Training

May 16, 2022

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix,accuracy_score,␣
 ↪classification_report, multilabel_confusion_matrix
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPool2D, Flatten, Dense, Dropout,␣
 ↪BatchNormalization
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
train = pd.read_csv('./sign_mnist_train.csv')
test = pd.read_csv('./sign_mnist_test.csv')
```

[3]:
```
train.head()
```

[3]:

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	\
0	3	107	118	127	134	139	143	146	150	
1	6	155	157	156	156	156	157	156	158	
2	2	187	188	188	187	187	186	187	188	
3	2	211	211	212	212	211	210	211	210	
4	13	164	167	170	172	176	179	180	184	

	pixel9	…	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	\
0	153	…	207	207	207	207	206	206	
1	158	…	69	149	128	87	94	163	
2	187	…	202	201	200	199	198	199	
3	210	…	235	234	233	231	230	226	
4	185	…	92	105	105	108	133	163	

	pixel781	pixel782	pixel783	pixel784
0	206	204	203	202
1	175	103	135	149
2	198	195	194	195

```
3       225       222       229       163
4       157       163       164       179

[5 rows x 785 columns]
```

[4]:
```
train.isnull()
```

[4]:
```
       label  pixel1  pixel2  pixel3  pixel4  pixel5  pixel6  pixel7  pixel8  \
0      False   False   False   False   False   False   False   False   False
1      False   False   False   False   False   False   False   False   False
2      False   False   False   False   False   False   False   False   False
3      False   False   False   False   False   False   False   False   False
4      False   False   False   False   False   False   False   False   False
…      …     …     …     …     …     …     …     …     …
27450  False   False   False   False   False   False   False   False   False
27451  False   False   False   False   False   False   False   False   False
27452  False   False   False   False   False   False   False   False   False
27453  False   False   False   False   False   False   False   False   False
27454  False   False   False   False   False   False   False   False   False

       pixel9  …  pixel775  pixel776  pixel777  pixel778  pixel779  \
0       False  …     False     False     False     False     False
1       False  …     False     False     False     False     False
2       False  …     False     False     False     False     False
3       False  …     False     False     False     False     False
4       False  …     False     False     False     False     False
…       …  …       …       …       …       …
27450   False  …     False     False     False     False     False
27451   False  …     False     False     False     False     False
27452   False  …     False     False     False     False     False
27453   False  …     False     False     False     False     False
27454   False  …     False     False     False     False     False

       pixel780  pixel781  pixel782  pixel783  pixel784
0         False     False     False     False     False
1         False     False     False     False     False
2         False     False     False     False     False
3         False     False     False     False     False
4         False     False     False     False     False
…         …       …       …       …
27450     False     False     False     False     False
27451     False     False     False     False     False
27452     False     False     False     False     False
27453     False     False     False     False     False
27454     False     False     False     False     False

[27455 rows x 785 columns]
```

[5]:
```
train.isna().sum()
```

[5]:
```
label       0
pixel1      0
pixel2      0
pixel3      0
pixel4      0
           ..
pixel780    0
pixel781    0
pixel782    0
pixel783    0
pixel784    0
Length: 785, dtype: int64
```

[6]:
```
train_df_original = train.copy()

# Split into training, test and validation sets
val_index = int(train.shape[0]*0.2)

train_df = train_df_original.iloc[val_index:]
val_df = train_df_original.iloc[:val_index]
```

[7]:
```
y = np.array(train_df['label'])
X = np.array(train_df.drop(columns='label'))
```

[8]:
```
X.shape,y.shape
```

[8]:
```
((21964, 784), (21964,))
```

[9]:
```
import random
r = random.randint(0,(21964-1))
def show_img():
    arr = np.array(X)
    some_value = arr[r]
    some_img = some_value.reshape(28,28)
    plt.imshow(some_img, cmap="gray")
    plt.axis("off")
    plt.show()

show_img()
print(y[r])
```

```
12
```

[10]:
```
y_train = pd.get_dummies(y)
y_train.head(5)
```

[10]:

	0	1	2	3	4	5	6	7	8	10	…	15	16	17	18	19	20	21	22	23	24
0	0	0	0	0	0	0	0	0	0	0	…	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	1	0	…	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	…	0	0	0	0	0	1	0	0	0	0
3	0	0	0	0	0	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	…	1	0	0	0	0	0	0	0	0	0

[5 rows x 24 columns]

[11]:
```
y_val = val_df['label']
X_val = val_df.drop(columns="label",axis=1)
```

[12]:
```
y_val = pd.get_dummies(y_val)
```

[13]:
```
y_train.shape
```

[13]:
```
(21964, 24)
```

[14]:
```
X_val = pd.DataFrame(X_val).values.reshape(X_val.shape[0] ,28, 28, 1)
```

[15]:
```
X_train = pd.DataFrame(X).values.reshape(X.shape[0] ,28, 28, 1)
```

[16]:
```
X_train.shape,y_train.shape
```

[16]:
```
((21964, 28, 28, 1), (21964, 24))
```

[17]:
```
generator = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,
    rotation_range=10,
    zoom_range=0.10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    horizontal_flip=False,
    fill_mode="nearest"
)

X_train_flow = generator.flow(X_train, y_train, batch_size=32)

X_val_flow = generator.flow(X_val, y_val, batch_size=32)
```

[18]:
```
model = Sequential()

model.add(Conv2D(filters=32,  kernel_size=(3,3), activation="relu",␣
 ↪input_shape=(28,28,1)))
model.add(MaxPool2D((2,2),padding='SAME'))
model.add(Dropout(rate=0.2))


model.add(Conv2D(filters=64,  kernel_size=(3,3), activation="relu",␣
 ↪input_shape=(28,28,1)))
model.add(MaxPool2D((2,2),padding='SAME'))
model.add(Dropout(rate=0.2))


model.add(Conv2D(filters=521,  kernel_size=(3,3), activation="relu",␣
 ↪input_shape=(28,28,1)))
model.add(MaxPool2D((2,2),padding='SAME'))
model.add(Dropout(rate=0.2))



model.add(Flatten())
```

```
model.add(Dense(units=521, activation="relu"))
model.add(Dense(units=256, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(units=24, activation="softmax"))


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

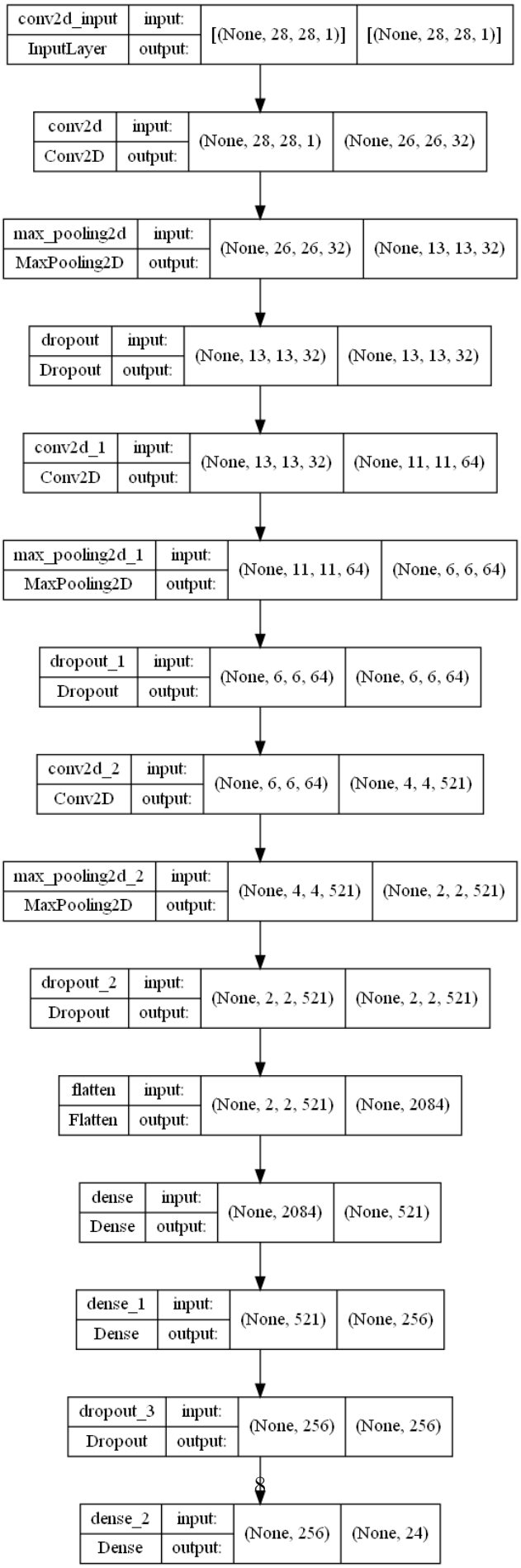
[19]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 dropout (Dropout)           (None, 13, 13, 32)        0

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 dropout_1 (Dropout)         (None, 6, 6, 64)          0

 conv2d_2 (Conv2D)           (None, 4, 4, 521)         300617

 max_pooling2d_2 (MaxPooling  (None, 2, 2, 521)        0
 2D)

 dropout_2 (Dropout)         (None, 2, 2, 521)         0

 flatten (Flatten)           (None, 2084)              0

 dense (Dense)               (None, 521)               1086285

 dense_1 (Dense)             (None, 256)               133632

 dropout_3 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 24)                6168

=================================================================
```

```
Total params: 1,545,518
Trainable params: 1,545,518
Non-trainable params: 0
_________________________________________________________________
```

[28]:
```
dot_img_file = './SignLanguageRecognitionModel.png'
tf.keras.utils.plot_model(model, to_file=dot_img_file, show_shapes=True)
```

[28]:


conv2d_input | input: | [(None, 28, 28, 1)] | [(None, 28, 28, 1)]
InputLayer | output:
conv2d | input: | (None, 28, 28, 1) | (None, 26, 26, 32)
Conv2D | output:
max_pooling2d | input: | (None, 26, 26, 32) | (None, 13, 13, 32)
MaxPooling2D | output:
dropout | input: | (None, 13, 13, 32) | (None, 13, 13, 32)
Dropout | output:
conv2d_1 | input: | (None, 13, 13, 32) | (None, 11, 11, 64)
Conv2D | output:
max_pooling2d_1 | input: | (None, 11, 11, 64) | (None, 6, 6, 64)
MaxPooling2D | output:
dropout_1 | input: | (None, 6, 6, 64) | (None, 6, 6, 64)
Dropout | output:
conv2d_2 | input: | (None, 6, 6, 64) | (None, 4, 4, 521)
Conv2D | output:
max_pooling2d_2 | input: | (None, 4, 4, 521) | (None, 2, 2, 521)
MaxPooling2D | output:
dropout_2 | input: | (None, 2, 2, 521) | (None, 2, 2, 521)
Dropout | output:
flatten | input: | (None, 2, 2, 521) | (None, 2084)
Flatten | output:
dense | input: | (None, 2084) | (None, 521)
Dense | output:
dense_1 | input: | (None, 521) | (None, 256)
Dense | output:
dropout_3 | input: | (None, 256) | (None, 256)
Dropout | output:
dense_2 | input: | (None, 256) | (None, 24)
Dense | output:

[20]:
```
learning_rate_reduction = tf.keras.callbacks.ReduceLROnPlateau(
    monitor='val_accuracy', patience = 2, verbose=1,factor=0.5, min_lr=0.00001
)
```

[21]:
```
history = model.fit(
    X_train_flow,
    validation_data=X_val_flow,
    epochs=100,
    callbacks=[
               tf.keras.callbacks.EarlyStopping(
                   monitor='val_loss',
                   patience=5,
                   restore_best_weights=True
                   ),
      learning_rate_reduction
    ])
```

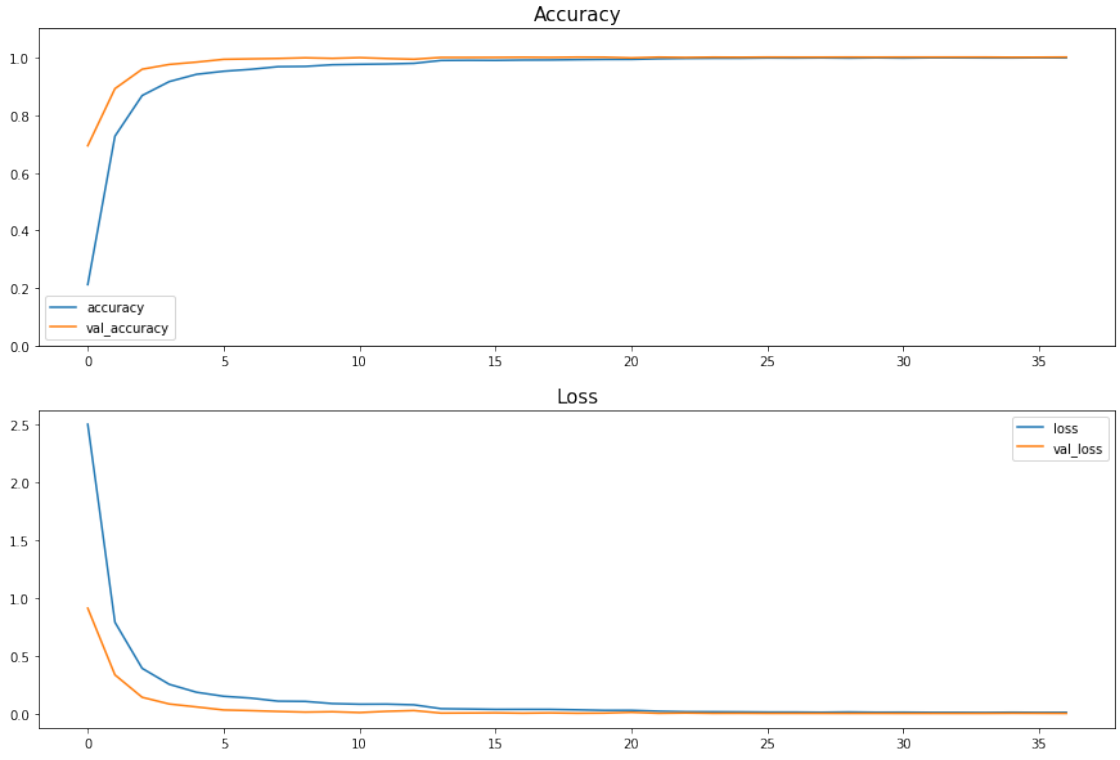
```
Epoch 1/100
687/687 [==============================] - 29s 41ms/step - loss: 2.5028 -
accuracy: 0.2132 - val_loss: 0.9107 - val_accuracy: 0.6933 - lr: 0.0010
Epoch 2/100
687/687 [==============================] - 29s 42ms/step - loss: 0.7890 -
accuracy: 0.7263 - val_loss: 0.3336 - val_accuracy: 0.8907 - lr: 0.0010
Epoch 3/100
687/687 [==============================] - 30s 43ms/step - loss: 0.3902 -
accuracy: 0.8669 - val_loss: 0.1397 - val_accuracy: 0.9579 - lr: 0.0010
Epoch 4/100
687/687 [==============================] - 28s 41ms/step - loss: 0.2517 -
accuracy: 0.9152 - val_loss: 0.0814 - val_accuracy: 0.9745 - lr: 0.0010
Epoch 5/100
687/687 [==============================] - 26s 38ms/step - loss: 0.1832 -
accuracy: 0.9403 - val_loss: 0.0561 - val_accuracy: 0.9825 - lr: 0.0010
Epoch 6/100
687/687 [==============================] - 26s 38ms/step - loss: 0.1482 -
accuracy: 0.9509 - val_loss: 0.0297 - val_accuracy: 0.9922 - lr: 0.0010
Epoch 7/100
687/687 [==============================] - 26s 38ms/step - loss: 0.1325 -
accuracy: 0.9576 - val_loss: 0.0240 - val_accuracy: 0.9938 - lr: 0.0010
Epoch 8/100
687/687 [==============================] - 26s 38ms/step - loss: 0.1058 -
accuracy: 0.9667 - val_loss: 0.0168 - val_accuracy: 0.9951 - lr: 0.0010
Epoch 9/100
687/687 [==============================] - 26s 38ms/step - loss: 0.1039 -
accuracy: 0.9677 - val_loss: 0.0112 - val_accuracy: 0.9975 - lr: 0.0010
Epoch 10/100
```

```
687/687 [==============================] - 26s 38ms/step - loss: 0.0850 -
accuracy: 0.9734 - val_loss: 0.0145 - val_accuracy: 0.9954 - lr: 0.0010
Epoch 11/100
687/687 [==============================] - 26s 38ms/step - loss: 0.0796 -
accuracy: 0.9749 - val_loss: 0.0072 - val_accuracy: 0.9980 - lr: 0.0010
Epoch 12/100
687/687 [==============================] - 26s 38ms/step - loss: 0.0803 -
accuracy: 0.9761 - val_loss: 0.0180 - val_accuracy: 0.9949 - lr: 0.0010
Epoch 13/100
686/687 [============================>.] - ETA: 0s - loss: 0.0736 - accuracy:
0.9783
Epoch 13: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
687/687 [==============================] - 26s 38ms/step - loss: 0.0741 -
accuracy: 0.9782 - val_loss: 0.0247 - val_accuracy: 0.9925 - lr: 0.0010
Epoch 14/100
687/687 [==============================] - 26s 38ms/step - loss: 0.0408 -
accuracy: 0.9883 - val_loss: 0.0026 - val_accuracy: 0.9987 - lr: 5.0000e-04
Epoch 15/100
687/687 [==============================] - 26s 38ms/step - loss: 0.0381 -
accuracy: 0.9888 - val_loss: 0.0036 - val_accuracy: 0.9987 - lr: 5.0000e-04
Epoch 16/100
687/687 [==============================] - 26s 38ms/step - loss: 0.0348 -
accuracy: 0.9886 - val_loss: 0.0048 - val_accuracy: 0.9989 - lr: 5.0000e-04
Epoch 17/100
687/687 [==============================] - 26s 38ms/step - loss: 0.0351 -
accuracy: 0.9897 - val_loss: 0.0019 - val_accuracy: 0.9995 - lr: 5.0000e-04
Epoch 18/100
687/687 [==============================] - 26s 38ms/step - loss: 0.0349 -
accuracy: 0.9900 - val_loss: 0.0046 - val_accuracy: 0.9989 - lr: 5.0000e-04
Epoch 19/100
687/687 [==============================] - 31s 45ms/step - loss: 0.0310 -
accuracy: 0.9911 - val_loss: 0.0015 - val_accuracy: 0.9998 - lr: 5.0000e-04
Epoch 20/100
687/687 [==============================] - 27s 39ms/step - loss: 0.0265 -
accuracy: 0.9919 - val_loss: 0.0028 - val_accuracy: 0.9996 - lr: 5.0000e-04
Epoch 21/100
687/687 [==============================] - ETA: 0s - loss: 0.0268 - accuracy:
0.9920
Epoch 21: ReduceLROnPlateau reducing learning rate to 0.0002502500011874362.
687/687 [==============================] - 27s 39ms/step - loss: 0.0268 -
accuracy: 0.9920 - val_loss: 0.0092 - val_accuracy: 0.9971 - lr: 5.0000e-04
Epoch 22/100
687/687 [==============================] - 27s 39ms/step - loss: 0.0190 -
accuracy: 0.9942 - val_loss: 0.0010 - val_accuracy: 0.9998 - lr: 2.5000e-04
Epoch 23/100
687/687 [==============================] - ETA: 0s - loss: 0.0147 - accuracy:
0.9953
Epoch 23: ReduceLROnPlateau reducing learning rate to 0.0001251250005937181.
```

```
687/687 [==============================] - 27s 39ms/step - loss: 0.0147 -
accuracy: 0.9953 - val_loss: 0.0034 - val_accuracy: 0.9987 - lr: 2.5000e-04
Epoch 24/100
687/687 [==============================] - 27s 39ms/step - loss: 0.0141 -
accuracy: 0.9958 - val_loss: 6.4741e-04 - val_accuracy: 0.9998 - lr: 1.2500e-04
Epoch 25/100
686/687 [============================>.] - ETA: 0s - loss: 0.0132 - accuracy:
0.9959
Epoch 25: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
687/687 [==============================] - 27s 39ms/step - loss: 0.0132 -
accuracy: 0.9959 - val_loss: 0.0011 - val_accuracy: 0.9995 - lr: 1.2500e-04
Epoch 26/100
687/687 [==============================] - 27s 39ms/step - loss: 0.0115 -
accuracy: 0.9969 - val_loss: 2.8248e-04 - val_accuracy: 1.0000 - lr: 6.2500e-05
Epoch 27/100
687/687 [==============================] - 26s 37ms/step - loss: 0.0114 -
accuracy: 0.9965 - val_loss: 4.3466e-04 - val_accuracy: 0.9998 - lr: 6.2500e-05
Epoch 28/100
687/687 [==============================] - ETA: 0s - loss: 0.0098 - accuracy:
0.9971
Epoch 28: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
687/687 [==============================] - 26s 37ms/step - loss: 0.0098 -
accuracy: 0.9971 - val_loss: 3.3614e-04 - val_accuracy: 0.9998 - lr: 6.2500e-05
Epoch 29/100
687/687 [==============================] - 27s 39ms/step - loss: 0.0123 -
accuracy: 0.9963 - val_loss: 1.6902e-04 - val_accuracy: 1.0000 - lr: 3.1250e-05
Epoch 30/100
687/687 [==============================] - ETA: 0s - loss: 0.0099 - accuracy:
0.9973
Epoch 30: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
687/687 [==============================] - 27s 39ms/step - loss: 0.0099 -
accuracy: 0.9973 - val_loss: 2.1925e-04 - val_accuracy: 0.9998 - lr: 3.1250e-05
Epoch 31/100
687/687 [==============================] - 27s 39ms/step - loss: 0.0103 -
accuracy: 0.9965 - val_loss: 1.2029e-04 - val_accuracy: 1.0000 - lr: 1.5625e-05
Epoch 32/100
686/687 [============================>.] - ETA: 0s - loss: 0.0089 - accuracy:
0.9974
Epoch 32: ReduceLROnPlateau reducing learning rate to 1e-05.
687/687 [==============================] - 27s 39ms/step - loss: 0.0089 -
accuracy: 0.9975 - val_loss: 8.1509e-05 - val_accuracy: 1.0000 - lr: 1.5625e-05
Epoch 33/100
687/687 [==============================] - 27s 39ms/step - loss: 0.0087 -
accuracy: 0.9974 - val_loss: 1.1326e-04 - val_accuracy: 1.0000 - lr: 1.0000e-05
Epoch 34/100
687/687 [==============================] - 27s 39ms/step - loss: 0.0079 -
accuracy: 0.9971 - val_loss: 1.4355e-04 - val_accuracy: 1.0000 - lr: 1.0000e-05
Epoch 35/100
```

```
687/687 [==============================] - 28s 41ms/step - loss: 0.0097 -
accuracy: 0.9970 - val_loss: 0.0013 - val_accuracy: 0.9995 - lr: 1.0000e-05
Epoch 36/100
687/687 [==============================] - 27s 40ms/step - loss: 0.0085 -
accuracy: 0.9976 - val_loss: 7.3692e-04 - val_accuracy: 0.9996 - lr: 1.0000e-05
Epoch 37/100
687/687 [==============================] - 27s 39ms/step - loss: 0.0091 -
accuracy: 0.9973 - val_loss: 2.8997e-04 - val_accuracy: 1.0000 - lr: 1.0000e-05
```

[22]:
```
fig, axes = plt.subplots(2, 1, figsize=(15, 10))
ax = axes.flat

pd.DataFrame(history.history)[['accuracy','val_accuracy']].plot(ax=ax[0])
ax[0].set_title("Accuracy", fontsize = 15)
ax[0].set_ylim(0,1.1)

pd.DataFrame(history.history)[['loss','val_loss']].plot(ax=ax[1])
ax[1].set_title("Loss", fontsize = 15)
plt.show()
```



[23]:
```
y_test = np.array(test['label'])
X_test = np.array(test.drop(columns='label'))
```

```
y_test = pd.get_dummies(y_test)
X_test = pd.DataFrame(X_test).values.reshape(X_test.shape[0] ,28, 28, 1)

# X_test_flow = generator.flow(X_test, y_test, batch_size=32)
# X_test.shape,X_train.shape

y_test = pd.get_dummies(y_test)
```

[24]:
```
from sklearn.metrics import classification_report

# predictions
pred = model.predict(X_test)

y_pred = np.argmax(pred,axis=1)
y_test = np.argmax(y_test.values,axis=1)
```

[25]:
```
acc = accuracy_score(y_test,y_pred)

# # Display the results
print(f'## {acc*100:.2f}% accuracy on the test set')
```

```
## 99.51% accuracy on the test set
```

[32]:
```
fer_json = model.to_json()
with open("./SavedModel/SignLanguageRecognitionModel.json", "w") as json_file:
    json_file.write(fer_json)
model.save('./SavedModel/SignLanguageRecognitionModel_tf',save_format='tf')
model.save("./SavedModel/SignLanguageRecognitionModel.h5")
model.save_weights("./SavedModel/SignLanguageRecognitionModel.h5")
```

```
INFO:tensorflow:Assets written to:
./SavedModel/SignLanguageRecognitionModel_tf\assets
```