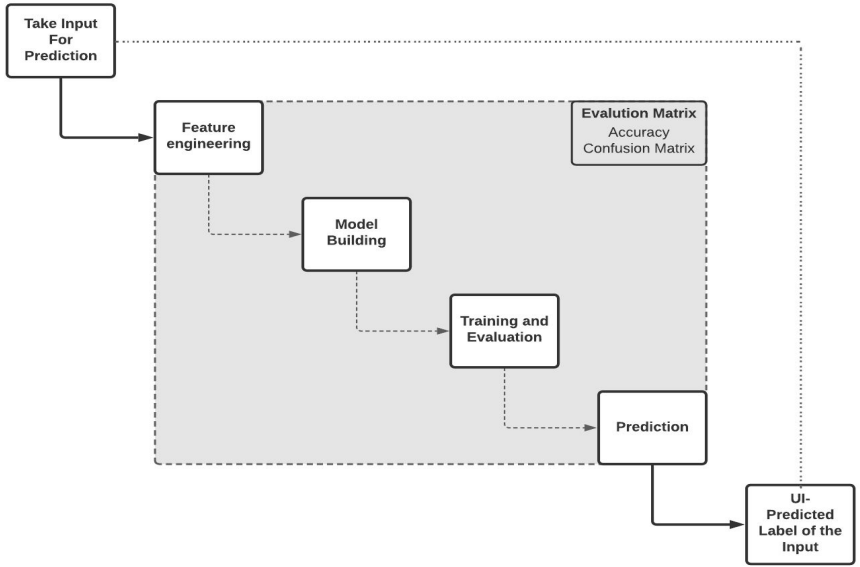
Letter Prediction

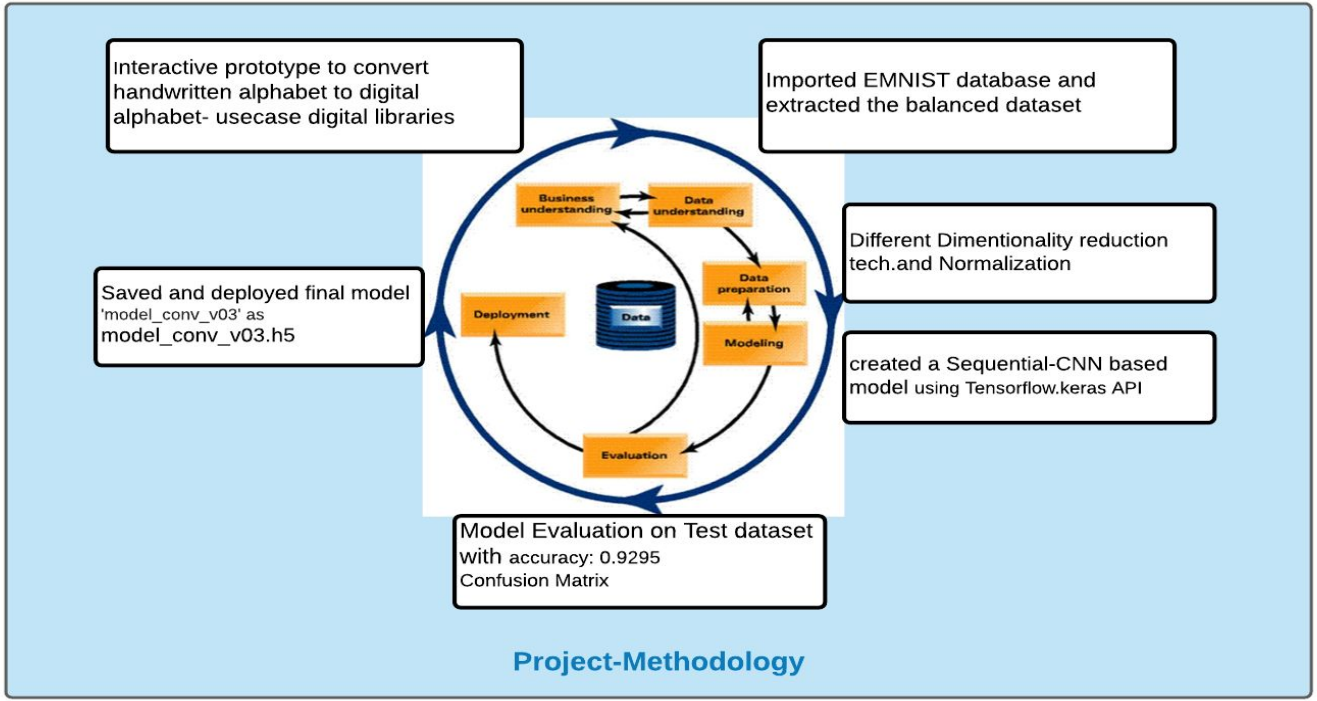
Cory Randolph, Abraham Kong,
Akanksha Rawat, Karishma Kuria

Abstract

- Build an whiteboard on a front end UI
- Take user drawn had written letters
- Apply Machine Learning Model to classify drawing
- Return a prediction based on the user's drawn letter



Methodology



Data Understanding

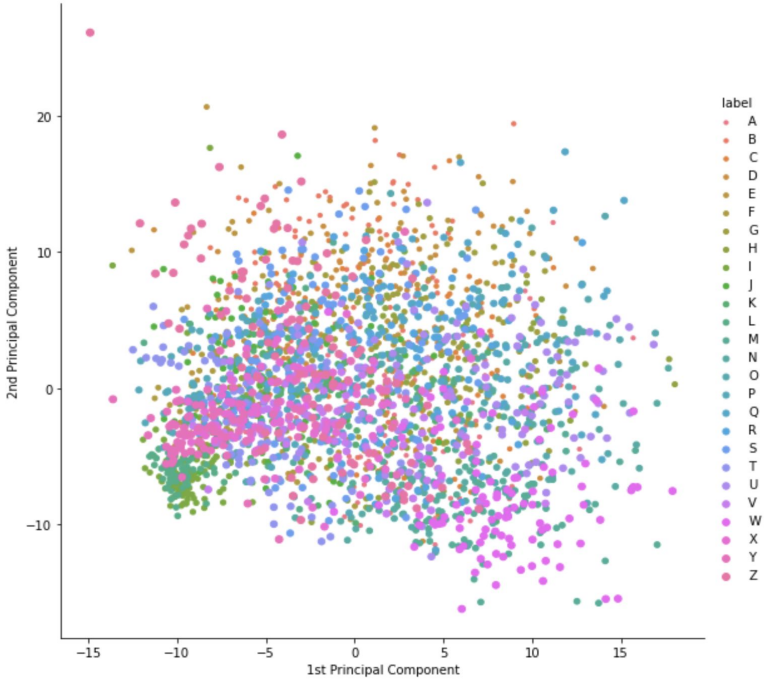
- Imported EMNIST- balanced dataset.
- The EMNIST dataset has a set of handwritten digits (0–9), (a-z), and (A-Z) being converted into 28x28 pixel pictures.
- The EMNIST Balanced dataset - 47-class dataset was chosen over the By Class dataset to avoid classification errors.

Data Preprocessing

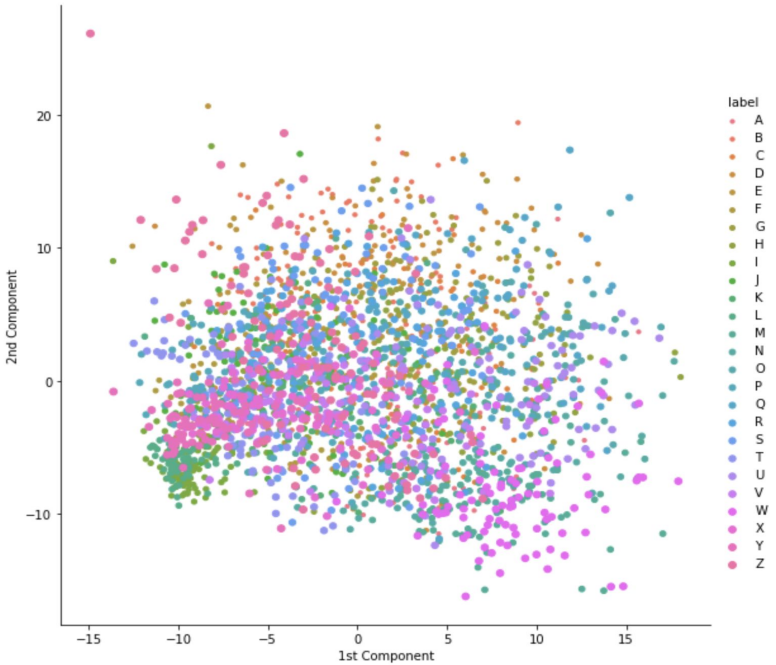
1. **Data Reduction Techniques used**:
    - ➢ Principal Component Analysis (PCA)
    - ➢ Singular Value Decomposition (SVD)
    - ➢ Locally Linear Embedding (LLE)
    - ➢ T-distributed Stochastic Neighbor Embedding (T-SNE)
    - ➢ Isometric Mapping(ISOMAP)
    - ➢ Uniform Manifold Approximation and Projection (UMAP)
2. **Data Normalization:** The data set is divided by 255 based on the RGB codes.

Data Reduction Visualizations

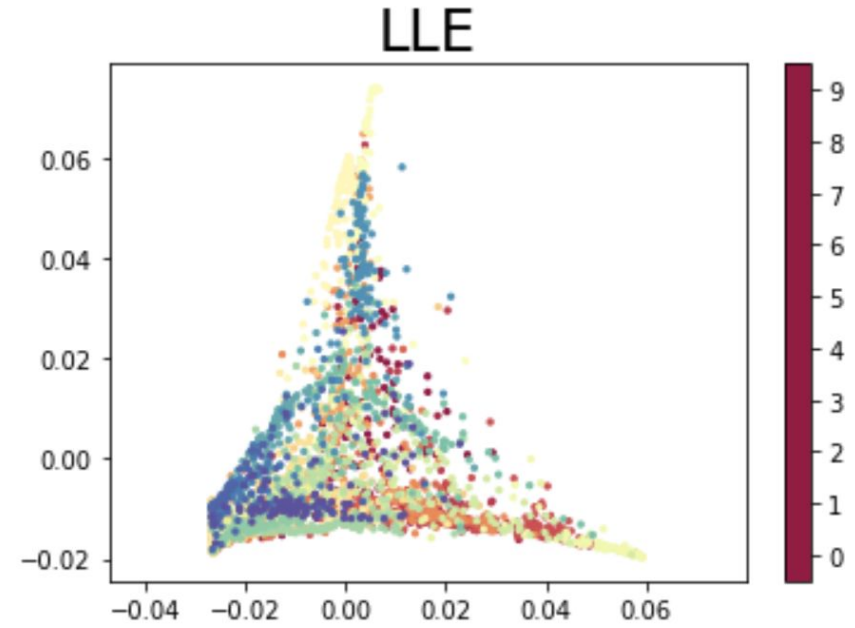
PCA

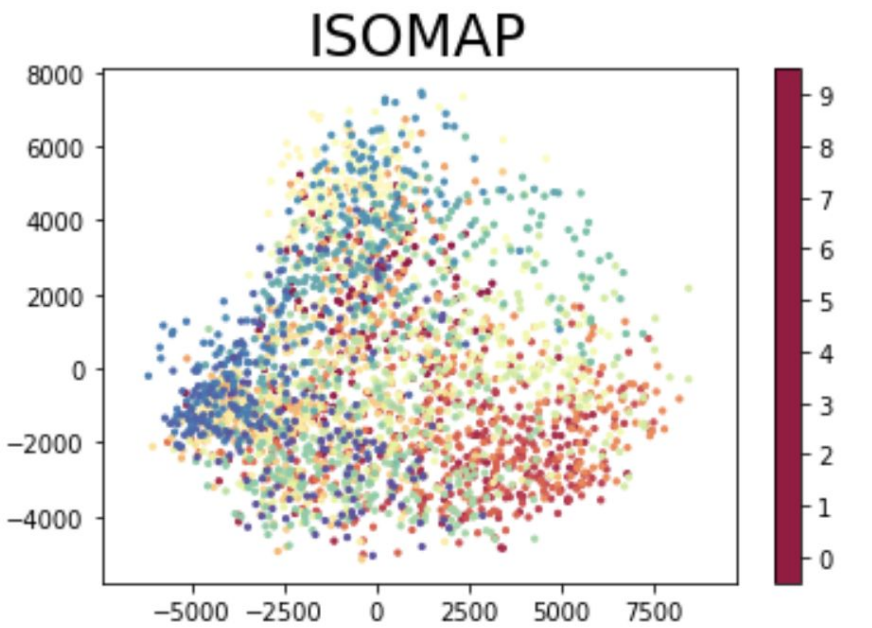


SVD

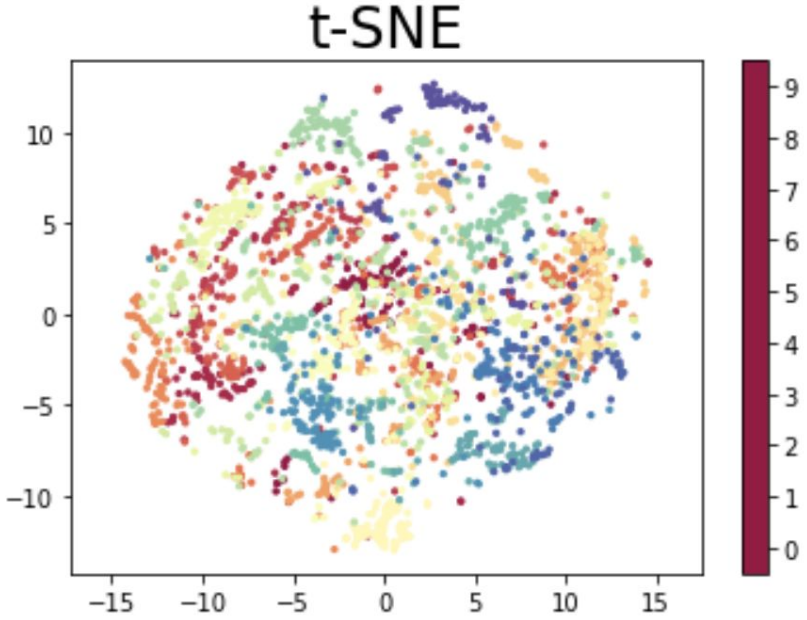


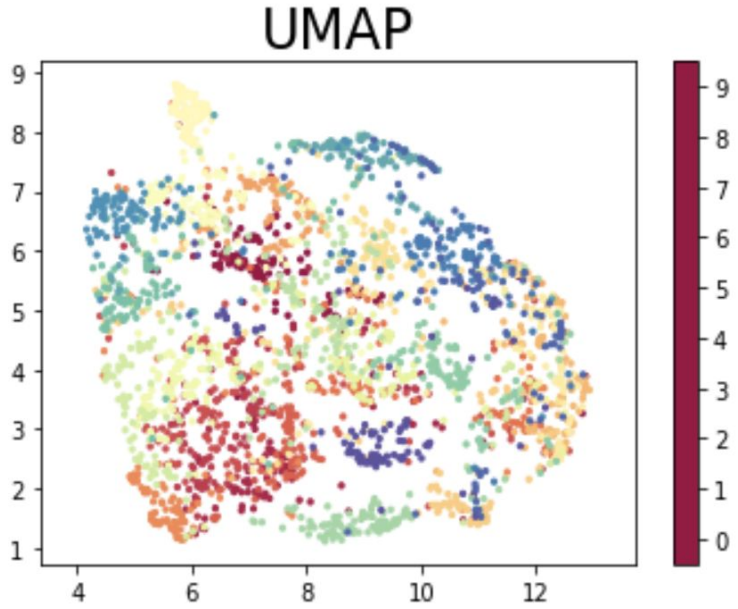
Data Reduction Visualizations





Data Reduction Visualizations





Modeling

1. **Building the model-** Using Keras- high level API of tensorflow framework build a layered sequential type Model.- Based on **Convolutional Neural Network architecture.** model type= Sequential()

```
model.add(Conv2D(32,kernel_size=3,activation='relu',input_shape=(28,28,1)))
```

2. **Compiling the model- parameters [optimizer, Loss, Metrics]**

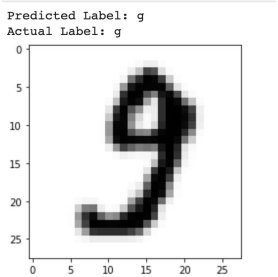
```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
model = Sequential()

model.add(Conv2D(32,kernel_size=3,activation='relu',input_shape=(28,28,1)))
model.add(BatchNormalization())
model.add(Conv2D(32,kernel_size=3,activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(32,kernel_size=5,strides=2,padding='same',activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.4))
```

3. **Training the Model**

```
history = model.fit(x = x_train,
                    y = y_train,
                    validation_data = (x_test, y_test),
                    epochs=20) #10
```



4. **Evaluation and Prediction**

model.evaluate(x_test, y_test)

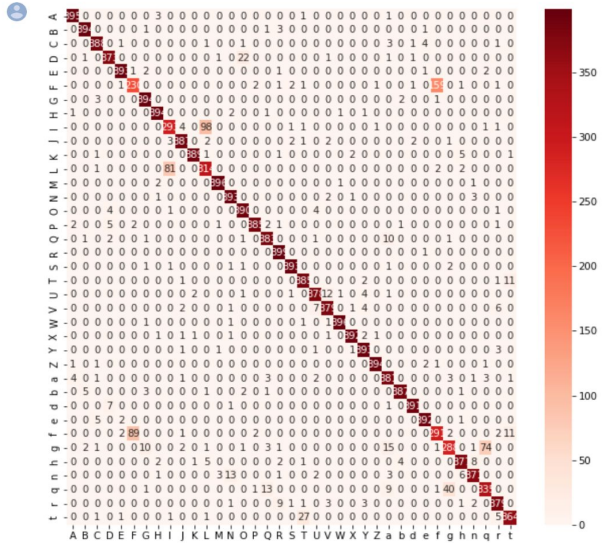
Evaluation

Evaluated on test data set. Using below measures:

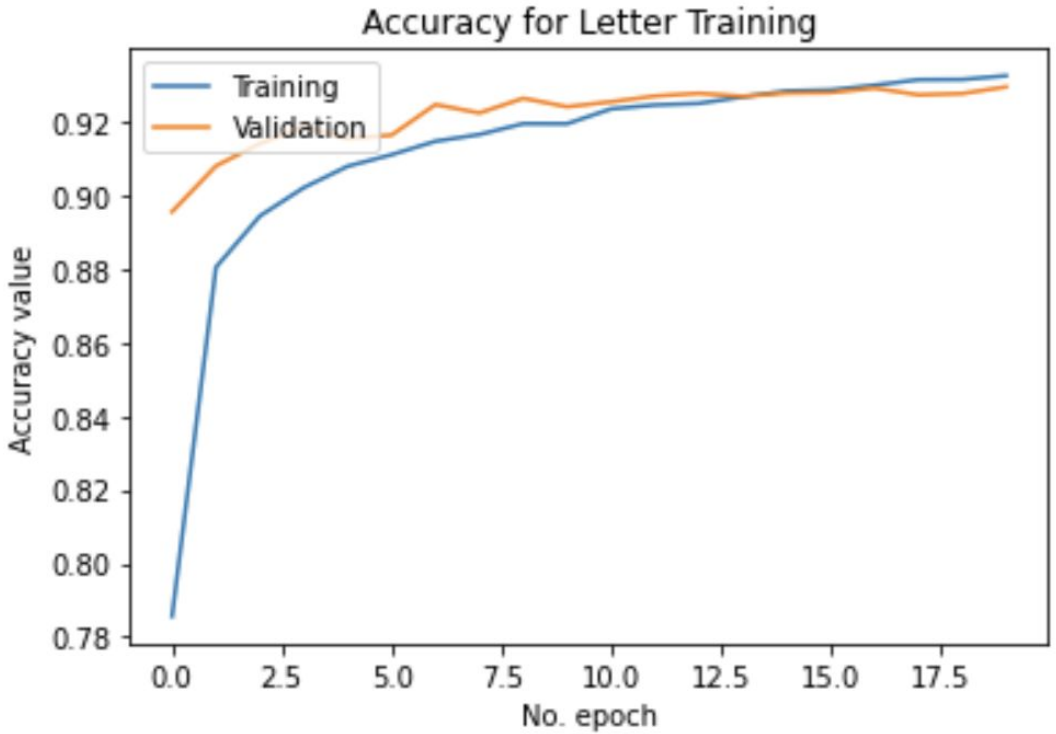
- Confusion Matrix
- Accuracy
- Loss
- Precision
- Recall
- f1-score

```
[ ] cm = confusion_matrix(y_test_as_labels, y_test_preds_as_labels)
```

```
plt.figure(figsize=[10,10])
sns.heatmap(cm, cmap="Reds", annot=True, fmt='.0f', xticklabels = labels_dict_caps.values(), yticklabels= labels_dict_caps.values())
plt.show()
```

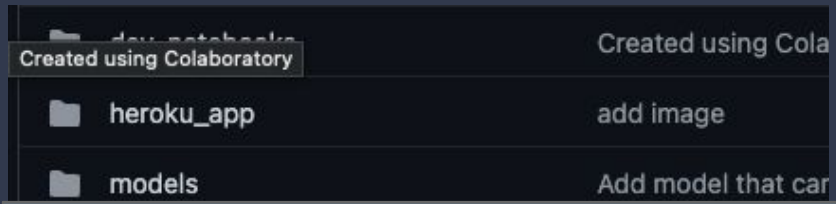


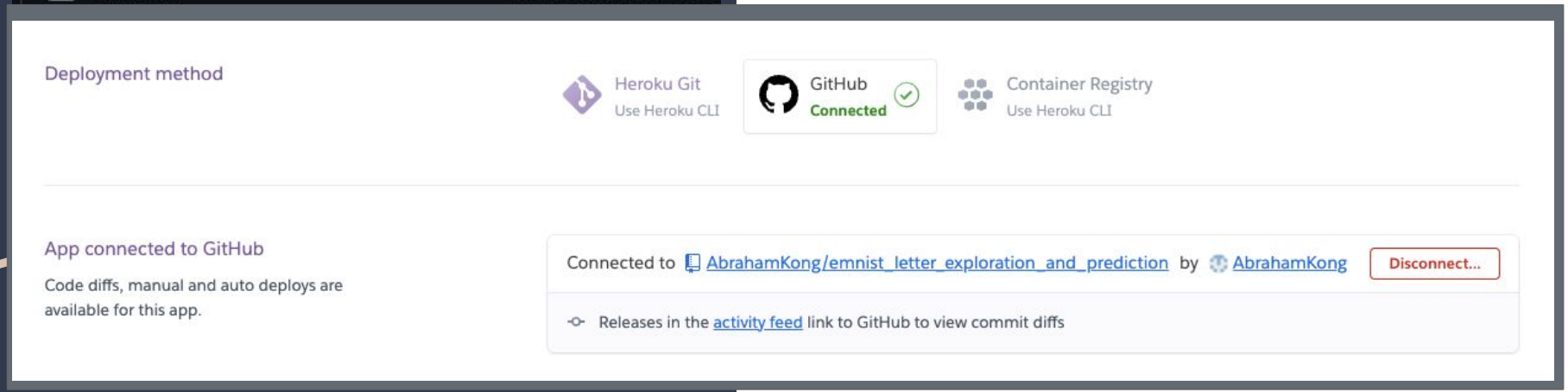
Evaluation


Accuracy for Letter Training
Training
Validation
Accuracy value
0.92
0.90
0.88
0.86
0.84
0.82
0.80
0.78
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
No. epoch

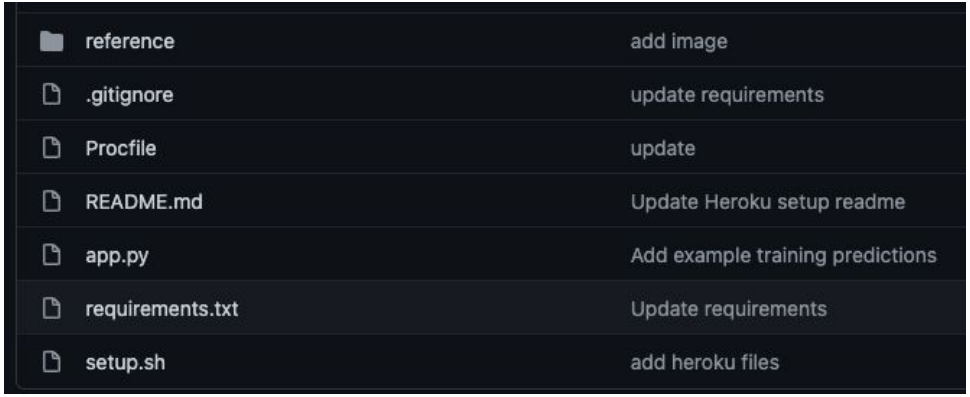
Deployment: Heroku

We choose Heroku as our web server
To deploy our app, as it is very easy to
Deploy once the Github is set right.





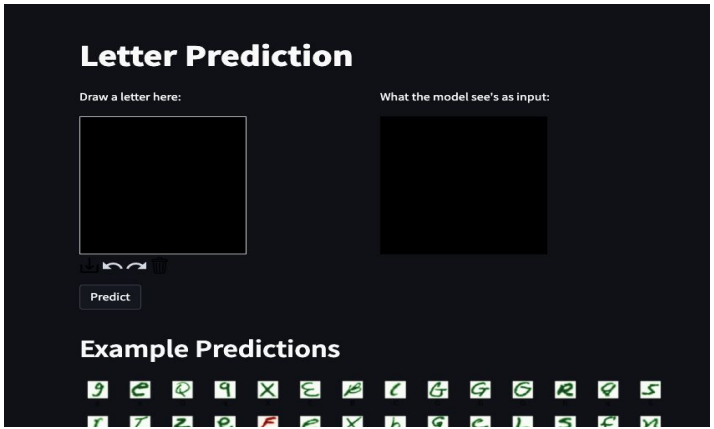
Deployment: Heroku



- app.py generated by our Google Colab file Including our Streamlit Front end and modeling
- Requirements.txt and setup.sh to set up the environment
- Procfile to tell Heroku to run the file

Deployment Link:
https://letter-prediction.herokuapp.com/

Will introduce more in the live demo part coming up



Front End Design (Streamlit)

Main Features

- User draws a letter
- Custom ML model makes a class/letter prediction
- Probability of labels displayed

```
with col1:
  st.markdown('Draw a letter here:')
  # Create a drawing canvas with desired properties
  canvas_result = st_canvas(
      fill_color="#ffffff",
      stroke_width=10,
      stroke_color='#ffffff',
      background_color="#000000",
      height=200,
      width=200,
      drawing_mode='freedraw',
      key="canvas",
  )

with col2:
  # Show that the resized image looks like
  st.markdown("What the model see's as input:")
  if canvas_result.image_data is not None:
      img = cv2.resize(canvas_result.image_data.astype('uint8'), (28, 28))
      img_rescaling = cv2.resize(img, (200, 200), interpolation=cv2.INTER_NEARES
      st.image(img_rescaling)

# Generate the prediction based on the users drawings
if st.button('Predict'):
    x_user_input = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    x_user_input = x_user_input.reshape(1, 28, 28, 1) / 255 # Reshape and norma
    pred = model.predict(x_user_input)
    pred_label = labels_dict[pred.argmax()]
    st.header(f'Predicted Label: {pred_label}')

    # Create a Plotly barchart of the predicted probabilities
    fig = create_probability_fig(pred)
    st.plotly_chart(fig, use_container_width = True)
```

Live Demo

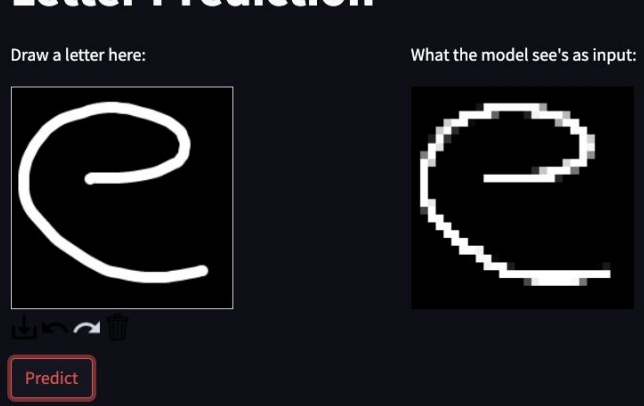
Heroku Hosted Demo:

https://letter-prediction.herokuapp.com/

Colab Demo:

https://github.com/coryroyce/emnist_letter_exploration_and_prediction/blob/main/streamlit_application/Streamlit_App.ipynb

* Can be run even if the demo app is no longer being actively hosted



Thank you!

- Cory Randolph, Abraham Kong, Akanksha Rawat, Karishma Kuria