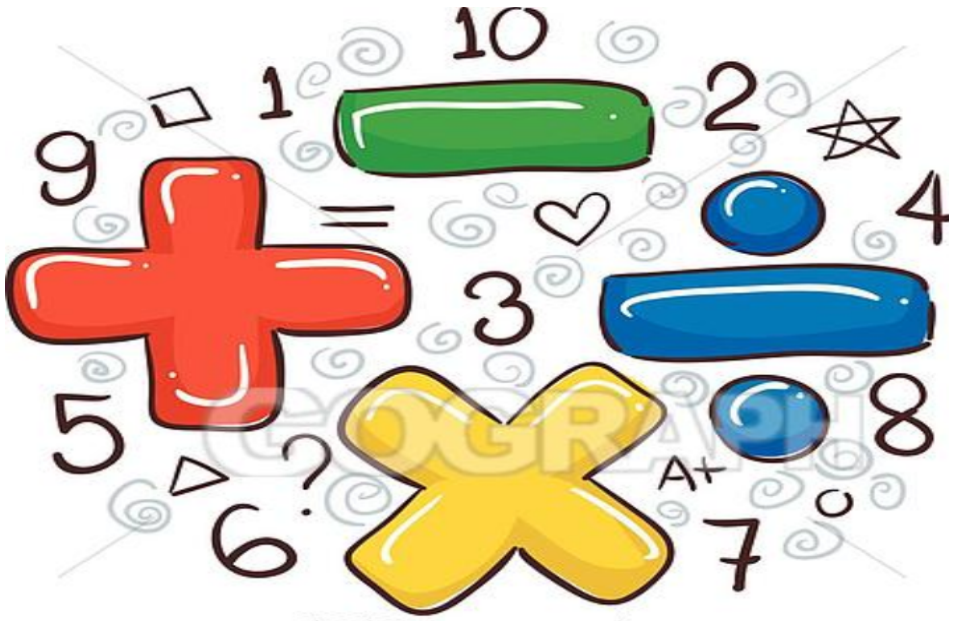
INK TO MATH

"App to solve your basic arithmetic equations"

GITHUB REPOSITORY LINK - https://github.com/am15h/InkToMath



Amish Garg 18114004

Tanav Shah 18114077

Vinay Kumar Jain 18114082

Yashaswi Jaiswal 18114083

ABSTRACT

The application helps the users to solve their arithmetical equations in a fast and creative way. The users type or sketch their equations on the application's black board. With a single touch, the application then gives the result corresponding to the input of the user. Owing to the availability of current resources, we have developed the application to solve the most basic arithmetic equations like addition, subtraction, etc. And with further availability of resources, we plan to develop a full fledged application including Optical Character Recognition (OCR) and more advanced functions.

INTRODUCTION

Mobile applications grew in less than two decades to achieve the status of the largest information repository in human history. By providing efficient, fast, consistent and authentic tools in the form of internet and mobile applications, information technology is penetrating human life and is playing an important role in changing lives of so many people around the globe.

Today many traditional industrial firms are moving towards utilizing information technology including mobile applications. Mobile applications run banking transactions, air traffic, and emergency room equipment. With this increasingly wide range of technology, both the hardware and software elements of the system can have failures which may result in catastrophic effects.

In this project our main is to develop an application for the processing of any mathematical expression with given arithmetic operators. Any customer C, who has this app, can use it to get values for simple mathematical expressions. The app is expected to give more than human level accuracy and can solve problems of a large section of the mobile using community in a few seconds.

USE CASE

The app's main focus is to help the students and the teachers in any institution to carry out their day-to-day computations with greater speeds and with more accuracy, thereby boosting their efficiency. Any customer C can compute trivial expressions without the use of any manual input to the calculator, thereby saving time and enhancing accuracy meanwhile boosting the efficiency.

Another set of targeted customers for this app is the Finance Oriented Firms. Mainly the chartered accountants can use this app in their audits thus, utilizing their time in the best possible way. They generally need a software or an app which can help them with the calculations and an app like this can prove to be a boon for them.

Although these are some of the primary target customers, this app can also be used by someone for day-to-day calculations. This app can save a lot of time which can be utilized for the greater good of the society.

TECHNICAL DETAILS

The project comprises a mobile application that can be used to compute values of handwritten equations. The main tools and technologies that are used include :

- Android Studio
- Flutter for app development
- Convolutional Neural Networks
- Deep Learning
- MNIST dataset
- TensorFlow Framework for building the DL model
- TensorFlow Lite Library for integrating the model with the mobile app environment
- Colab for training the DL model

Overview

We have built the mobile application in Android Studio Environment, which is most widely used for mobile application development. We have added the functionality for the user to draw symbols and digits in the app, which will be recognized and further, the result will be computed as and when commanded by the user. The input from the user is passed through a convolutional neural network, which is already trained, and from its output, the symbol drawn is recognized. While building the model, we have used the TensorFlow framework to code the different layers of the CNN. The model is trained on the MNIST dataset, which is a labelled dataset of digits from 0-9. We have appended this dataset with suitable training examples for recognition of various arithmetic symbols.

The detailed usage and requirement for the various tools and technologies are described below:

Android Studio :

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development.

Android Studio supports all the same programming languages of IntelliJ (and CLion) e.g. Java, C++, and more with extensions, such as Go; and Android Studio 3.0 or later supports Kotlin and "all Java 7 language features and a subset of Java 8 language features that vary by platform version." External projects backport some Java 9 features. While IntelliJ states that Android Studio is built on supports all released Java versions, and Java 12, it's not clear to what level Android Studio supports Java versions up to Java 12 (the documentation mentions partial Java 8 support). At least some new language features up to Java 12 are usable in Android.

Deep Learning and Convolutional Neural Networks :

Deep learning (also known as **deep structured learning**) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised.

Deep learning architectures such as deep neural networks, deep belief networks, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance.

In deep learning, a **convolutional neural network** (**CNN**, or **ConvNet**) is a class of deep neural networks, most commonly applied to analysing visual imagery.[1] They are also known as **shift invariant** or **space invariant artificial neural networks** (**SIANN**), based on their shared-weights architecture and translation invariance characteristics.[2][3] They have applications in image and video recognition, recommender systems,[4] image classification, medical image analysis, natural language processing,[5] and financial time series.[6]

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.

MNIST Dataset

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 database is also widely used for training and testing in the field of machine learning. It was created by "re-mixing" the samples from NIST's original datasets.

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. The original creators of the database keep a list of some of the methods tested on it. In their original paper, they use a support-vector machine to get an error rate of 0.8%.[9] An extended dataset similar to MNIST called EMNIST has been published in 2017, which contains 240,000 training images, and 40,000 testing images of handwritten digits and characters.

TensorFlow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 on November 9, 2015.

TensorFlow is an end-to-end platform that makes it easy to build and deploy ML models.

TensorFlow Lite

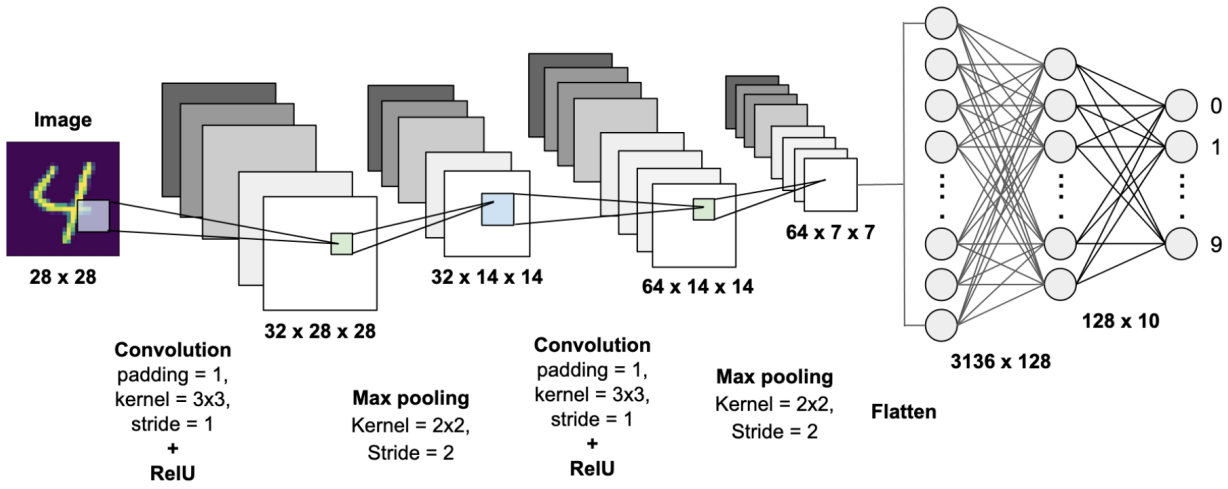
TensorFlow Lite is a set of tools to help developers run TensorFlow models on mobile, embedded, and IoT devices. It enables on-device machine learning inference with low latency and a small binary size.

TensorFlow Lite consists of two main components:

- The TensorFlow Lite interpreter, which runs specially optimized models on many different hardware types, including mobile phones, embedded Linux devices, and microcontrollers.
- The TensorFlow Lite converter, which converts TensorFlow models into an efficient form for use by the interpreter, and can introduce optimizations to improve binary size and performance.

The architecture of the Deep Learning Model:

```
model = keras.Sequential([
  keras.layers.InputLayer(input_shape=(28, 28)),
  keras.layers.Reshape(target_shape=(28, 28, 1)),
  keras.layers.Conv2D(filters=32, kernel_size=(3, 3), activation=tf.nn.relu),
  keras.layers.Conv2D(filters=64, kernel_size=(3, 3), activation=tf.nn.relu),
  keras.layers.MaxPooling2D(pool_size=(2, 2)),
  keras.layers.Dropout(0.25),
  keras.layers.Flatten(),
  keras.layers.Dense(13)
])
```



The model takes input as an image vector of dimensions (28, 28). This vector comprises the pixel values of the image of the symbol or digit to be recognized. These values are further normalized and fed into the Convolutional Neural Network (CNN).

The model has 2 layers of the CNN computing the convolution over the input image vector with filters of appropriate dimensions, followed by a MaxPooling layer. The earlier layers of the model are responsible for detecting simple features such as edges and corners, while the deeper layers are more responsible towards extracting more complex features.

After the Dropout, the Model has a Softmax Classifier. This is where the CNN outputs a vector of dimensions (13, 1) consisting of the probabilities of the given image being any of the required characters. The class with the highest probability is predicted as the output by the model. The 13 classes include digits 0-9 and operators +, -, x.

We have used the MNIST dataset for training the model for the digits 0-9. For the various operators, we have added specific labels and images with the exact dimensions into the MNIST dataset. We have prepared this complete dataset and trained the above model on it.

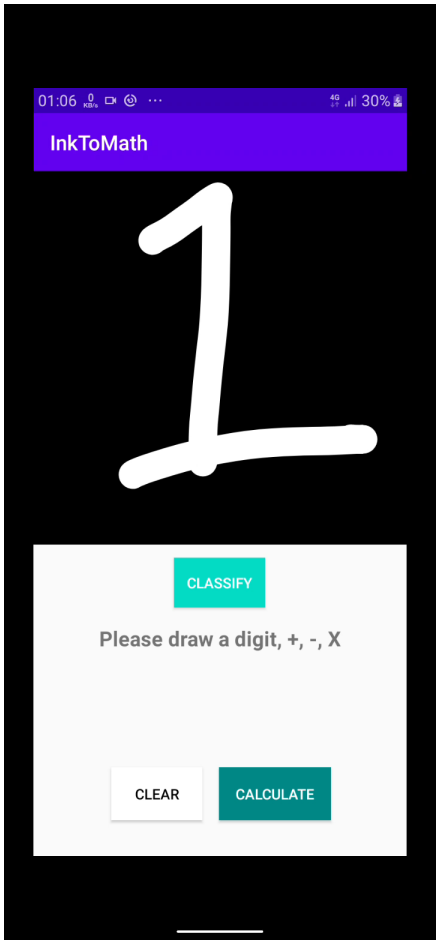
This model gives pretty good accuracy. We have particularly put in a lot of effort to tune the hyper-parameters and improve the overall architecture to get very high accuracy.

While running the app, the single image vector is passed through this CNN and based on the prediction made by the model (in the softmax layer) the digit or operator is recognized. This process is pretty fast and appropriate for real time usage.

Moreover, this app uses the .tflite file of the model, so that no internet is required while computing and the model can be integrated directly within the app. Had we not used the TensorFlow Lite library, we would have had to deploy the model on cloud and make the computations there itself. This process would be slow and would drain more resources from the user. Thus, this app is in itself quite efficient, both in terms of resource usage and accuracy of computation.

WORKING OF THE APPLICATION

After opening the application, the user can see a blackboard on which he can sketch the arithmetic equations. The application accepts the input one at a time and then **classifies** the input using the MNIST dataset including the operators +,-,*.

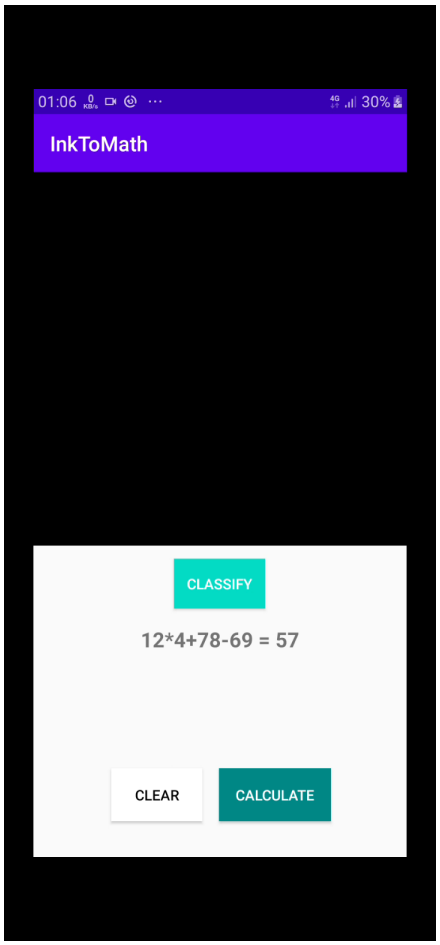


Here you can see a snapshot of the application. The application asks the user for a digit, and the operators +,-,*. And you can also see the sketch board on which 1 is currently sketched.

As the user gives input, the input equation appears below the blackboard. When the user has completed writing the equation, he can then tap on the **calculate** button which gives the output on the same box.

As the user gives input it appears as whole in the box below the black board or the sketch board.

There is also a **clear** button to allow the user to clear his sketch as he finds appropriate.



When the user taps on the calculate button, the application then shows the final result along with the input arithmetic equation.

CONCLUSION

The application gets creative as the user uses it. It can help a lot of people in the teaching sections. Especially who wants their daily calculations to be creative. It could be the teachers teaching the kindergarten students as the kids can find it attractive and creative. The app's main focus is to help the students and the teachers in any institution to carry out their day-to-day computations with greater speeds and with more accuracy, thereby boosting their efficiency. This app can save a lot of time which can be utilized for the greater good of the society. Thus this app can be a boost for many.