Multi-Class Classifier ¶

The goal of this project is to build a multi-class classifier to recognise sign language

In [1]:
```
import csv
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from os import getcwd
```

In [2]:
```
# Function for reading the file passed
def get_data(filename):
    with open(filename) as training_file:
        reader = csv.reader(training_file, delimiter=',')
        imgs = []
        labels = []

        next(reader, None)

        for row in reader:
            label = row[0]
            data = row[1:]
            img = np.array(data).reshape((28, 28))

            imgs.append(img)
            labels.append(label)

        images = np.array(imgs).astype(float)
        labels = np.array(labels).astype(float)
    return images, labels

path_sign_mnist_train = f"{getcwd()}/../tmp2/sign_mnist_train.csv"
path_sign_mnist_test = f"{getcwd()}/../tmp2/sign_mnist_test.csv"
training_images, training_labels = get_data(path_sign_mnist_train)
testing_images, testing_labels = get_data(path_sign_mnist_test)

# checking dimensions
print(training_images.shape)
print(training_labels.shape)
print(testing_images.shape)
print(testing_labels.shape)
```

```
(27455, 28, 28)
(27455,)
(7172, 28, 28)
(7172,)
```

In [10]:
```
# Add dimension to the data
training_images = np.expand_dims(training_images, -1)
testing_images = np.expand_dims(testing_images, -1)

# Createing an ImageDataGenerator and doing Image Augmentation
train_datagen = ImageDataGenerator(
    rescale = 1./255,
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1
    )

validation_datagen = ImageDataGenerator(
    rescale = 1./255)

# Checking the new dimensions
#print(training_images.shape)
#print(testing_images.shape)
#print(training_labels)
#print(testing_labels)
```

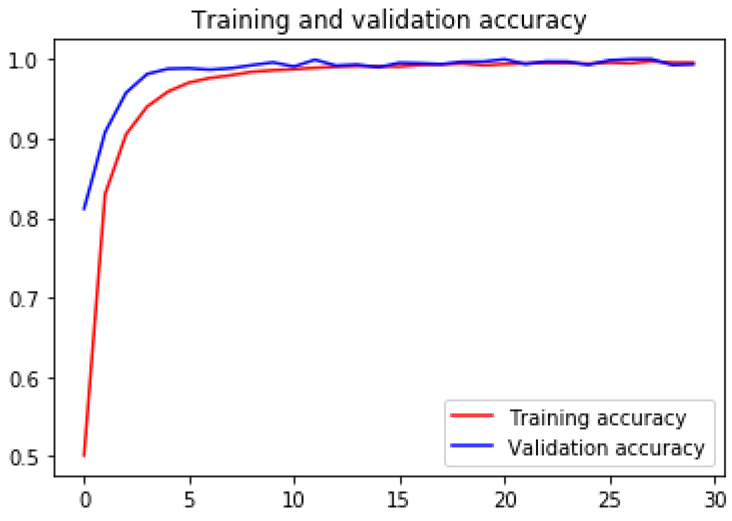
In [4]:
```
# ConvNet Model Definition
model = tf.keras.models.Sequential([
   # This is the first convolution
    tf.keras.layers.Conv2D(32, (3,3), activation='relu' ),
    tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # Flatten the results to feed into a DNN
    tf.keras.layers.Flatten(),
    #tf.keras.layers.Dropout(0.5),
    # 512 neuron hidden layer
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(26, activation='softmax')
])

# Compile Model.
model.compile(loss = 'sparse_categorical_crossentropy', optimizer='adam', metr
ics=['accuracy'])

# Train the Model
history = model.fit_generator(train_datagen.flow(training_images, training_lab
els, batch_size=64),
                              validation_data = validation_datagen.flow(testin
g_images, testing_labels, batch_size = 64),
                              epochs=30,verbose = 1)

model.evaluate(testing_images, testing_labels, verbose=0)
```

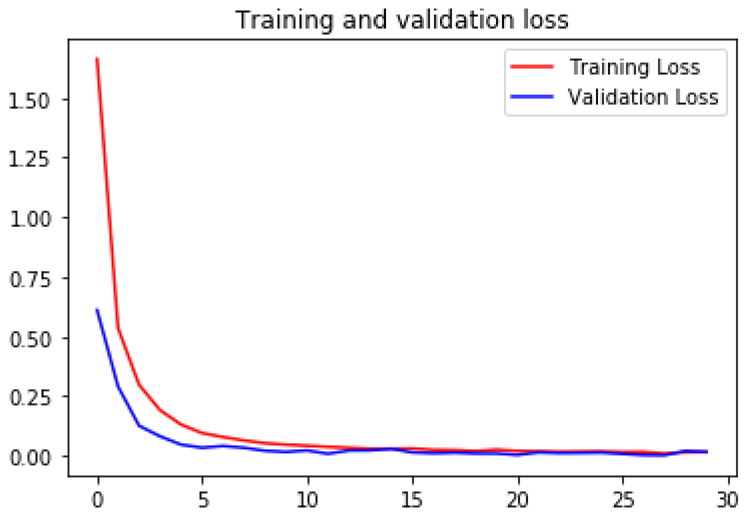
```
Epoch 1/30
429/429 [==============================] - 49s 114ms/step - loss: 1.6637 - ac
curacy: 0.2941 - val_loss: 0.6117 - val_accuracy: 0.8115
Epoch 2/30
429/429 [==============================] - 42s 98ms/step - loss: 0.5357 - acc
uracy: 0.7991 - val_loss: 0.2904 - val_accuracy: 0.9078
Epoch 3/30
429/429 [==============================] - 41s 96ms/step - loss: 0.2978 - acc
uracy: 0.8925 - val_loss: 0.1258 - val_accuracy: 0.9576
Epoch 4/30
429/429 [==============================] - 41s 95ms/step - loss: 0.1913 - acc
uracy: 0.9340 - val_loss: 0.0815 - val_accuracy: 0.9810
Epoch 5/30
429/429 [==============================] - 42s 99ms/step - loss: 0.1309 - acc
uracy: 0.9546 - val_loss: 0.0465 - val_accuracy: 0.9877
Epoch 6/30
429/429 [==============================] - 41s 96ms/step - loss: 0.0952 - acc
uracy: 0.9689 - val_loss: 0.0336 - val_accuracy: 0.9883
Epoch 7/30
429/429 [==============================] - 43s 100ms/step - loss: 0.0784 - ac
curacy: 0.9725 - val_loss: 0.0407 - val_accuracy: 0.9866
Epoch 8/30
429/429 [==============================] - 41s 96ms/step - loss: 0.0641 - acc
uracy: 0.9815 - val_loss: 0.0333 - val_accuracy: 0.9884
Epoch 9/30
429/429 [==============================] - 43s 100ms/step - loss: 0.0525 - ac
curacy: 0.9806 - val_loss: 0.0207 - val_accuracy: 0.9923
Epoch 10/30
429/429 [==============================] - 41s 94ms/step - loss: 0.0464 - acc
uracy: 0.9866 - val_loss: 0.0159 - val_accuracy: 0.9958
Epoch 11/30
429/429 [==============================] - 42s 98ms/step - loss: 0.0421 - acc
uracy: 0.9867 - val_loss: 0.0220 - val_accuracy: 0.9905
Epoch 12/30
429/429 [==============================] - 42s 98ms/step - loss: 0.0372 - acc
uracy: 0.9865 - val_loss: 0.0083 - val_accuracy: 0.9992
Epoch 13/30
429/429 [==============================] - 42s 98ms/step - loss: 0.0331 - acc
uracy: 0.9893 - val_loss: 0.0218 - val_accuracy: 0.9918
Epoch 14/30
429/429 [==============================] - 39s 92ms/step - loss: 0.0295 - acc
uracy: 0.9904 - val_loss: 0.0223 - val_accuracy: 0.9930
Epoch 15/30
429/429 [==============================] - 40s 93ms/step - loss: 0.0292 - acc
uracy: 0.9913 - val_loss: 0.0294 - val_accuracy: 0.9895
Epoch 16/30
429/429 [==============================] - 44s 101ms/step - loss: 0.0302 - ac
curacy: 0.9915 - val_loss: 0.0136 - val_accuracy: 0.9953
Epoch 17/30
429/429 [==============================] - 45s 105ms/step - loss: 0.0245 - ac
curacy: 0.9923 - val_loss: 0.0106 - val_accuracy: 0.9947
Epoch 18/30
429/429 [==============================] - 41s 95ms/step - loss: 0.0236 - acc
uracy: 0.9931 - val_loss: 0.0128 - val_accuracy: 0.9936
Epoch 19/30
429/429 [==============================] - 41s 95ms/step - loss: 0.0192 - acc
uracy: 0.9934 - val_loss: 0.0097 - val_accuracy: 0.9967
```

```
Epoch 20/30
429/429 [==============================] - 42s 98ms/step - loss: 0.0255 - acc
uracy: 0.9927 - val_loss: 0.0097 - val_accuracy: 0.9968
Epoch 21/30
429/429 [==============================] - 41s 96ms/step - loss: 0.0197 - acc
uracy: 0.9944 - val_loss: 0.0032 - val_accuracy: 0.9997
Epoch 22/30
429/429 [==============================] - 40s 93ms/step - loss: 0.0180 - acc
uracy: 0.9936 - val_loss: 0.0139 - val_accuracy: 0.9937
Epoch 23/30
429/429 [==============================] - 42s 98ms/step - loss: 0.0167 - acc
uracy: 0.9952 - val_loss: 0.0112 - val_accuracy: 0.9969
Epoch 24/30
429/429 [==============================] - 42s 98ms/step - loss: 0.0170 - acc
uracy: 0.9955 - val_loss: 0.0121 - val_accuracy: 0.9968
Epoch 25/30
429/429 [==============================] - 43s 100ms/step - loss: 0.0178 - ac
curacy: 0.9938 - val_loss: 0.0131 - val_accuracy: 0.9927
Epoch 26/30
429/429 [==============================] - 42s 97ms/step - loss: 0.0155 - acc
uracy: 0.9965 - val_loss: 0.0079 - val_accuracy: 0.9985
Epoch 27/30
429/429 [==============================] - 40s 93ms/step - loss: 0.0158 - acc
uracy: 0.9953 - val_loss: 0.0026 - val_accuracy: 0.9999
Epoch 28/30
429/429 [==============================] - 40s 93ms/step - loss: 0.0092 - acc
uracy: 0.9974 - val_loss: 0.0018 - val_accuracy: 1.0000
Epoch 29/30
429/429 [==============================] - 39s 92ms/step - loss: 0.0145 - acc
uracy: 0.9958 - val_loss: 0.0196 - val_accuracy: 0.9925
Epoch 30/30
429/429 [==============================] - 41s 96ms/step - loss: 0.0151 - acc
uracy: 0.9952 - val_loss: 0.0158 - val_accuracy: 0.9933
```

```
Out[4]: [14.791172086494432, 0.9689069]
```

In [5]:
```
# Plotting the chart for accuracy and loss on both training and validation
%matplotlib inline
import matplotlib.pyplot as plt
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'r', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()

plt.plot(epochs, loss, 'r', label='Training Loss')
plt.plot(epochs, val_loss, 'b', label='Validation Loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```





In [ ]: