Unleashing the Power of the Primitive Brain: Understanding Our Instincts and Impulses

Roshan Upadhyay RA2011047010015, Arnab Banarjee RA2011047010002, Abhinn Daga RA2011047010007, Glenn Paul Aby RA2011047010017, Yash Singh RA2011047010019

Abstract

Multitasking ability is a highly desirable trait in both humans and machines. In this project, we propose a novel approach to achieve multitasking ability in machines by combining the outputs of different models trained for different tasks. Specifically, we use a neural network architecture that takes as input a task description and selects the appropriate model to use for that task. The selected model produces an output, which is then combined with the outputs of other models to produce a final output. We demonstrate the effectiveness of our approach on a set of benchmark tasks and show that it outperforms single-task models and other multitask learning approaches. In this approach we used 4 different models which are pokemon clasifier, lukemia clasifier, car plate detection and hand written digit recognition. The above model gave a very good accuracy i.e. 90%, 84%, 95%, 99% respectively.

Introduction

Multitasking is a highly desirable trait in both humans and machines. In humans, the ability to perform multiple tasks simultaneously is essential for success in many domains. Similarly, in machines, the ability to multitask can improve efficiency and performance in various applications. However, achieving multitasking ability in machines is challenging due to the complexity and diversity of tasks.

In recent years, there has been a growing interest in multitask learning, where a single model is trained to perform multiple tasks simultaneously. While multitask learning has shown promising results in some domains, it has limitations, such as the need for large amounts of data, the difficulty of choosing appropriate task combinations, and the potential for negative transfer between tasks.

To address these limitations, we propose a novel approach that combines the outputs of different models trained for different tasks to achieve multitasking ability in machines.

Project Goal

The goal of this project is to use different models trained for different tasks to work together like multitasking ability of the brain to give correct output for a given task

Dataset

Pokemon Classifier

- The dataset contains 151 folders, one for each gen 1 pokemon and each folder contains 60 image for every pokemon.
- In all there are over 10,000+ images
- https://www.kaggle.com/datasets/thedagger/pokemon-generation-one

Car License Plate Detection

- This dataset contains 433 images with bounding box annotations of the car license plates within the image.
- Annotations are provided in the PASCAL VOC format.
- https://www.kaggle.com/datasets/andrewmvd/car-plate-detection?select=images

Lukemia Detection

- Acute lymphoblastic leukemia (ALL) is the most common type of childhood cancer and accounts for approximately 25% of the pediatric cancers.
- These cells have been segmented from microscopic images and are representative of images in the real-world because they contain some staining noise and illumination errors, although these errors have largely been fixed in the course of acquisition.
- The task of identifying immature leukemic blasts from normal cells under the microscope is challenging due to morphological similarity and thus the ground truth labels were annotated by an expert oncologist.
- In total there are 15,135 images from 118 patients with two labelled classes:

  Normal cell;
  Leukemia blast.

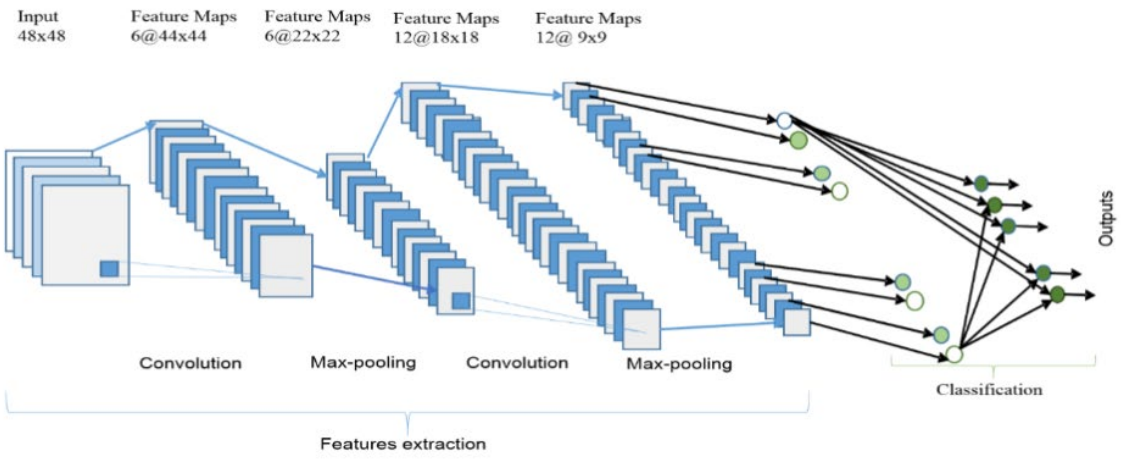
- https://www.kaggle.com/datasets/andrewmvd/leukemia-classification

Handwritten Digit Recognition

- The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems.
- The black and white images from NIST were normalized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels.
- The MNIST database contains 60,000 training images and 10,000 testing images. Half of the training set and half of the test set were taken from NIST's training dataset, while the other half of the training set and the other half of the test set were taken from NIST's testing dataset.

Model Architecture:

Pokemon Classifier



The model architecture of the model is as follows:

```
model = Sequential()
model.add(Conv2D(32, 3, padding = 'same', activation = 'relu', input_shape =(96, 96, 3), kernel_initializer = 'he_normal'))
model.add(BatchNormalization(axis = -1))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, 3, padding = 'same', kernel_initializer = 'he_normal', activation = 'relu'))
model.add(BatchNormalization(axis = -1))
model.add(Conv2D(64, 3, padding = 'same', kernel_initializer = 'he_normal', activation = 'relu'))
model.add(BatchNormalization(axis = -1))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, 3, padding = 'same', kernel_initializer = 'he_normal', activation = 'relu'))
model.add(BatchNormalization(axis = -1))
model.add(Conv2D(128, 3, padding = 'same', kernel_initializer = 'he_normal', activation = 'relu'))
model.add(BatchNormalization(axis = -1))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(256, 3, padding = 'same', kernel_initializer = 'he_normal', activation = 'relu'))
model.add(BatchNormalization(axis = -1))
model.add(Conv2D(256, 3, padding = 'same', kernel_initializer = 'he_normal', activation = 'relu'))
model.add(BatchNormalization(axis = -1))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512, activation = 'relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(256, activation = 'relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(len(imbalanced), activation = 'softmax'))
```

This is a convolutional neural network (CNN) architecture with 4 sets of convolutional, batch normalization, max pooling, and dropout layers, followed by 3 fully connected layers with batch normalization and dropout.

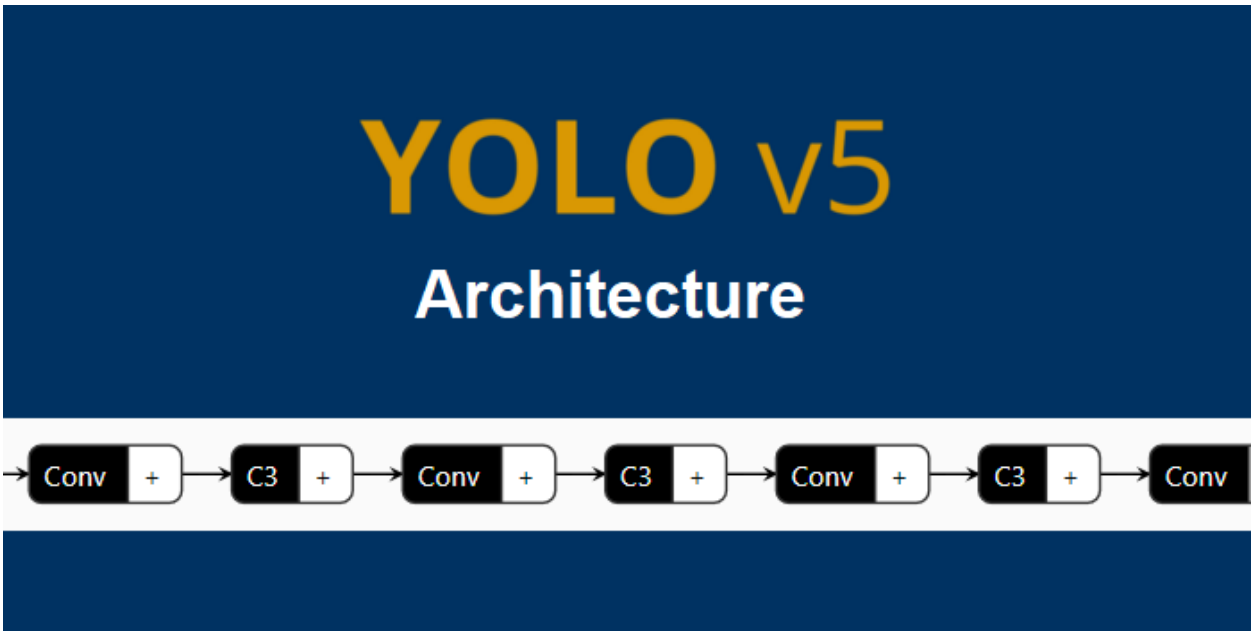
The first layer is a 2D convolutional layer with 32 filters, a kernel size of 3x3, and ReLU activation, which takes an input shape of (96, 96, 3) and uses the He normal initialization method. It is followed by batch normalization and max pooling with a pool size of 2x2 and a dropout rate of 0.25.

The next three sets of convolutional layers have 64, 128, and 256 filters respectively, each with a kernel size of 3x3, ReLU activation, He normal initialization, and batch normalization. Each set is followed by max pooling, with the second and fourth sets also having a dropout rate of 0.25.

After the convolutional layers, a flatten layer is added to convert the output to a 1D vector, which is then passed through two fully connected layers with 512 and 256 neurons respectively, each with ReLU activation, batch normalization, and a dropout rate of 0.5.

Finally, a dense layer with a softmax activation function is added, which produces an output with the same number of dimensions as the number of classes in the imbalanced dataset.

Car License Plate Detection



The model architecture of the model is as follows:

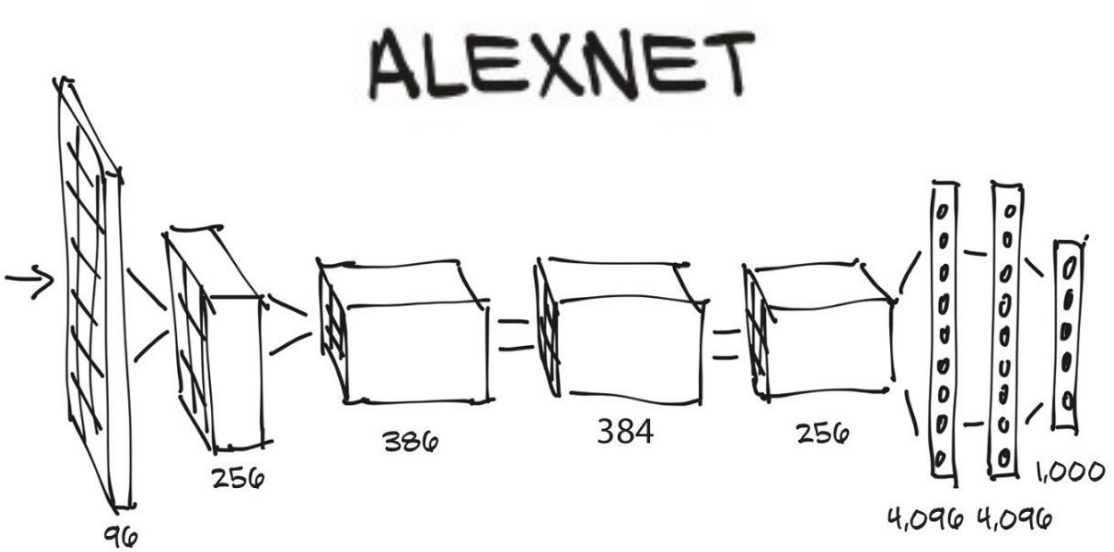
```
!python train.py --img 480 --batch 128 --epochs 100 --data plate_detection.yaml --weights yolov5s.pt --
```

```
      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      99/99      15.2G    0.02225   0.006538          0        190        480: 100% 3/3 [00:15<00:0
0,  5.18s/it]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 1/1
[00:01<00:00,  1.25s/it]
                   all         87         96       0.93      0.974      0.939      0.526

100 epochs completed in 0.518 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 14.3MB
Optimizer stripped from runs/train/exp/weights/best.pt, 14.3MB

Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model summary: 157 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 1/1
[00:03<00:00,  3.05s/it]
                   all         87         96      0.908      0.969      0.938      0.543
Results saved to runs/train/exp
```

YOLOv5 (You Only Look Once version 5) is an object detection model architecture that uses a single neural network to simultaneously predict bounding boxes and class probabilities for multiple objects in an input image. YOLOv5 model architecture:

- The input image is first resized to a fixed size of 640x640 pixels.
- The image is then passed through a backbone network, which is a modified version of the EfficientNet architecture. The backbone network has a series of convolutional layers that extract features from the image at different scales.
- The feature maps produced by the backbone network are then passed through a neck network, which is a series of convolutional layers that fuse the features from different scales and resolutions.
- The fused feature maps are then passed through a head network, which is responsible for predicting bounding boxes and class probabilities for the objects in the image. The head network has a series of convolutional layers, followed by a global average pooling layer, and finally a set of fully connected layers that output the final predictions.
- The output of the head network is a tensor with shape (N, S, S, 3*(5+C)), where N is the batch size, S is the grid size (in YOLOv5, S=20), C is the number of object classes, and 3*(5+C) is the number of predicted values per grid cell. The 3 corresponds to the number of predicted bounding boxes per grid cell, and the 5+C corresponds to the predicted coordinates, objectness score, and class probabilities for each bounding box.
- The final output of the model is a set of bounding boxes and class probabilities for the objects in the input image.

Lukemia Detection



The model architecture of the model is as follows:

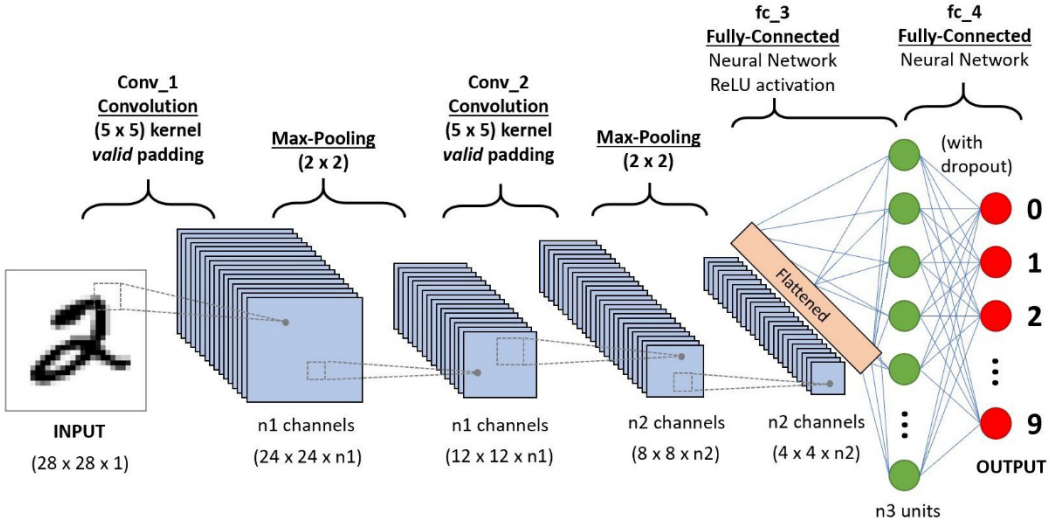
```
def alexnet(train_generator,epchs,val_generator,batch_size,lr):
    import keras as ks
    model = ks.models.Sequential()
    opt = ks.optimizers.Nadam(learning_rate=lr,decay=1e-4)
    callback = ks.callbacks.EarlyStopping(monitor="val_loss",
                                          patience=10,
                                          verbose=2)
    #building architecture
    #Adding layers
    model.add(ks.layers.Conv2D(96,(11,11),
                               strides=4,
                               activation="elu",
                               name="layer1",
                               input_shape=(227,227,3)))
    model.add(ks.layers.MaxPooling2D((3,3),strides=2,name="layer2"))
    model.add(ks.layers.Conv2D(256,(5,5),padding="valid",activation="elu",name="layer3"))
    model.add(ks.layers.MaxPooling2D((3,3),strides=2,name="layer4"))
    model.add(ks.layers.Conv2D(384,(3,3),padding="valid",activation="elu",name="layer5"))
    model.add(ks.layers.Conv2D(384,(3,3),padding="valid",activation="elu",name="layer6"))
    model.add(ks.layers.Conv2D(256,(3,3),padding="valid",activation="elu",name="layer7"))
    model.add(ks.layers.MaxPooling2D((3,3),strides=2,name="layer8"))
    model.add(ks.layers.Flatten())
    model.add(ks.layers.Dense(1024,activation="elu",
                              kernel_initializer="he_normal",
                              kernel_regularizer=ks.regularizers.l2(0.01)))
    model.add(ks.layers.BatchNormalization())
    model.add(ks.layers.Dense(1024,activation="elu",
                               kernel_regularizer=ks.regularizers.l2(0.01)))
    model.add(ks.layers.BatchNormalization())

    model.add(ks.layers.Dense(1,activation="sigmoid",
                              kernel_initializer="glorot_uniform",
                              name="output"))#2 classes
```

AlexNet is a deep convolutional neural network (CNN) architecture that was introduced in 2012 by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. It was one of the first deep learning models to achieve state-of-the-art performance on the ImageNet dataset. AlexNet model architecture:

- The input to the network is a 227x227 RGB image.
- The first layer is a convolutional layer with 96 filters of size 11x11 and a stride of 4. The activation function is ReLU.
- The output of the first layer is passed through a normalization layer and then a max pooling layer with a size of 3x3 and a stride of 2.
- The next layer is another convolutional layer with 256 filters of size 5x5 and a stride of 1. The activation function is ReLU.
- The output of the second layer is again normalized and max pooled.
- The third layer is a convolutional layer with 384 filters of size 3x3 and a stride of 1. The activation function is ReLU.

- The fourth layer is another convolutional layer with 384 filters of size 3x3 and a stride of 1. The activation function is ReLU.
- The fifth layer is a final convolutional layer with 256 filters of size 3x3 and a stride of 1. The activation function is ReLU.
- The output of the fifth layer is max pooled and then passed through three fully connected layers with 4096 neurons each. The activation function is ReLU.
- The final layer is a fully connected layer with 1000 neurons, one for each class in the ImageNet dataset. The activation function is softmax.

Handwritten Digit Recognition



The model architecture of the model is as follows:

```
model = Sequential()

model.add(Conv2D(32, kernel_size=(3,3), activation='relu', input_shape=INPUT_SHAPE))
model.add(MaxPool2D((2,2)))

model.add(Conv2D(64, kernel_size=(3,3), activation='relu'))
model.add(MaxPool2D((2,2)))

model.add(Flatten())

model.add(Dense(128, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(64, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(10, activation='softmax'))
```

- This is a simple convolutional neural network (CNN) architecture with two sets of convolutional and max pooling layers, followed by three fully connected layers with dropout.

- The first layer is a 2D convolutional layer with 32 filters, a kernel size of 3x3, and ReLU activation, which takes an input shape of INPUT_SHAPE. It is followed by max pooling with a pool size of 2x2.

- The second set of convolutional layer has 64 filters, a kernel size of 3x3, and ReLU activation, followed by max pooling with a pool size of 2x2.

- After the convolutional layers, a flatten layer is added to convert the output to a 1D vector, which is then passed through two fully connected layers with 128 and 64 neurons respectively, each with ReLU activation and dropout rate of 0.2.

- Finally, a dense layer with a softmax activation function is added, which produces an output with 10 dimensions, corresponding to the number of classes in the dataset.

Mathematical Concepts used:

Convolution:

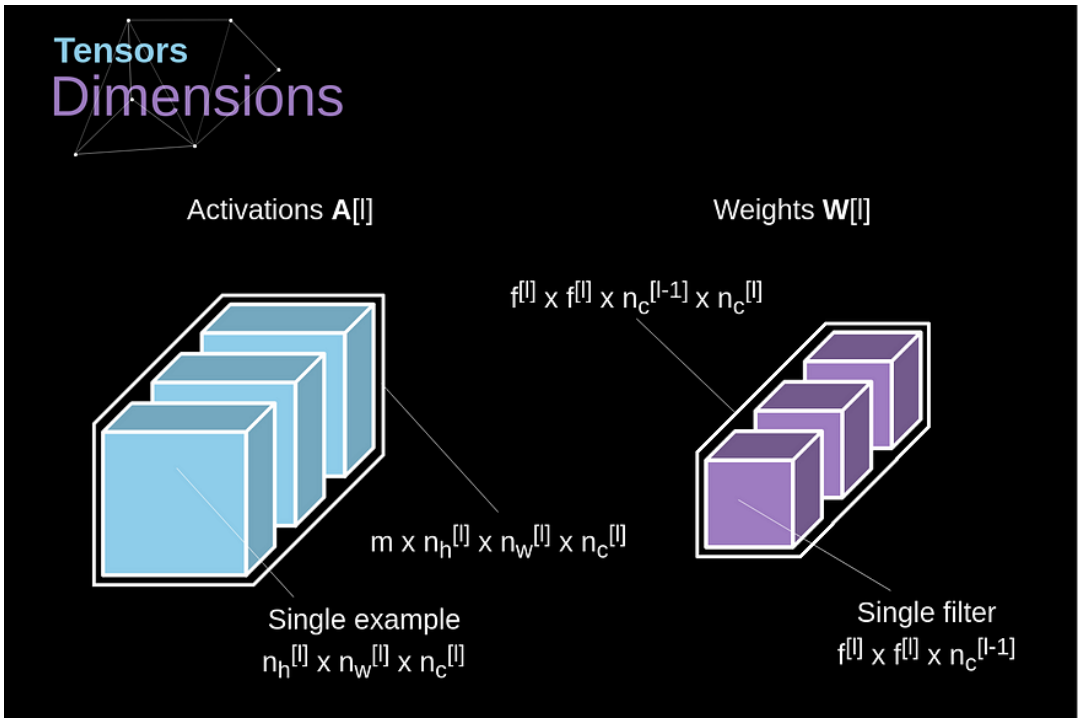
Convolution is often represented mathematically with an asterisk * sign. If we have an input image represented as X and a filter represented with f, then the expression would be:

$$Z = X * f$$

$$G[m,n] = (f * h)[m,n] = \sum_j \sum_k h[j,k] f[m-j, n-k]$$

The dimensions of the output matrix - taking into account padding and stride - can be calculated using the following formula.

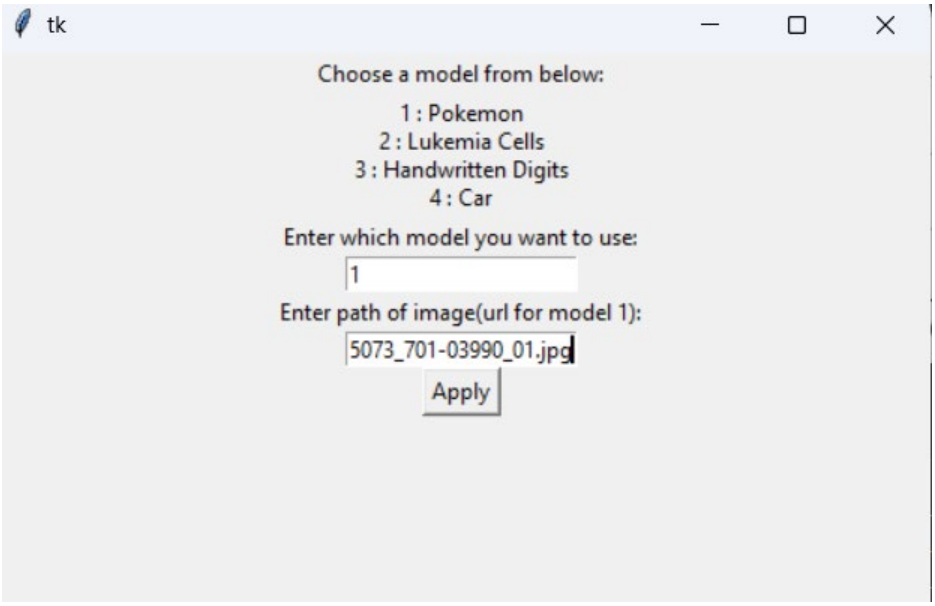
$$n_{out} = \left\lfloor \frac{n_{in} + 2p - f}{s} + 1 \right\rfloor$$

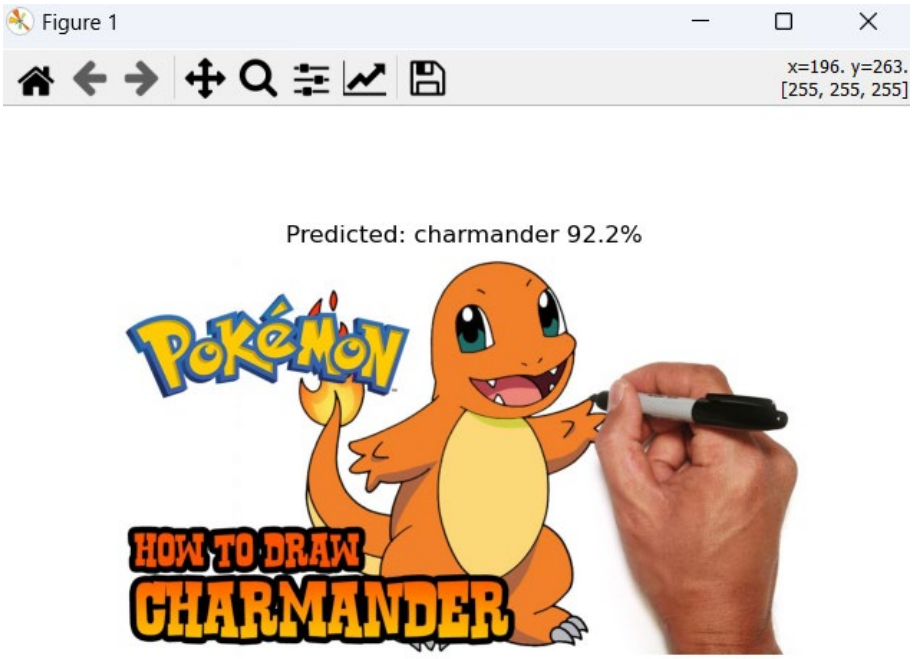


Results:

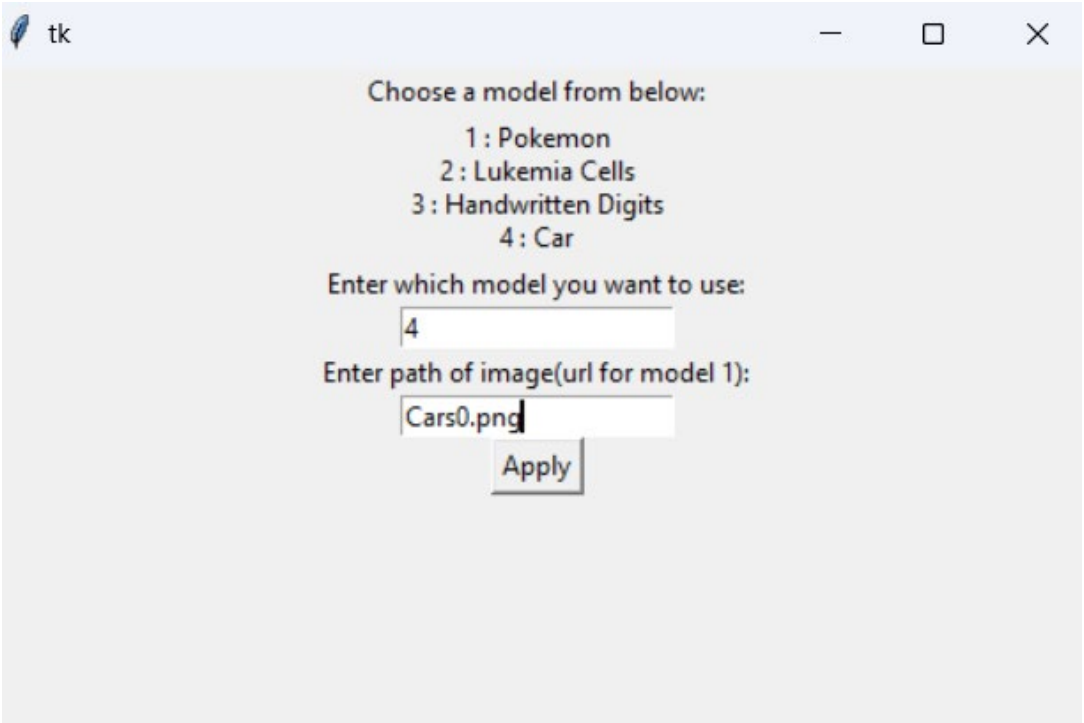
Below are the screenshots of the GUI created using tkinter library of Python and also the outputs of each model which are implemented in this project through the GUI.

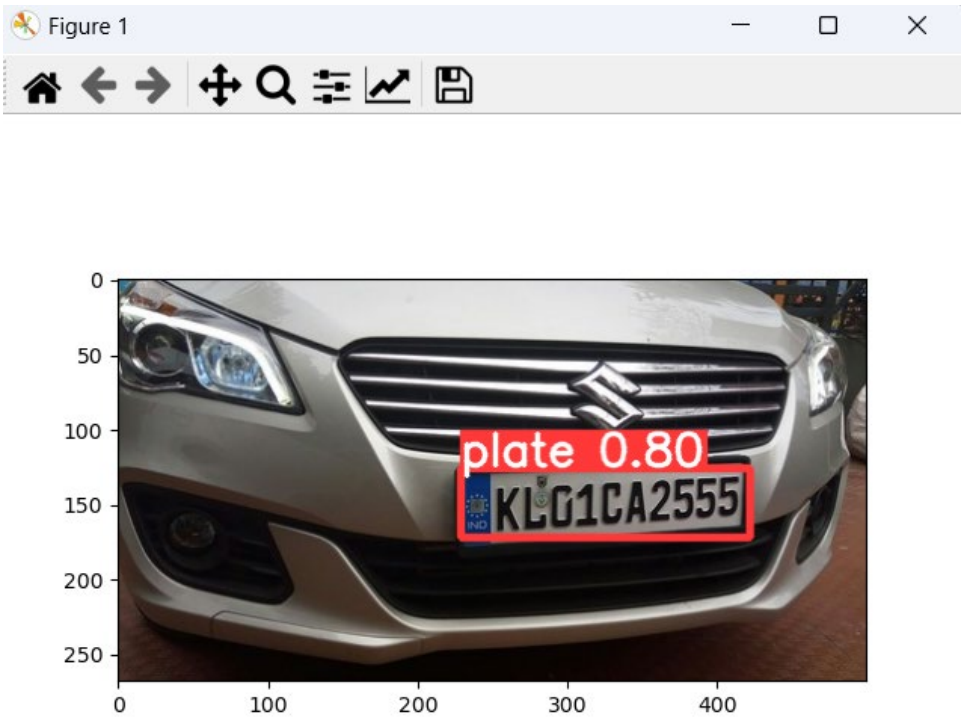
Pokemon Classifier



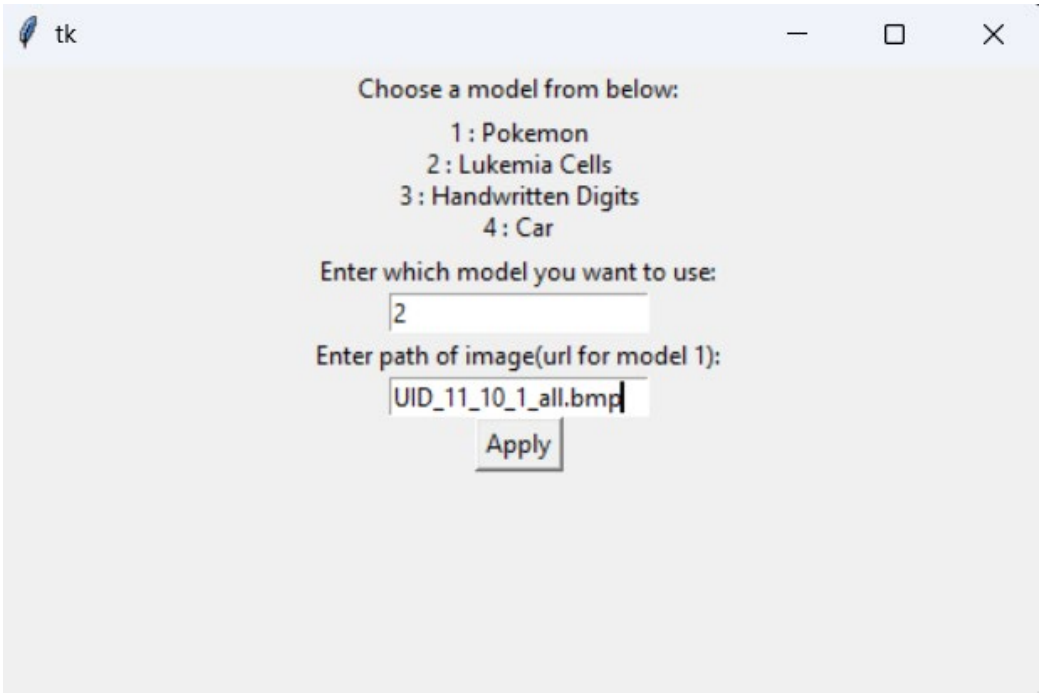


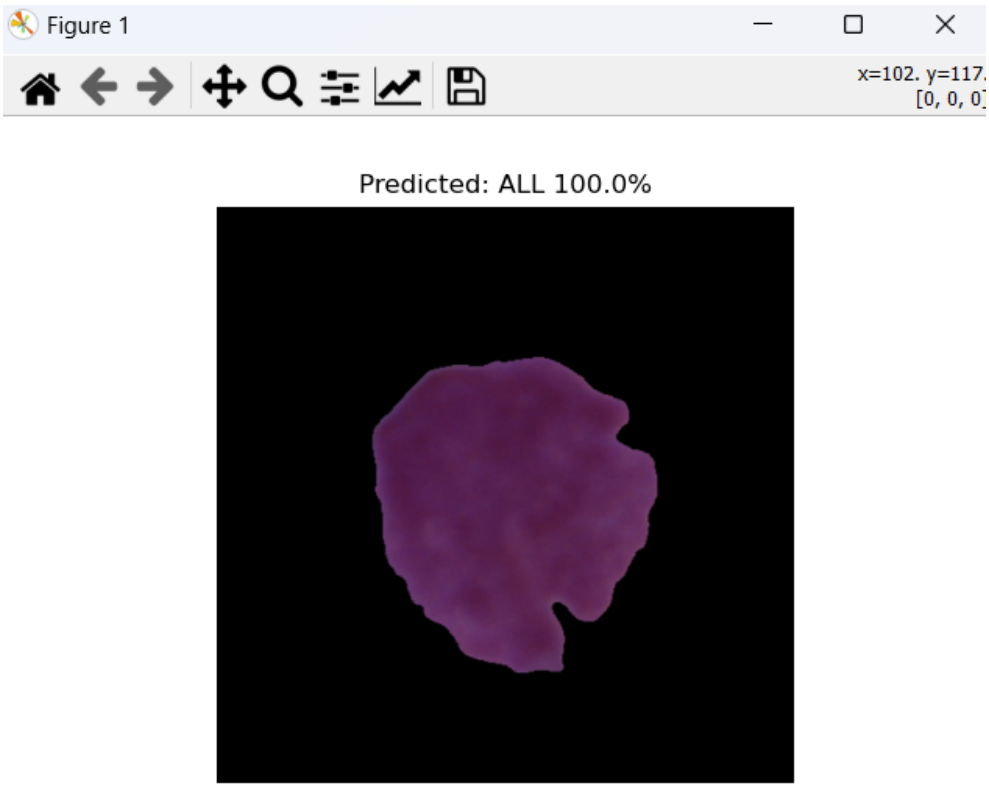
Car License Plate Detection



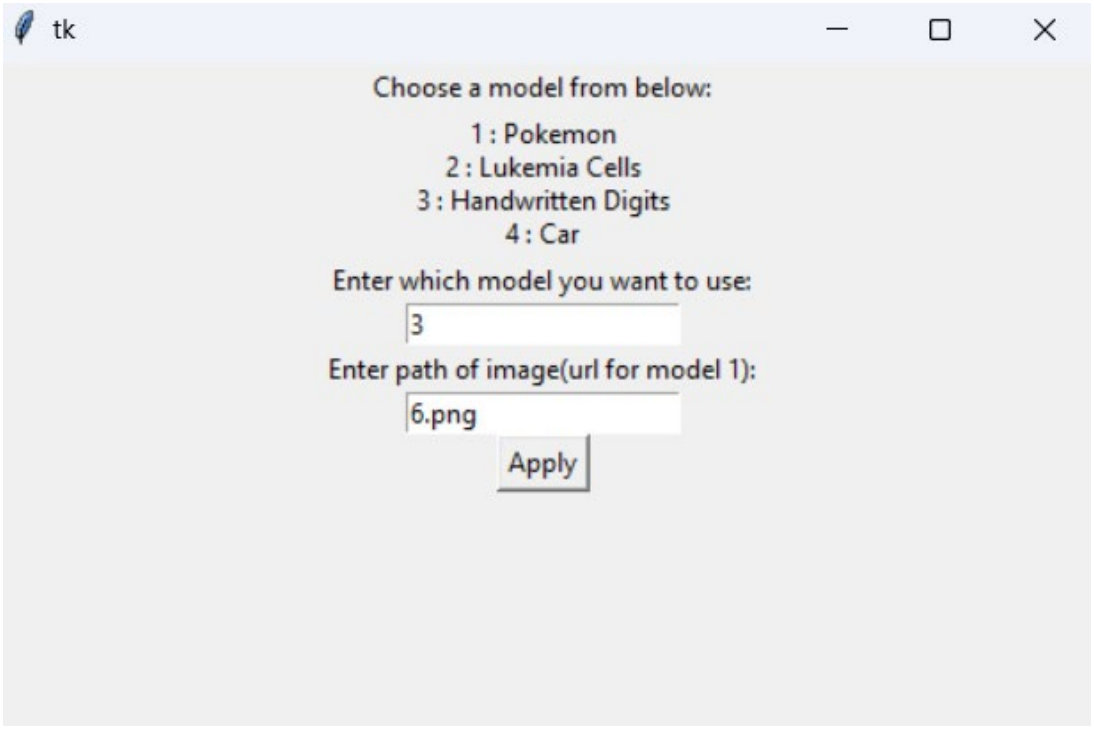


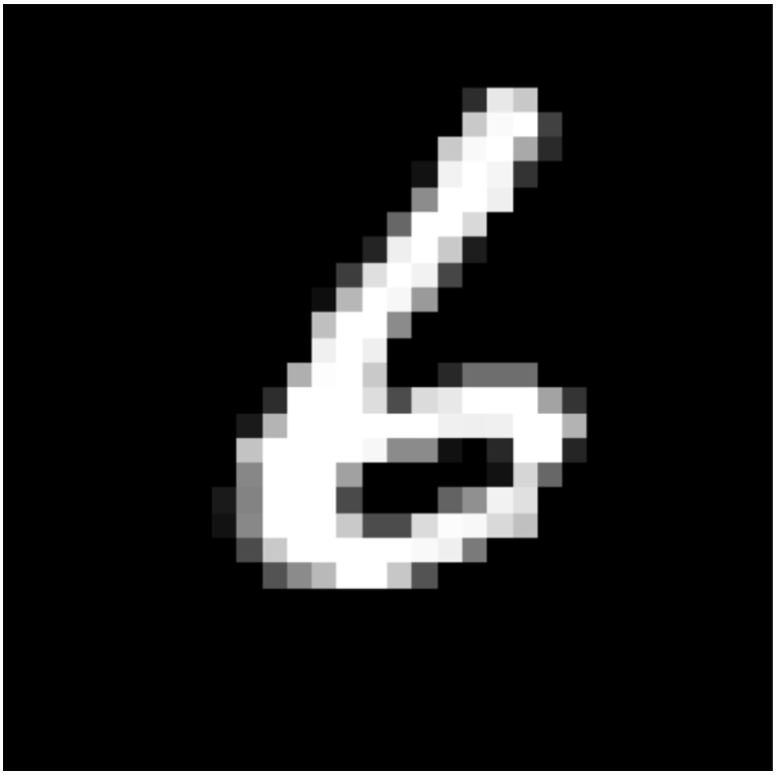
Lukemia Detection



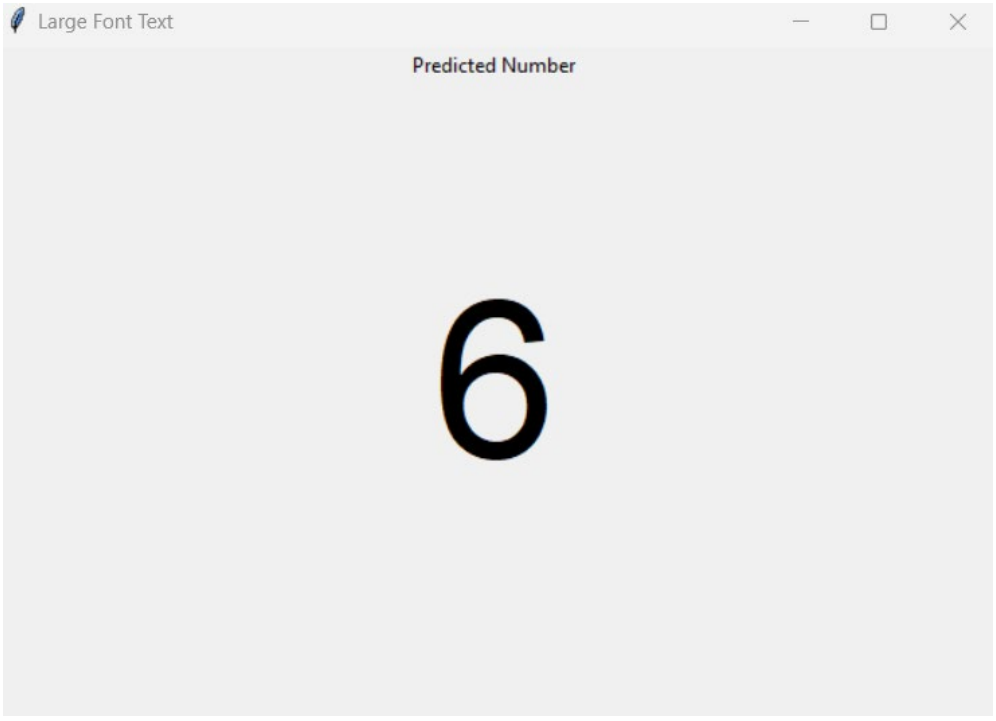


Handwritten Digit Recognition





6.png



Future Insights

- **Increased Accuracy: By leveraging the strengths of multiple models, the** project may lead to improved accuracy and performance on a variety of tasks. For example, combining a computer vision model with a natural language processing model could help improve accuracy in tasks that involve both visual and textual information.
- **Enhanced Efficiency:** In addition to improving accuracy, the project may also lead to enhanced efficiency. By allowing models to work together on a single task, you may be able to reduce the amount of computation required to achieve a given result. This could be particularly valuable in resource-constrained environments such as mobile devices or edge computing.
- **Improved Generalization:** One potential benefit of combining multiple models is that it could help improve the generalization capabilities of AI systems. By using models that have been trained on different datasets or with different techniques, your project could help overcome some of the limitations of individual models and lead to more robust and adaptable AI systems.
- **Novel Applications:** As project progresses, we may discover new and unexpected ways in which multiple models can be combined to solve complex problems. These insights could lead to the development of novel applications and new areas of research in the field of AI.
- **Ethical Considerations:** As AI systems become increasingly powerful and sophisticated, there are growing concerns about the potential impact on society. The project may help address some of these concerns by developing AI systems that are more transparent, explainable, and trustworthy. This could help foster greater public trust in AI systems and promote the responsible development and deployment of AI technologies.

Conclusion

In this project, we used different models trained for different tasks to work together like the multitasking ability of the brain to give correct output for a given task has the potential to lead to significant advancements in the field of AI. By combining the strengths of multiple models, we can improve accuracy, efficiency,

generalization, and develop novel applications that were previously impossible to achieve with a single model.

As AI systems become more ubiquitous in our society, it is crucial to ensure that they are developed and deployed in a responsible and ethical manner. The insights gained from this project may help address some of the ethical concerns surrounding AI by promoting transparency, explainability, and trustworthiness.

Moving forward, there are still many challenges to overcome in developing AI systems that can effectively collaborate and work together. However, the potential benefits are significant, and the development of such systems could have far-reaching implications for a wide range of applications and industries. Overall, this project represents an important step forward in the ongoing development of AI technologies.

References

1. Andrew Gibiansky: Convolutional Neural Networks
2. Jamshaid Shahir Multi-Label; Classification Of Pokemon Types With Tensorflow
3. Matan Shelomi , Andrew Richards , Ivana Li , Yukinari Okido; A Phylogeny And Evolutionary History Of The Pokémon
4. Code Ai Blogs; Classifying Pokémon Images With Machine Learning
5. N.Palanivel Ap; T. Vigneshwaran; M.Sriv Arappradhan; R. Madhanraj; Automatic Number Plate Detection In Vehicles Using Faster R-Cnn
6. Abhishek Kashyap; B. Suresh; Anukul Patil; Saksham Sharma; Ankit Jaiswal; Automatic Number Plate Recognition
7. Umar Shahid; Number Plate Recognition System
8. Gaurav Srivastav Automatic Number Plate Recognition
9. Vanshika Rai, Deepali Kamthania; Automatic Number Plate Recognition
10. Junaid Khan, Kyungsup Kim; An Efficient Cnn-Based Automated Leukemia Diagnosis Using Microscopic Blood Smear Images And Subtypes Classification
11. Shakir M Abass, Adnan Mohsin Abdulazeez; Detection And Classification Of Leukocytes In Leukemia Using Yolov2 With Cnn
12. V. Anagha, A. Disha, B. Y. Aishwarya, R. Nikkita & Vidyadevi G. Biradar; Detection Of Leukemia Using Convolutional Neural Network
13. Angelo Genovese; Mahdi S. Hosseini; Vincenzo Piuri; Histopathological Transfer Learning For Acute Lymphoblastic Leukemia Detection
14. Ahmed T. Sahlol, Philip Kollmannsberger & Ahmed A. Ewees; Efficient Classification Of White Blood Cell Leukemia With Improved Swarm Optimization Of Deep Features

15. Aren Carpenter; Detecting Leukemia With A Cnn
16. Ritik Dixit, Rishika Kushwah, Samay Pashine; Handwritten Digit Recognition Using Machine And Deep Learning Algorithms
17. Shubham Mendapara, Krish Pabani, Yash Paneliya; Handwritten Digit Recognition System
18. Anchit Shrivastava; Isha Jaggi; Sheifali Gupta; Deepali Gupta; Handwritten Digit Recognition Using Machine Learning: A Review
19. Arjun Seth; Akshat Bhandari;Trisha Sharma; Handwritten Digit Classification
20. Priya, Rajendra Singh, Dr. Soni Changlani; Review On Handwritten Digit Recognition