 Model has an accuracy of 89 percent and 91 percent with a custom data set of 32 images and an epoch of 100 iteration the object detection has.

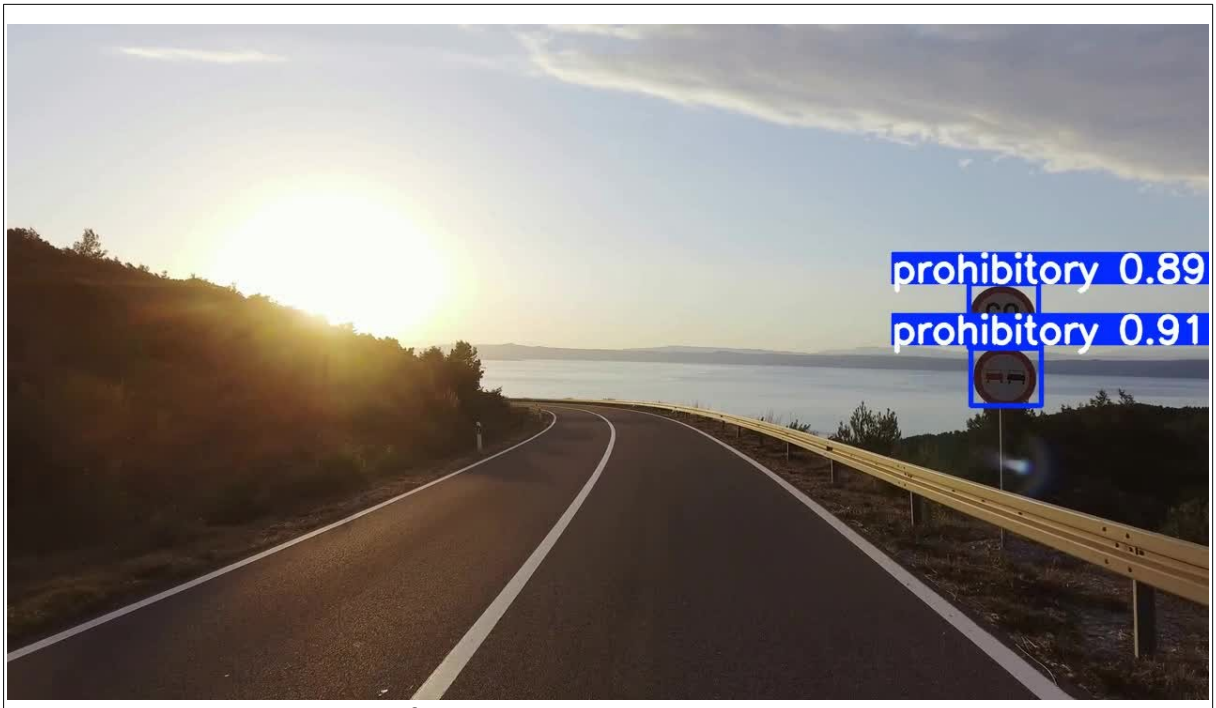


*Figure 3: Object Detection performed by YOLOv5 model.*

The YOLOV5 model has 82 percent to 90 percent accuracy with a custom data set of 40 images and an epoch of 100 iteration.

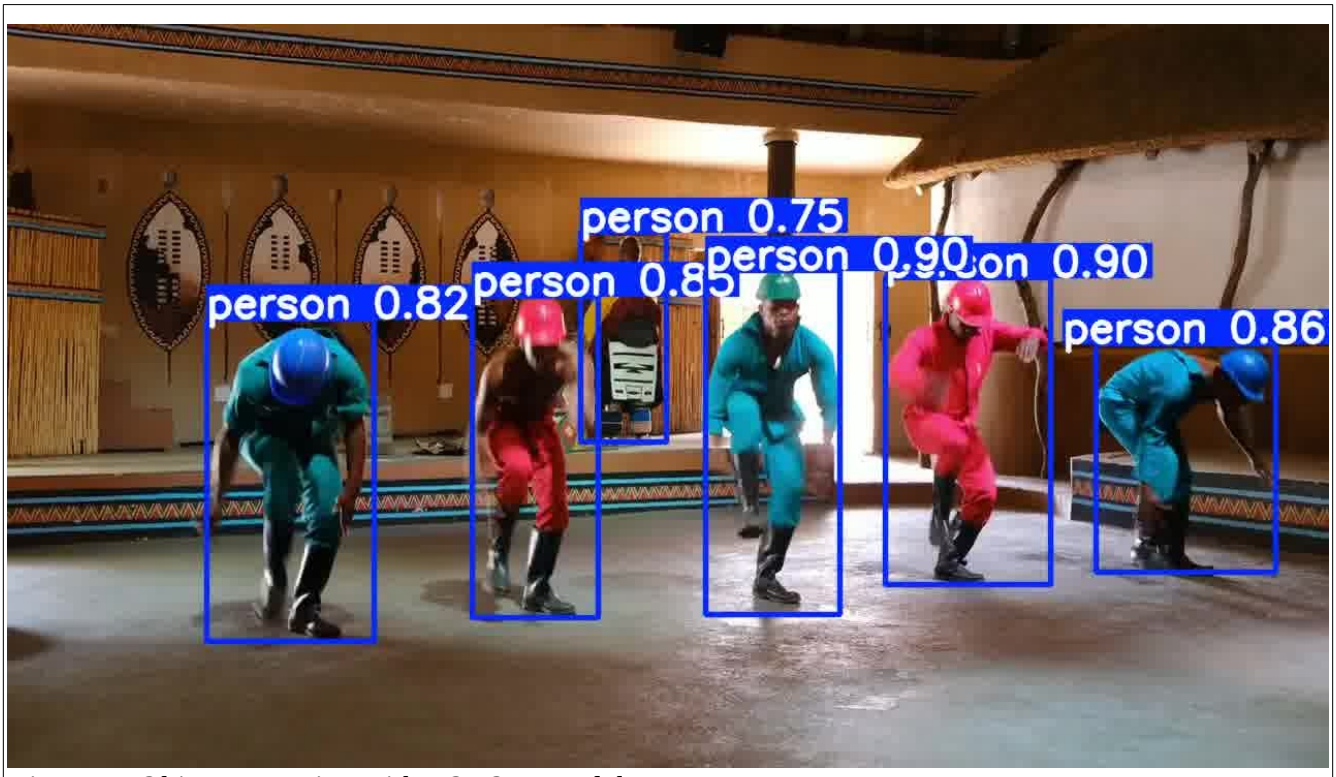


*Figure 4: Object Detection with YOLOv5 model*

The YOLOV5 model has 62 percent to 90 percent accuracy with a custom data set of 12 images and an epoch of 100 iteration.

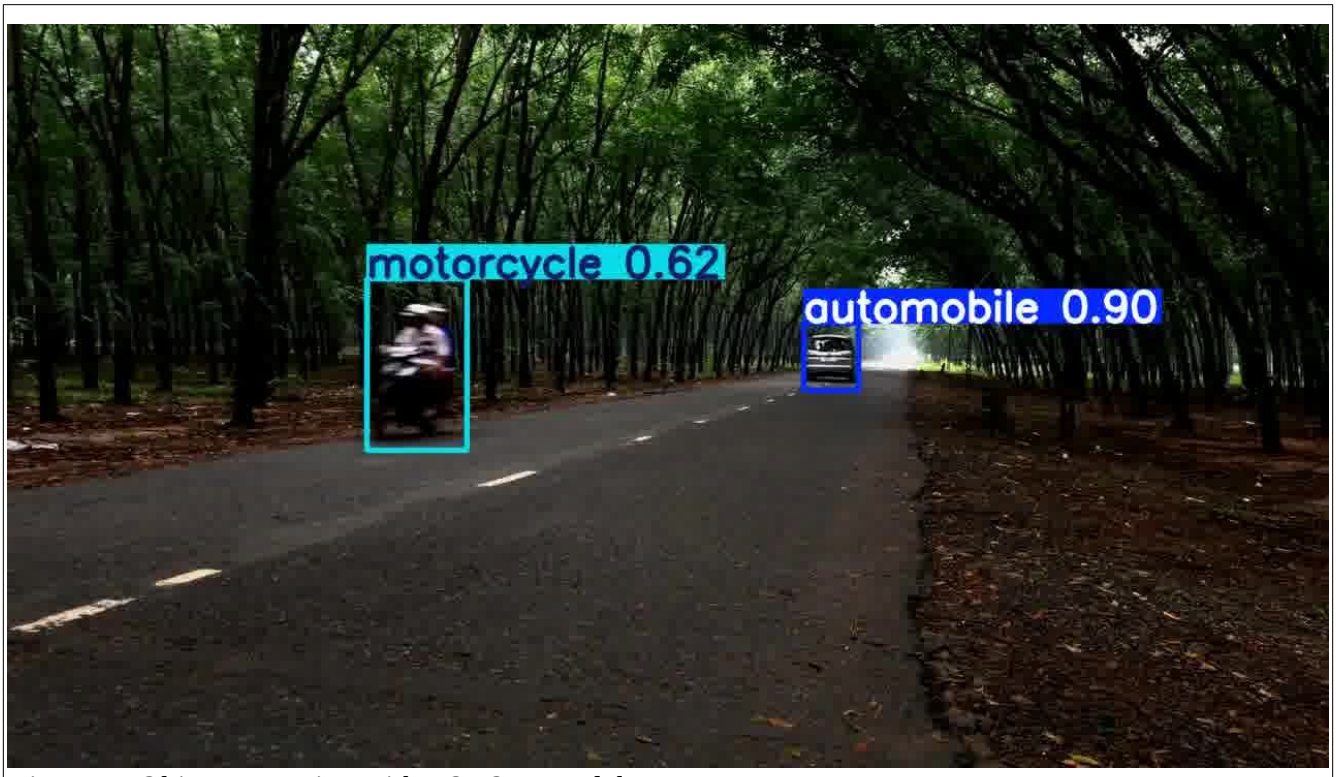


*Figure 5: Object Detection with YOLOv5 model.*

The YOLOV5 model has very low accuracy which result in wrong object detection.

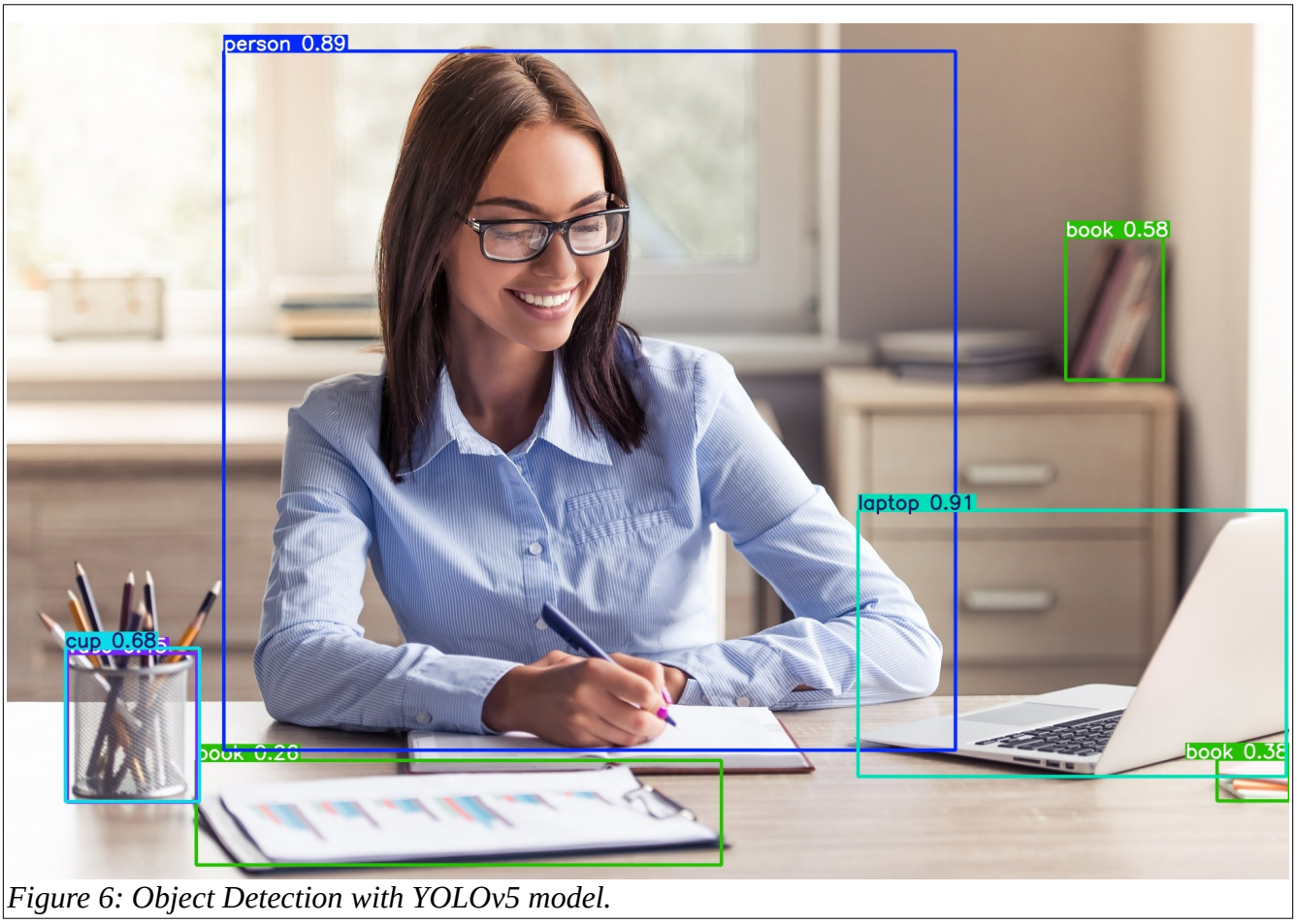


*Figure 6: Object Detection with YOLOv5 model.*

The image classification model using CNN (AlexNet) had a 94 accuracy for and written digits from zero to nine. The image classification model using VGG-16 had a 97 percent accuracy due the 16 layer model. The directories starting with keyword 'ONNX' have the python code and C/C++ code.

**Next**

This effort has compared implementation and utilization of YOLOv3 vs YOLOv5 deep neural network models. The mathematical mode of YOLOv5 has additional layers vs YOLOv3, so they are more accurate by 5 percent. There are additional models like Mask RCNN which has not been explored in this effort due to time constraint. AlexNet with seven layers have been compared against VGG with sixteen layers. The result is a five percent increase in accuracy of 97

percent.  Additional accuracy must be obtained by increasing the number of samples in the data set and increasing the pixel resolution of samples.

## Conclusion

The deep network family of ML algorithms are 20 percent more accurate than the traditional cascading classifiers.  The combination of CUDA and Multi Layer Architecture of the neural network models have reduced the training time and have increased the accuracy of the software solutions in the recent years.  This trend is very pragmatic, since YOLOv10 has already been released with the promise of higher accuracy.

## Reference

[1] www.udemy.com (machine learning A to Z)

[2] Code section of this document.

[3] https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf