Automated data labeling system with active learning

04-03-2022

Prince Kumar

Contents

Problem definition

For any supervised learning system to perform well, it requires hundreds and thousands of labeled datasets. Sometimes these labeled datasets come at little or at no cost but many a time,labeled instances are very difficult,time consuming and expensive to obtain.Here are few examples:

- Speech recognition.: Accurate labeling of speech utterances is extremely time consuming and requires trained linguists.
- Classification and filtering : classifying and filtering hundreds of media contents like audio,video images texts to find which one is relevant is tedious and time consuming business

Active learning for automated data labeling systems will attempt to solve this bottleneck problem for supervised learning systems where data may be abundant and labels are scarce or expensive to obtain,thereby minimizing the costs and time to acquire the data

Literature Survey:

Most of the current research in active learning involves the best method to choose a subset of the dataset that is to be labeled.

Scenarios:

- **Membership Query Synthesis :** In this setting, the learner can request labels for any unlabeled instance in the input space, the learner can generate instances from the natural distribution and can query the labels.
- **Pool-Based Sampling :** Here instances are drawn from the large pool of the data and the learner query only those instances for which it is least confident.
- **Stream-Based Selective Sampling :** here all the data points are examined and the learner decides for each datapoint whether to assign a label or query the teacher.

Query Strategies :

All active learning scenarios involve evaluating the informativeness of unlabeled instances, which can either be generated or sampled from a given distribution.some of the common strategies are as**:**

- **Uncertainty Sampling :** in this strategy, an active learner queries the instances about which it is least certain how to label.
- **Query-By-Committee :** different models are trained on the labeled data, and vote on the output for unlabeled data; label those points for which the "committee" or models disagrees the most
- **Expected Model Change :** uses a decision-theoretic approach, selecting the instance that would impart the greatest change to the current model if we knew its label.
- **Variance reduction**: label those points that would minimize output variance, which is one of the components of error.

Approach and Key metric (KPI)

- Hand label a small portion of the data set.
- Using our labeled data, train a classifier that returns classification probabilities.
- Use the trained classifier to predict class probabilities for all unlabeled data.
- Assume that all classifications are accurate if they have a probability above the set threshold
- Hand label some portion of the unlabeled data with the lowest predicted class probabilities and add to the training set
- Repeat steps 2–5 until a small portion of unlabeled data remains.
- Hand label the remaining data

For validating the approach , we will be using **Accuracy.**

One challenge with using Accuracy is obviously it fails when we have an imbalanced dataset.

Real world challenges and constraints:

Noisy images can also hamper the efficiency of the entire active learning system.
Finding the best classifier for this task is also a time consuming task.
Evaluating the efficiency of the entire Active learning System is always a real world constraint.

Data Collection:

For this project I am using A-Z handwritten datasets obtained from kaggle.
https://www.kaggle.com/sachinpatel21/az-handwritten-alphabets-in-csv-format
The dataset contains 26 folders (A-Z) containing handwritten images in size 28*28* pixels, each alphabet in the image is center fitted to a 2020 pixel box
Each image is stored as Gray-level.

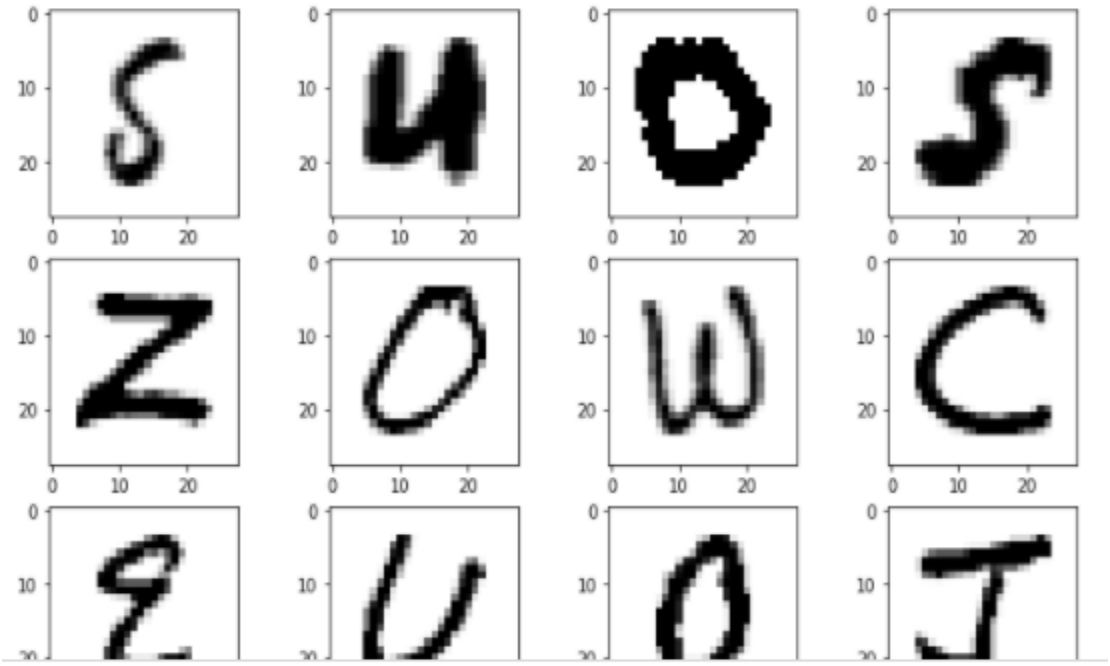
I have written a python script using cv2 library to convert a .CSV file obtained from this source to actual images in .png format in a structured folder.

Total number of images in this dataset is close to 3 lakh.

EDA and Feature Extraction

Reading the random images from the datasets using sklearn shuffle

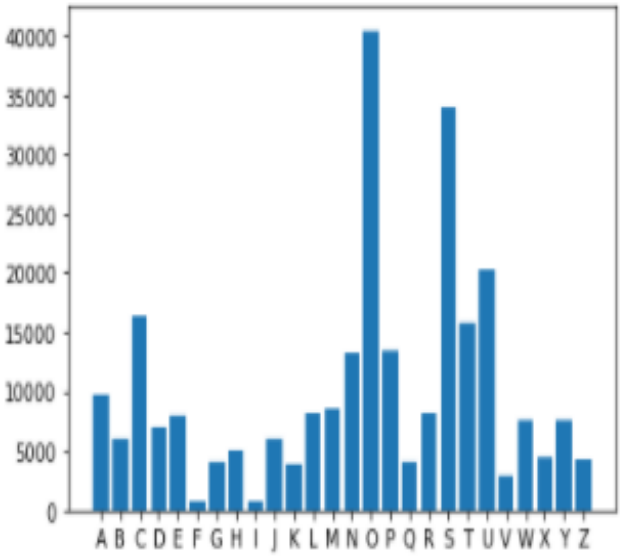
```
from sklearn.utils import shuffle

X_shuffle = shuffle(X)

plt.figure(figsize = (12,10))
row, colums = 4, 4
for i in range(16):
    plt.subplot(colums, row, i+1)
    plt.imshow(X_shuffle.iloc[i].values.reshape(28,28),interpolation='nearest', cmap='Grey
plt.show()
```



Checking the distribution of the labels

```
labeldf=pd.DataFrame({'label':labelarr})
plt.bar(labeldf.groupby("label")["label"].count().index, labeldf.groupby("label")["label"].count())
```

```
<BarContainer object of 26 artists>
```



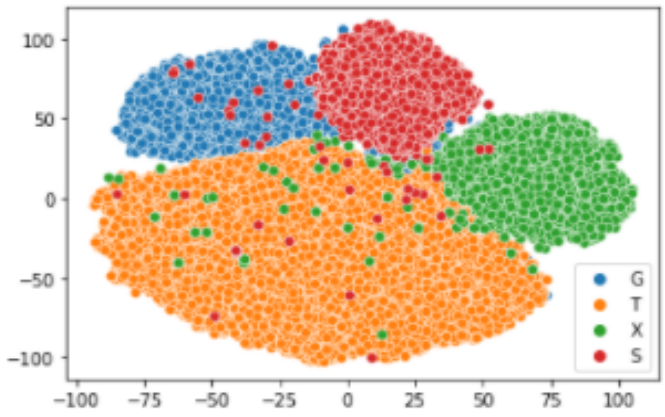
We observed that classes are imbalance

Performing dimension reduction on a small subset of the data using T SNE technique to see if we can plot them on a lower dimension

```
import numpy as np
from sklearn.manifold import TSNE
tsne=TSNE(n_components=2,learning_rate='auto',init='random',perplexity=30)
X_embedded=tsne.fit_transform(np.array(imgarr[0:40000]))
X_embedded.shape
```

```
import warnings
warnings.filterwarnings('ignore')
sb.scatterplot(X_embedded[0:40000,0],X_embedded[0:40000,1],hue=labelarr[0:40000])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f89c4a5be90>
```



We can easily see that that almost all the labels in the subset are clustered together

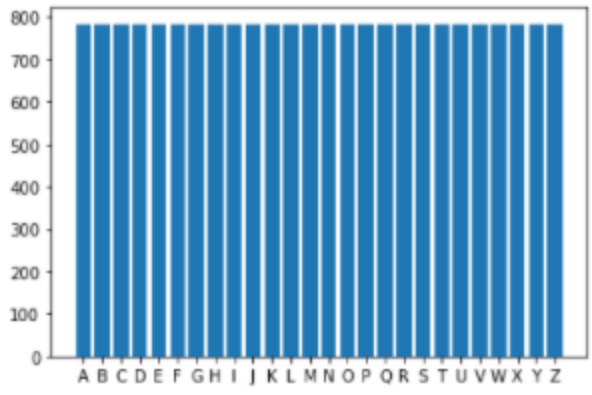
Since our dataset is imbalanced , I am performing undersampling on the train dataset which is 70% of the entire dataset to balance it.

```
#performing the undersampling for balancing the classes
from imblearn.under_sampling import RandomUnderSampler
from collections import Counter
under_sampler = RandomUnderSampler(random_state=42)
X_res, y_res = under_sampler.fit_resample(imgarr, labelarr)
print(f"Training target statistics: {Counter(y_res)}")
```

```
Training target statistics: Counter({'A': 784, 'B': 784, 'C': 784, 'D': 784, 'E': 784
```

```
#checking for the result
plt.bar(Counter(y_res).keys(),Counter(y_res).values())
```

```
<BarContainer object of 26 artists>
```

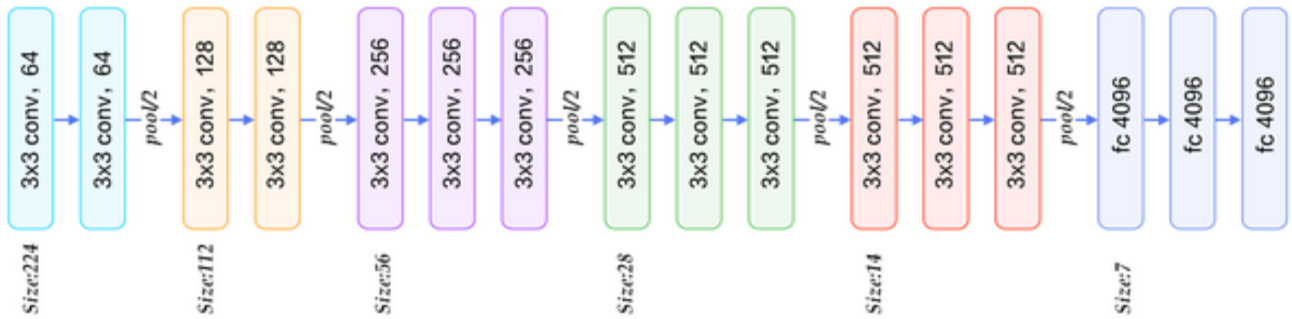


We can see that we have now balanced all the classes in the training set which we will be using to train the classifier

Modeling and Error Analysis:

For the classification task , I am using deep learning neural networks with transfer learning as deep neural networks tend to perform well on image datasets compared to classical machine learning approaches like random forest and svms.

Transfer learning can ease the training time and efforts, vgg16 convolution layers are extracted from the vgg16 trained models on imagenet dataset present in keras.



VGG16, a simple and widely used Convolutional Neural Network (CNN) Architecture, uses Image classification. It has 16 layers.The VGG16 Architecture was developed and introduced by Karen Simonyan and Andrew Zisserman from the University of Oxford. Its convolution layers consists of 3x3 filter with a stride 1 and always use the same padding and maxpool layer of 2x2 filter of stride 2.at the end it has 2 FC(fully connected layers) followed by a softmax for output.

```
vgg16_model = vgg16.VGG16(pooling='avg', weights='imagenet', include_top=False, input_shape=(32,32,3))
for layers in vgg16_model.layers:
            layers.trainable=False
last_output = vgg16_model.layers[-1].output
vgg_x = Flatten()(last_output)
vgg_x = Dense(128, activation = 'relu')(vgg_x)
vgg_x = Dense(26, activation = 'softmax')(vgg_x)
vgg16_final_model = Model(vgg16_model.input, vgg_x)
vgg16_final_model.compile(loss = 'categorical_crossentropy', optimizer= 'adam', metrics=['acc'])
vgg16_final_model.fit(train_X,train_Y_one_hot,epochs=10,shuffle=True)
predicted_prob=vgg16_final_model.predict(test_X)
acc=accuracy_metric(predicted_prob,test_Y_one_hot)
prf_metric(predicted_prob,test_Y_one_hot)
```

I am adding two densely connected layers with 128 and 26 perceptrons in each layer with relu and softmax activation units.

Training the Model:

10 % of the training sets is chosen using stratified sampling methods which will be used to train in the first iteration,each iteration of the active learning process contains 10 epochs and the iteration continues percentage of classification with confidence score less than 0.6 is less than 10.Samples with low confidence score is added to the training set and iteration continues.

```
def build_model(train_X,train_Y_one_hot,test_X,test_Y_one_hot):
  vgg16_model = vgg16.VGG16(pooling='avg', weights='imagenet',
include_top=False, input_shape=(32,32,3))
  for layers in vgg16_model.layers:
              layers.trainable=False
  last_output = vgg16_model.layers[-1].output
  vgg_x = Flatten()(last_output)
  vgg_x = Dense(128, activation = 'relu')(vgg_x)
  vgg_x = Dense(26, activation = 'softmax')(vgg_x)
  vgg16_final_model = Model(vgg16_model.input, vgg_x)
```

```
  vgg16_final_model.compile(loss = 'categorical_crossentropy', optimizer=
'adam', metrics=['acc'])
  vgg16_final_model.fit(train_X,train_Y_one_hot,epochs=10,shuffle=True)
  predicted_prob=vgg16_final_model.predict(test_X)
  acc=accuracy_metric(predicted_prob,test_Y_one_hot)
  prf_metric(predicted_prob,test_Y_one_hot)

  count=0
  for i,j in enumerate(predicted_prob):
    if(np.max(j)<0.6):
      try:
        train_X=np.append(train_X,np.array([test_X[i]]),axis=0)

train_Y_one_hot=np.append(train_Y_one_hot,np.array([test_Y_one_hot[i]]),ax
is=0)
        test_X=np.delete(test_X,i,axis=0)
        test_Y_one_hot=np.delete(test_Y_one_hot,i,axis=0)
        count +=1
      except:
        pass
  thres=count/len(predicted_prob)
  print("Percentage of classification with confidence score less than 0.6
is {}".format(thres))

  if(thres>0.10):#build the model again
    build_model(train_X,train_Y_one_hot,test_X,test_Y_one_hot)
  else:
    model_structure=vgg16_final_model.to_json()
    f=Path("model_structure.json")
    f.write_text(model_structure)
    vgg16_final_model.save_weights("model-detection.h5")

  return "Accuracy is {}".format(acc)
```

Training epochs

```
Epoch 1/10
114/114 [==============================] - 38s 326ms/step - loss: 3.3551 - acc: 0.4396
Epoch 2/10
114/114 [==============================] - 36s 320ms/step - loss: 0.9424 - acc: 0.7319
Epoch 3/10
114/114 [==============================] - 36s 314ms/step - loss: 0.6316 - acc: 0.8126
Epoch 4/10
114/114 [==============================] - 36s 314ms/step - loss: 0.4513 - acc: 0.8604
Epoch 5/10
114/114 [==============================] - 36s 313ms/step - loss: 0.3620 - acc: 0.8918
Epoch 6/10
114/114 [==============================] - 36s 314ms/step - loss: 0.2882 - acc: 0.9096
Epoch 7/10
114/114 [==============================] - 38s 334ms/step - loss: 0.2525 - acc: 0.9181
Epoch 8/10
114/114 [==============================] - 37s 320ms/step - loss: 0.2071 - acc: 0.9283
Epoch 9/10
114/114 [==============================] - 36s 314ms/step - loss: 0.1865 - acc: 0.9371
Epoch 10/10
114/114 [==============================] - 36s 314ms/step - loss: 0.1585 - acc: 0.9522
Avg precision :0.8431917963333797 Avg recall : 0.8346546310832025 Avg fcore: 0.8356868532325372
Percentage of false classification with confidence score less than 0.6 is 0.09211145996860283
'Accuracy is 83.46546310832025'
```

We can see at the end of the 10 epochs in the first iteration , we have got Accuracy of 83.46

Validation on Test Data

We will now validate the model on test data to check it has not overfitted on the test data.

```
#reading the testdata
imgarr=[]
labelarr=[]
for i in os.listdir('/content/test/'):
  if i != '.ipynb_checkpoints':#ignoring checkpoints if present in the same directory
    for j in os.listdir(os.path.join('/content/test/',i)):
      try:
        arr=cv2.imread('/content/test/'+i+'/'+j,cv2.IMREAD_GRAYSCALE)
        imgarr.append(arr.ravel())
        labelarr.append(i[0])
      except:
        continue
#defining key value pair for label encoding
key_val = {0:'A',1:'B',2:'C',3:'D',4:'E',5:'F',6:'G',7:'H',8:'I',9:'J',10:'K',11:'L',12:'M',13:'N',14:'O',15:'P',16:'Q',17:'R',18:'S',19:'T',2
label_encoder=dict()
X_test=np.array(imgarr)
y_test=np.array(labelarr)
for i in key_val.keys():
    label_encoder[key_val[i]]=i
y_test=np.array([label_encoder[i] for i in y_test])
test_Y_one_hot = to_categorical(y_test)
X_test=np.dstack([X_test]*3)
X_test= X_test.reshape (-1,28,28,3)
test_X = np.asarray([img_to_array(array_to_img(im, scale=False).resize((32,32))) for im in X_test])
```

```
predicted_prob=loaded_model.predict(test_X)
accuracy_metric(predicted_prob,test_Y_one_hot)
```

```
82.99637535239629
```

Since we are getting almost the same accuracy on test data which has the same distribution as the entire population , we can conclude that the model is not overfitted. Models can also be improved by doing some hyperparameter tuning in the next stage

Saving the Model:

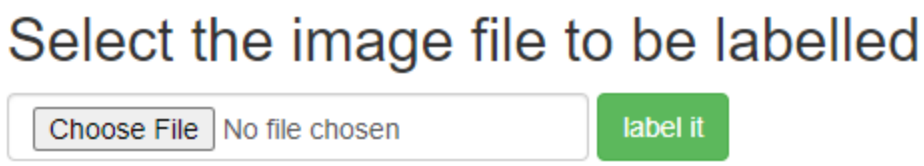
We will be saving the model architecture and trained weight in json and h5 format which will be used in production deployment using Flask

```
import tensorflow as tf
```

```
json_file = open('model_structure.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = tf.keras.models.model_from_json(loaded_model_json)
# load weights into new model
loaded_model.load_weights("model-detection.h5")
```

Deployment and Productionisation:

We will be using Flask to deploy the the final model as application which will be used as a data labeling tool.User will be provided with interface to select the image to be labelled.Results with high confidence score is moved to their respective folder and and results with low confidence score will be prompted with the input box to label the image manually,it is then moved to the respective folder based on user inputs



Prediction is: C with confidence:99.98%
Label More

References:

- Burr Settles. Active Learning Literature Survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison. 2009.
- http://burrsettles.com/pub/settles.activelearning.pdf
- https://en.wikipedia.org/wiki/Active_learning_(machine_learning)
- https://www.v7labs.com/blog/data-labeling-guide#data-labeling-approaches.
- https://medium.com/mindboard/active-learning-for-fast-data-set-labeling-890d4080d750
- https://www.nature.com/articles/s41598-018-24876-0