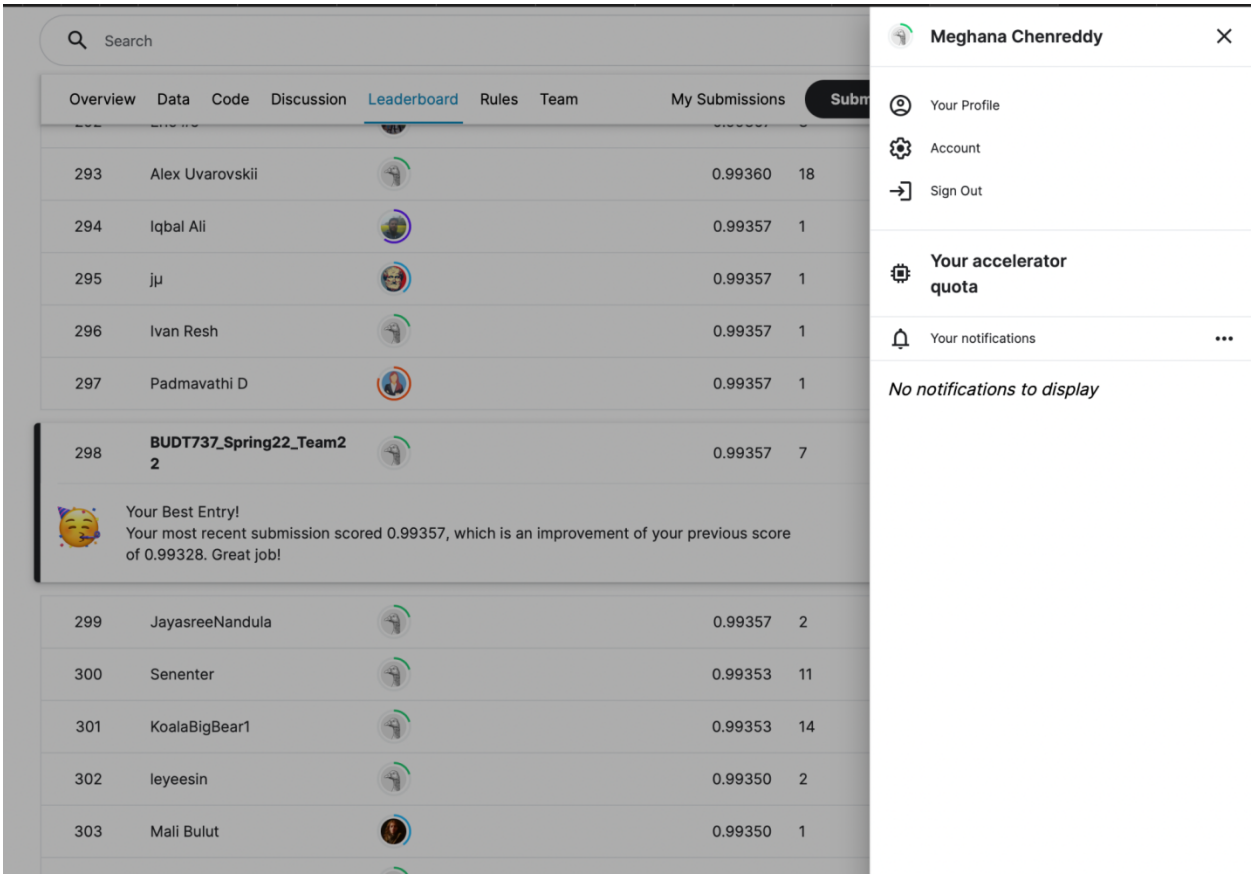
Course: BUDT 737 - Big Data & Artificial Intelligence for business
Submission: Final project

Team: BUDT737_Spring22_Team22
Team members:

Meghana Chenreddy meghanachenreddy (https://www.kaggle.com/meghanachenreddy)
Sai Jayasree Karthik Nandula jayasreenandula(https://www.kaggle.com/jayasreenandula)
Antara Kumar antarakr(https://www.kaggle.com/antarakr)
Rohin Bhagavatula rohinbhagavatula(https://www.kaggle.com/rohinbhagavatula)

Team Submission on Kaggle:



Abstract

We come across handwritten digits in almost every walk of life - postal mails, data entered on forms, bank cheques etc. Most of these have to be dealt with, through scanning the documents on which these digits are written. It's important for the digits to not be read incorrectly. Otherwise, mails could be delivered to wrong addresses, incorrect bank account numbers or incorrect amounts read from the cheques could lead to check bounces or incorrect transactions.
The problem at hand is to use machine learning algorithms & neural networks to train the data in such a way that one of them efficiently recognizes handwritten digits with as much maximum accuracy as possible on the validation data.

Dataset

The dataset used is from Kaggle competition - Digit Recognizer. Since the competition page explained the dataset clearly, we haven't explored the data structure.

Link to the data: https://www.kaggle.com/competitions/digit-recognizer/data?select=train.csv

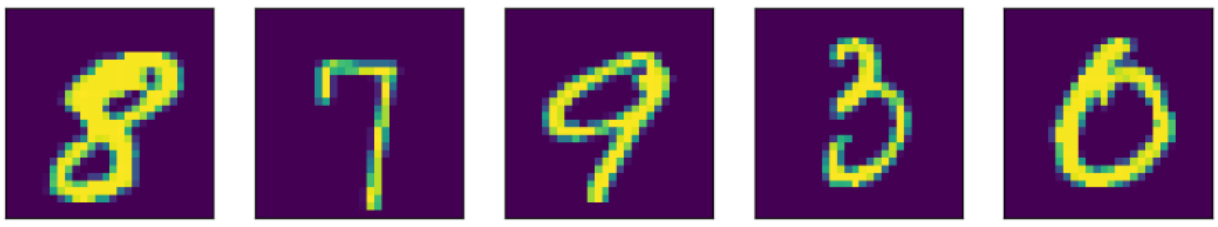
The data files train.csv and test.csv contain gray-scale images of hand-drawn digits, from zero through nine. The data consists of 785 columns - 1 Label and 784 pixel columns. Every row has a 28x28 image which is shown in tabular format in a row with 784 columns which consist of values from 0-255 indicating lightness/darkness of the respective pixel. Label column shows the digit drawn by the user.

The shape of the train dataset is (42000, 785) which means there are 42000 digits in the training dataset. The shape of the test dataset is (28000, 784) which means the label needs to be predicted for 28000 images.

Raw train data looks as below:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pix
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
3	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	

When plotted, images from sample of train dataset looks as below:



Methodology

There are 42000 rows (digit images) in the train dataset. The models can be trained directly on the complete train dataset but the purpose of validation data is to provide an unbiased evaluation of a model fit on the training dataset while fitting and fine-tuning models. We split the train dataset into train and validation sets with 80% and 20% sizes respectively.

We used the "ShuffleSplit" technique to split the dataset. ShuffleSplit will randomly sample the entire dataset during each iteration to generate a training set and a validation set. Hence the train set now has 33600 rows and the validation set has 8400 images.

Then we trained the models on "train" dataset i.e. 80% of actual train data provided in Kaggle and then validated the performance of the models on "validation" dataset i.e. 20% of the actual train data.

How to execute the code:

- Upload the file "kaggle.json"(By triggering the API on Kaggle) in the files section on Google Colab.
- Use GPU as the hardware accelerator
- Save the file and execute all the cells
- Make sure to upload the data file - kaggle.json each time the session re-starts.
-

We have used the following 6 machine-learning based classifiers for validating the train dataset:

- Logistic Regression
- Stochastic Gradient Descent
- Support Vector Machine
- Decision Tree
- Random Forest
- Convolutional Neural Network

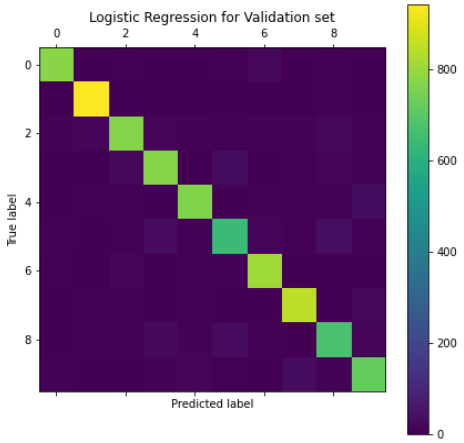
Logistic Regression:

The goal of a multiple logistic regression is to find an equation that best predicts the probability of a value of the Y variable as a function of the X variables. The independent variables then can be measured on a new value and the probability of that variable having a particular value of the dependent variable can be estimated.

An accuracy of **91.75%** was achieved on the validation data.

Logistic Regression Classification Report:

	precision	recall	f1-score	support
0	0.95	0.95	0.95	813
1	0.96	0.98	0.97	961
2	0.92	0.89	0.91	860
3	0.91	0.89	0.90	863
4	0.93	0.92	0.93	827
5	0.88	0.85	0.86	756
6	0.93	0.96	0.94	841
7	0.93	0.94	0.94	899
8	0.86	0.87	0.87	768
9	0.89	0.89	0.89	812
accuracy			0.92	8400
macro avg	0.92	0.92	0.92	8400
weighted avg	0.92	0.92	0.92	8400



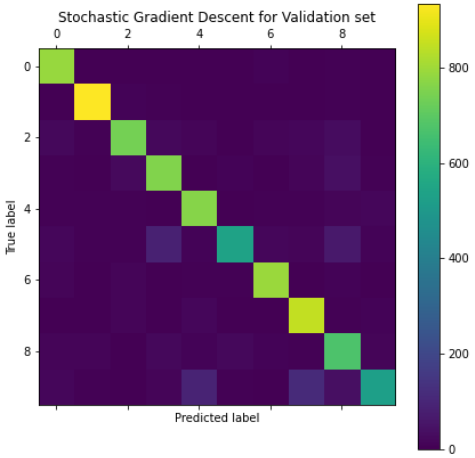
Stochastic Gradient Descent:

Stochastic gradient descent is an optimization algorithm and it is used very often in machine learning applications to find the model parameters that correspond to the best fit between actual and predicted outputs. It's a naive approach but a very powerful technique.

An accuracy of **87.58%** was achieved on the validation data.

Stochastic Gradient Descent Classification Report:

	precision	recall	f1-score	support
0	0.90	0.97	0.93	813
1	0.96	0.97	0.97	961
2	0.91	0.86	0.88	860
3	0.84	0.88	0.86	863
4	0.83	0.93	0.88	827
5	0.91	0.71	0.80	756
6	0.94	0.94	0.94	841
7	0.84	0.94	0.89	899
8	0.76	0.87	0.81	768
9	0.90	0.65	0.75	812
accuracy			0.88	8400
macro avg	0.88	0.87	0.87	8400
weighted avg	0.88	0.88	0.87	8400



Support Vector Machine:

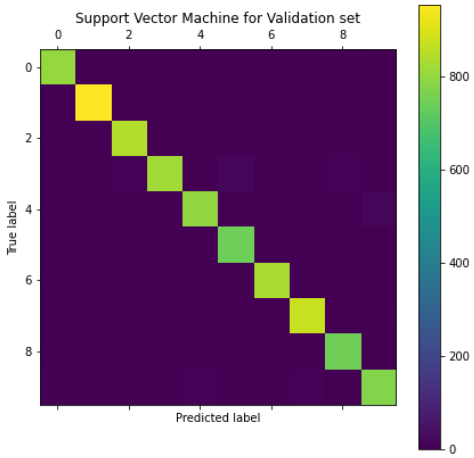
Support vector machines are supervised machine learning algorithms. They are suitable for both regression and classification tasks. One or more hyperplanes are constructed in n-dimensional

space and the hyperplane having the largest distance to the nearest training points of any class is considered as the best separation for the classifier.

An accuracy of **97.47%** was achieved on the validation data.

Support Vector Machine Classification Report:

	precision	recall	f1-score	support
0	0.98	0.99	0.98	813
1	0.99	0.99	0.99	961
2	0.97	0.98	0.98	860
3	0.98	0.95	0.97	863
4	0.97	0.97	0.97	827
5	0.96	0.98	0.97	756
6	0.98	0.99	0.98	841
7	0.97	0.97	0.97	899
8	0.96	0.97	0.97	768
9	0.97	0.96	0.96	812
accuracy			0.97	8400
macro avg	0.97	0.97	0.97	8400
weighted avg	0.97	0.97	0.97	8400



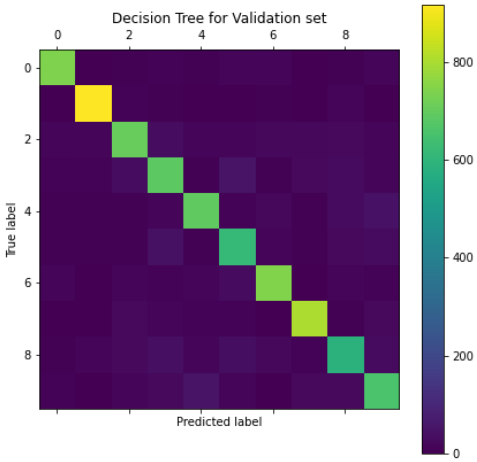
Decision Tree:

A decision tree is a supervised machine learning technique. Decision trees make predictions based on how a previous set of questions were answered in respective with the data.

An accuracy of **85.14%** was achieved on the validation data.

Decision Tree Classification Report:

	precision	recall	f1-score	support
0	0.92	0.91	0.91	813
1	0.95	0.95	0.95	961
2	0.84	0.82	0.83	860
3	0.79	0.80	0.79	863
4	0.86	0.84	0.85	827
5	0.79	0.81	0.80	756
6	0.88	0.88	0.88	841
7	0.90	0.90	0.90	899
8	0.78	0.77	0.77	768
9	0.79	0.81	0.80	812
accuracy			0.85	8400
macro avg	0.85	0.85	0.85	8400
weighted avg	0.85	0.85	0.85	8400



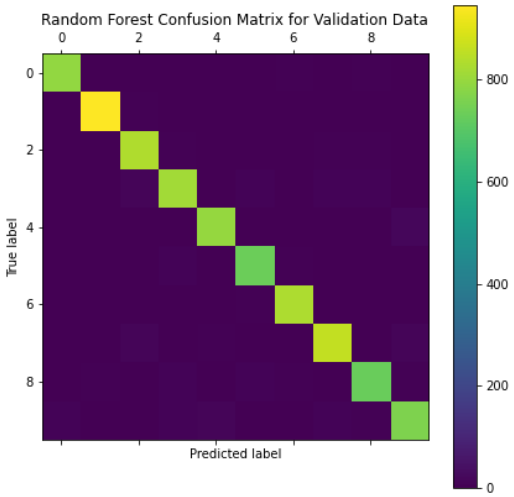
Random Forest:

Random Forest Classifier is an ensemble learning model.The main idea behind random forest is to prevent decision trees from overfitting. Random forest makes a decision based on votes from many decision trees which are created by using a subset of the attributes. Random Forest is suitable for both regression and classification tasks. It is also less sensitive to Outliers.

An accuracy of **96.46%** was achieved on the validation data.

Random Forest Classification Report:

	precision	recall	f1-score	support
0	0.97	0.98	0.97	813
1	0.99	0.98	0.99	961
2	0.95	0.97	0.96	860
3	0.96	0.94	0.95	863
4	0.97	0.96	0.97	827
5	0.96	0.97	0.96	756
6	0.97	0.99	0.98	841
7	0.97	0.96	0.97	899
8	0.95	0.95	0.95	768
9	0.95	0.94	0.95	812
accuracy			0.96	8400
macro avg	0.96	0.96	0.96	8400
weighted avg	0.96	0.96	0.96	8400



Convolution Neural Network:

CNN's are used to perform analysis on images and visuals. These classes of neural networks can input a multi-channel image and work on it easily with minimal preprocessing required. CNN can learn multiple layers of feature representations of an image by applying filters, or transformations. We have used a network with 6 convolutional layers, 3 maxpool layers and few other layers.

Convolution layer:
This is the main part of CNN. It is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed.

Pooling layer:
In general, convolutional layers are followed by a pooling layer. The main aim of this layer is to decrease the size of the convolved features to reduce the computational costs. There are different kinds of pooling techniques. In Max Pooling, the largest element is taken from the feature map. Average Pooling calculates the average of the elements in an image section. In our model, we have used **MaxPool**.

Dropout layer:
A dropout layer is used to drop a few neurons from the neural network during the training process resulting in reduced size of the model and to overcome model overfitting.

Activation functions:
These are one of the most important parameters of the CNN model. They decide which information of the model should move in the forward direction and which ones should not be in the network. There are several commonly used activation functions such as the ReLU, Softmax, tanH and the Sigmoid functions. In our model, we have used the **ReLU** function.

The following is the summary of the CNN model.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 64)	640
conv2d_1 (Conv2D)	(None, 28, 28, 64)	36928
batch_normalization (BatchNormalization)	(None, 28, 28, 64)	256
max_pooling2d (MaxPooling2D)	(None, 14, 14, 64)	0
dropout (Dropout)	(None, 14, 14, 64)	0
conv2d_2 (Conv2D)	(None, 14, 14, 128)	73856
conv2d_3 (Conv2D)	(None, 14, 14, 128)	147584
batch_normalization_1 (BatchNormalization)	(None, 14, 14, 128)	512
max_pooling2d_1 (MaxPooling2D)	(None, 7, 7, 128)	0
dropout_1 (Dropout)	(None, 7, 7, 128)	0
conv2d_4 (Conv2D)	(None, 7, 7, 256)	295168
conv2d_5 (Conv2D)	(None, 7, 7, 256)	590080

batch_normalization_2 (Batc (None, 7, 7, 256) 1024
hNormalization)

max_pooling2d_2 (MaxPooling (None, 3, 3, 256) 0
2D)

dropout_2 (Dropout) (None, 3, 3, 256) 0

flatten (Flatten) (None, 2304) 0

dense (Dense) (None, 2048) 4720640

dropout_3 (Dropout) (None, 2048) 0

dense_1 (Dense) (None, 512) 1049088

dropout_4 (Dropout) (None, 512) 0

dense_2 (Dense) (None, 128) 65664

batch_normalization_3 (Batc (None, 128) 512
hNormalization)

dropout_5 (Dropout) (None, 128) 0

dense_3 (Dense) (None, 10) 1290

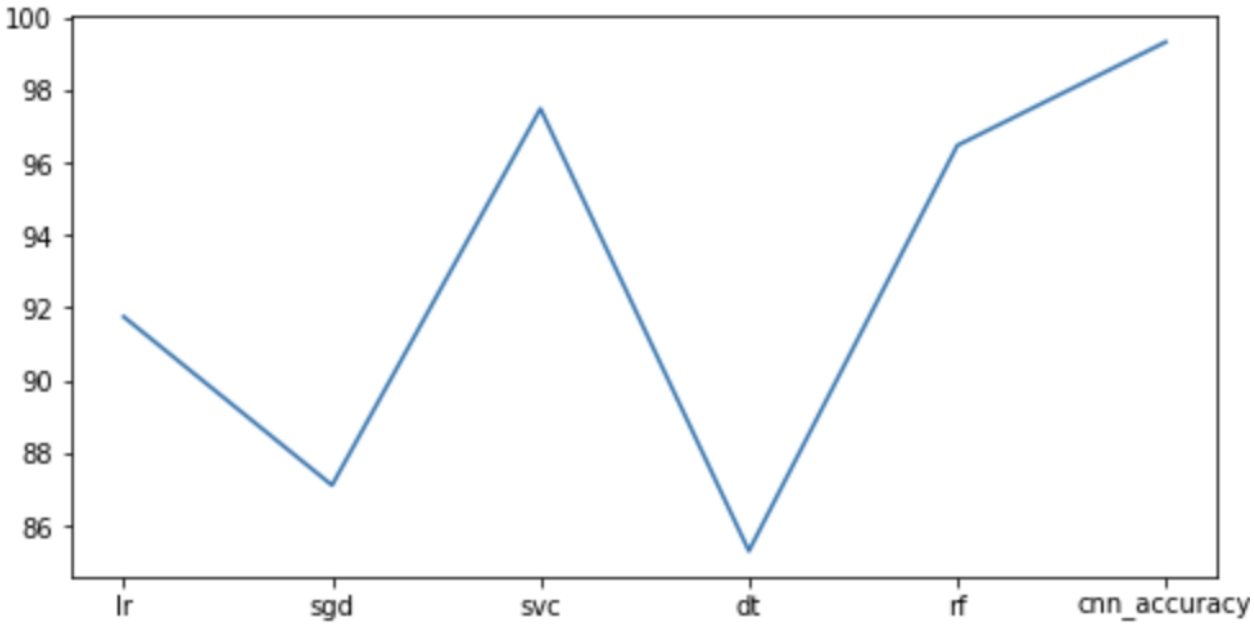
==========

Total params: 6,983,242
Trainable params: 6,982,090
Non-trainable params: 1,152

An accuracy of **99.357%** was achieved on the validation data.

Results

Comparison of Accuracies of all the models on the validation dataset:



Comparison of different performance metrics on validation data:

Model	**Accuracy**
Logistic Regression	91.75 %
Stochastic Gradient Descent	87.58 %
Support Vector Machine	97.47 %
Decision tree	85.14 %
Random Forest	96.46 %
CNN	99.357%

Conclusion

By considering all the models, the best accuracy is obtained with the Convolutional Neural Network. Therefore, it is the best predictor in this case.

What have we learned from this project?

Laying more emphasis on learning the importance of the below, in addition to the content mentioned in the 'Methods used' section.

It is crucial to know the importance of hyperparameters to tune:

Number of neurons, activation functions, optimizer, learning rate, batch size and epochs.

- Tensor is an object which helps us to understand the following:
  If there is a vector space, tensor then helps to explain the multiple linear relations between algebraic units in such a space.
- When a sequential is right to use - when each layer has one input & one output. First instance of the class 'Sequential' is created and other model layers are added.
- Convolution is useful in extracting certain features from input images. Kernel is a matrix. It is moved across the image, the values are multiplied with input, so that the output is enhanced.
- Kernel initialization is for assigning initial weights to the model, based on statistics.It is used to generate the weights & for distributing them. In our model, we have used 'he_normal'which picks values from normal distribution, with mean = 0, standard deviation = square-root(2 / no. of inputs units in weight tensor object)
- Padding - assumes either "same" or "valid" values for it. If the value is "same", both input and output sizes match and if the value is "valid" input values are allowed to reduce as per the convolution applied.
- Loss function - this is used to find the difference between expected and the current values of the model built.
- Batch normalization - normalize input values by scaling them. This makes the neural networks fast, efficient and reliable. This can be applied to the inputs of the current layer directly or can also be applied to activation functions of prior layers. This is helpful in reducing the number of epochs.
- Hyper parameter - The values of this hyper parameter are helpful in having a control over the learning process of the neural network.
- Derived are the values of other parameters, for example: weights.
- Epochs - number of times the model iterates through the training data.
- Batch size - number of training samples a model should iterate over prior to updating weights or values.

References:

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

https://scikit-learn.org/stable/modules/sgd.html

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

https://scikit-learn.org/stable/modules/neural_networks_supervised.html

https://www.analyticsvidhya.com/blog/2021/06/how-to-load-kaggle-datasets-directly-into-google-colab/
https://www.analyticsvidhya.com/blog/2018/12/guide-convolutional-neural-network-cnn/

https://www.geeksforgeeks.org/keras-conv2d-class/#:~:text=Keras%20Conv2D%20is%20a%202D,produce%20a%20tensor%20of%20outputs.

https://machinelearningmastery.com/

https://www.analyticsvidhya.com/blog/2021/05/tuning-the-hyperparameters-and-layers-of-neural-network-deep-learning/#:~:text=The%20hyperparameters%20to%20tune%20are,layers%20can%20affect%20the%20accuracy.