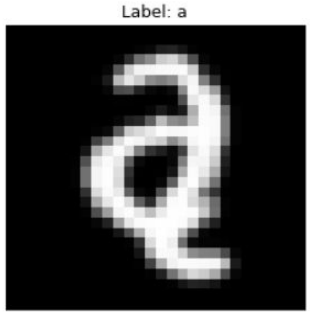
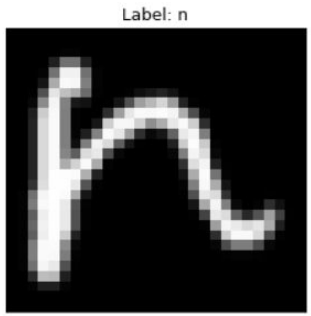
Handwritten Letter Prediction

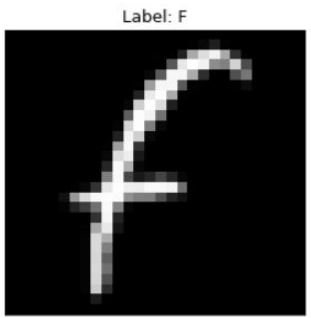
3/1/2022
Cory Randolph

Data Overview

- 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.
- This project disregarded the numerical (0-9) classes to focus just on letters







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]**

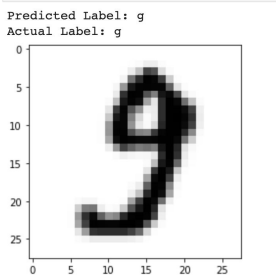
```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
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)

Model Architecture

```
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))

model.add(Conv2D(64,kernel_size=3,activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(64,kernel_size=3,activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(64,kernel_size=5,strides=2,padding='same',activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.4))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.4))
model.add(Dense(37, activation='softmax')) #26 fro just Capital Letters
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
batch_normalization (BatchNormalization)	(None, 26, 26, 32)	128
conv2d_1 (Conv2D)	(None, 24, 24, 32)	9248
batch_normalization_1 (BatchNormalization)	(None, 24, 24, 32)	128
conv2d_2 (Conv2D)	(None, 12, 12, 32)	25632
batch_normalization_2 (BatchNormalization)	(None, 12, 12, 32)	128
dropout (Dropout)	(None, 12, 12, 32)	0
conv2d_3 (Conv2D)	(None, 10, 10, 64)	18496
batch_normalization_3 (BatchNormalization)	(None, 10, 10, 64)	256
conv2d_4 (Conv2D)	(None, 8, 8, 64)	36928
batch_normalization_4 (BatchNormalization)	(None, 8, 8, 64)	256
conv2d_5 (Conv2D)	(None, 4, 4, 64)	102464
batch_normalization_5 (BatchNormalization)	(None, 4, 4, 64)	256
dropout_1 (Dropout)	(None, 4, 4, 64)	0
flatten (Flatten)	(None, 1024)	0
dense (Dense)	(None, 128)	131200
batch_normalization_6 (BatchNormalization)	(None, 128)	512
dropout_2 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 37)	4773

Total params: 330,725
Trainable params: 329,893
Non-trainable params: 832

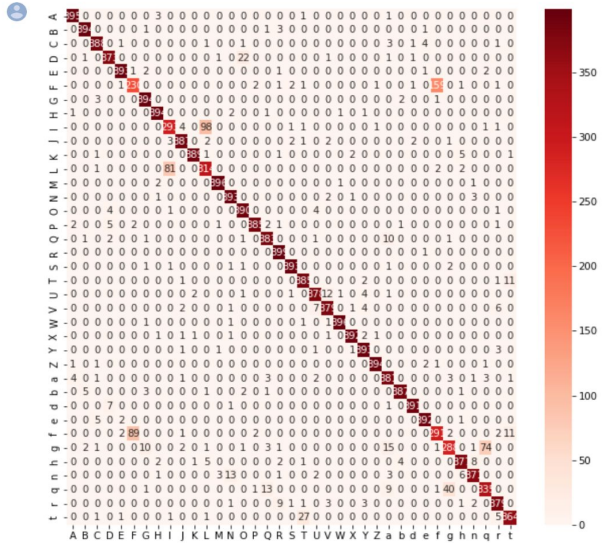
Evaluation

Evaluated on test data set. Using below measures:

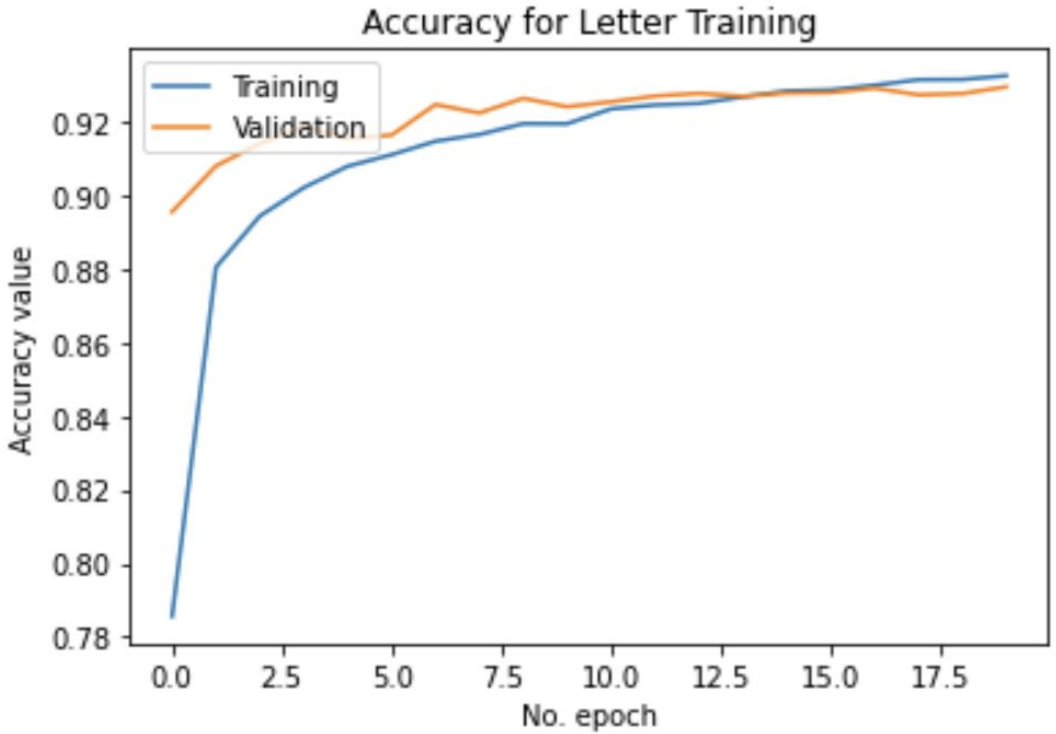
- Confusion Matrix
- Accuracy
- Loss
- Precision
- Recall
- f1-score

```
[ ] 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

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

Colab Demo:

Thank you!