GTSRB_CNN

January 7, 2021

[2]:
```
pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.6/dist-packages
(1.5.9)
Requirement already satisfied: slugify in /usr/local/lib/python3.6/dist-packages
(from kaggle) (0.0.1)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.6/dist-
packages (from kaggle) (4.0.1)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.6/dist-
packages (from kaggle) (1.15.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-packages
(from kaggle) (4.41.1)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.6/dist-packages
(from kaggle) (1.24.3)
Requirement already satisfied: certifi in /usr/local/lib/python3.6/dist-packages
(from kaggle) (2020.6.20)
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-
packages (from kaggle) (2.23.0)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.6/dist-
packages (from kaggle) (2.8.1)
Requirement already satisfied: text-unidecode>=1.3 in
/usr/local/lib/python3.6/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.6/dist-packages (from requests->kaggle) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-
packages (from requests->kaggle) (2.10)
```

[1]:
```python
from PIL import Image
import tensorflow as tf
from tensorflow import keras
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
import os
import sklearn.model_selection as skl
from zipfile import ZipFile
import google.colab.files
```

```
# Specify plot label tick size
matplotlib.rc("xtick", labelsize=12)
matplotlib.rc("ytick", labelsize=12)
```

[2]:

```
#Downloading the dataset from kaggle
google.colab.files.upload()
!mkdir -p ~/.kaggle
!chmod 600 ~/.kaggle/kaggle.json
!cp kaggle.json ~/.kaggle
!kaggle datasets download -d meowmeowmeowmeowmeow/gtsrb-german-traffic-sign
```

```
<IPython.core.display.HTML object>
```

```
Saving kaggle.json to kaggle.json
chmod: cannot access '/root/.kaggle/kaggle.json': No such file or directory
Warning: Your Kaggle API key is readable by other users on this system! To fix
this, you can run 'chmod 600 /root/.kaggle/kaggle.json'
Downloading gtsrb-german-traffic-sign.zip to /content
 98% 599M/612M [00:03<00:00, 195MB/s]
100% 612M/612M [00:03<00:00, 206MB/s]
```

[3]:

```
with ZipFile('/content/gtsrb-german-traffic-sign.zip',mode='r') as info:
  info.extractall()
```

[12]:

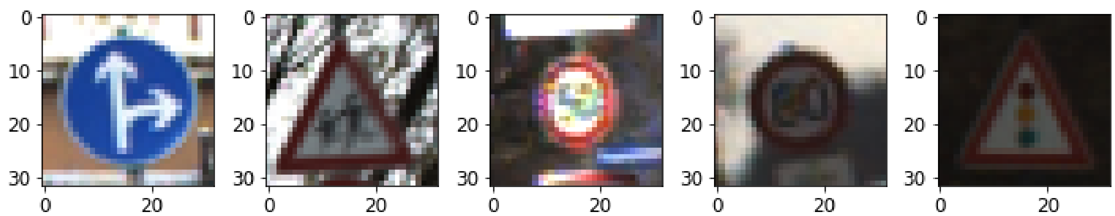
```
num_classes=43
test_data=[]
test_labels=[]
test_info=pd.read_csv('Test.csv')
image_data=[]
image_labels=[]

def dim_change(file_path):
  im=Image.open(file_path)
  im=im.resize((32,32))
  data=np.array(im).astype('float32')
  return data/255.0

  #Preparing the training data
for image_class in range(num_classes):
  cnt=1
  for file_name in os.listdir(f'Train/{image_class}'):
    image_data.append(dim_change(f'Train/{image_class}/{file_name}'))
    image_labels.append(image_class)
    # print(f'{cnt} Done')
    cnt+=1
  print(f'Class {image_class} done')
```

```python
image_data=np.array(image_data)
image_labels=np.array(image_labels)
print("Training Data Done")
print(image_data.shape)
print(image_labels.shape)

#Preparing the test data
print('Starting Test Data')
for id, path in zip(test_info.ClassId, test_info.Path):
  test_data.append(dim_change(path))
  test_labels.append(id)

test_data=np.array(test_data)
test_labels=np.array(test_labels)
print(test_data.shape)
print(test_labels.shape)
```

```
Class 0 done
Class 1 done
Class 2 done
Class 3 done
Class 4 done
Class 5 done
Class 6 done
Class 7 done
Class 8 done
Class 9 done
Class 10 done
Class 11 done
Class 12 done
Class 13 done
Class 14 done
Class 15 done
Class 16 done
Class 17 done
Class 18 done
Class 19 done
Class 20 done
Class 21 done
Class 22 done
Class 23 done
Class 24 done
Class 25 done
Class 26 done
Class 27 done
Class 28 done
Class 29 done
```

```
Class 30 done
Class 31 done
Class 32 done
Class 33 done
Class 34 done
Class 35 done
Class 36 done
Class 37 done
Class 38 done
Class 39 done
Class 40 done
Class 41 done
Class 42 done
Training Data Done
(39209, 32, 32, 3)
(39209,)
Starting Test Data
(12630, 32, 32, 3)
(12630,)
```

[13]:
```
#Split data into training and validation sets
x_train, x_val, y_train, y_val = skl.
 ↪train_test_split(image_data,image_labels,train_size=0.8, test_size=0.2,␣
 ↪random_state=42)

#Display sample data
fig=plt.figure(figsize=(10,20))
for i in range(5):
  fig.add_subplot(1,5,i+1)
  plt.imshow(x_train[i])
fig.tight_layout()
```



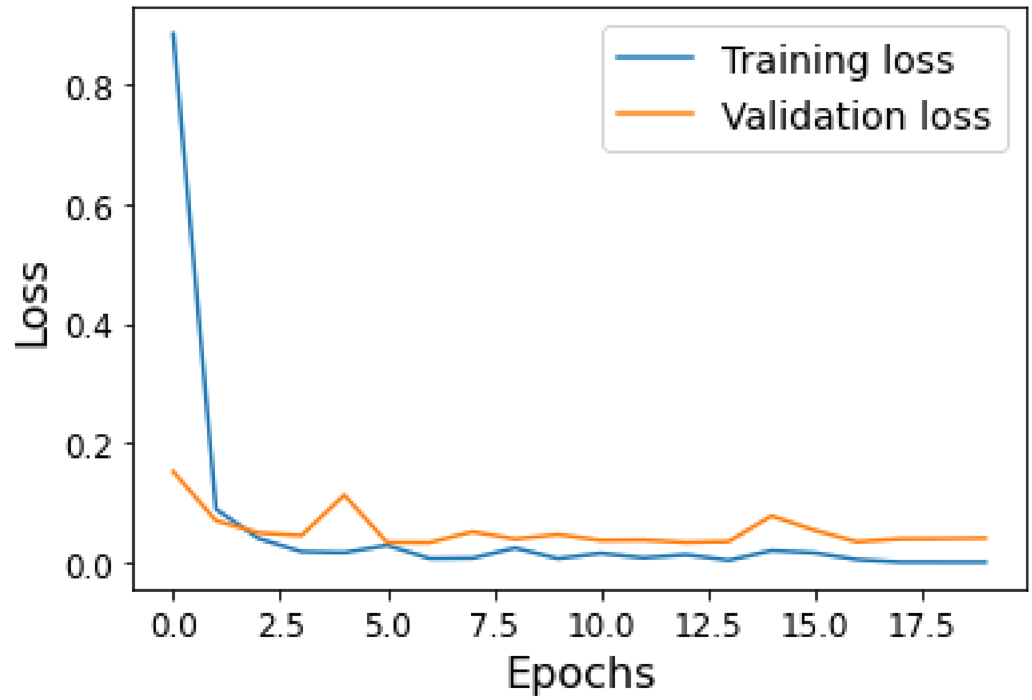
CNN MODEL WITH A CONV-POOL-CONV-POOL-CONV-POOL-FC-FC ARCHITECTURE

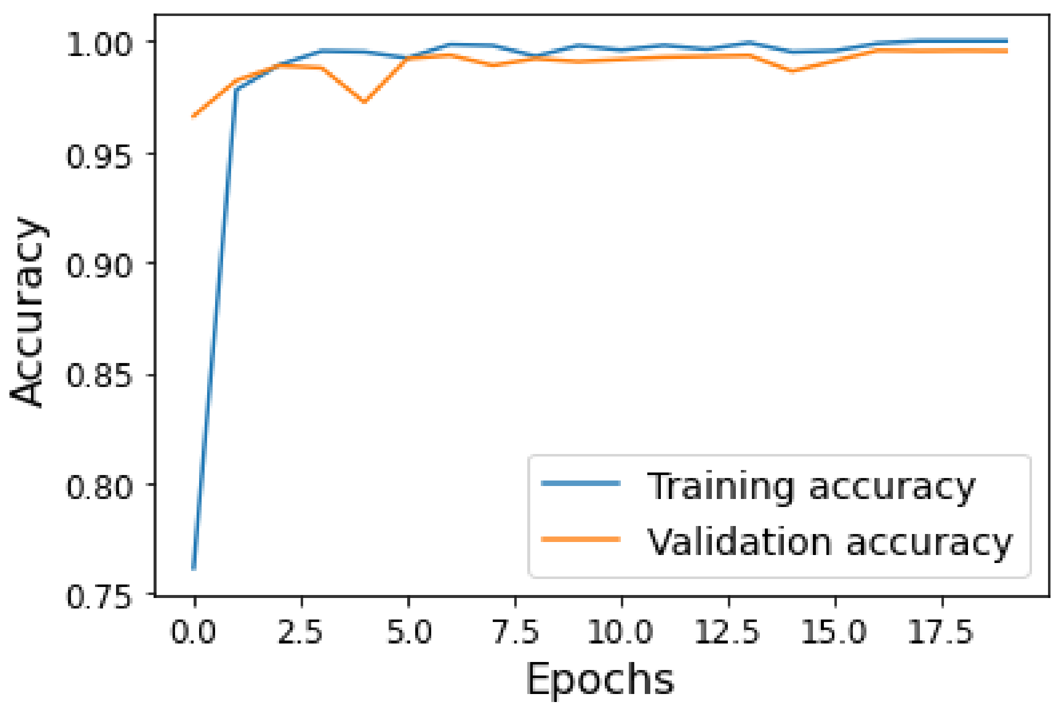
[14]:
```
model = keras.models.Sequential(
     [keras.layers.Conv2D(filters=128, kernel_size=(3,3), activation='relu',␣
 ↪input_shape=(32,32,3)),
```

```
        keras.layers.MaxPool2D(pool_size=(2, 2), strides=1),
        keras.layers.Conv2D(filters=128, kernel_size=(3, 3), activation='relu',␣
↪strides=1),
        keras.layers.MaxPool2D(pool_size=(2, 2), strides=1),
        keras.layers.Conv2D(filters=128, kernel_size=(3, 3), activation='relu',␣
↪strides=1),
        keras.layers.MaxPool2D(pool_size=(2, 2), strides=1),
        keras.layers.Flatten(),
        keras.layers.Dense(256, activation='relu'),
        keras.layers.Dense(43, activation='softmax')]
    )

# Compile model using adam optimizer
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',␣
↪metrics=['accuracy'])

# Train model
history=model.fit(x_train, y_train, batch_size=128,␣
↪epochs=20,validation_data=(x_val,y_val))

# Display various layers of CNN
model.summary()


# Evaluate model against test data
test_loss, test_acc = model.evaluate(test_data, test_labels, verbose=2)
print(f'\n Test accuracy: {test_acc}')


# Compare training and validation loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('Epochs',fontsize=16)
plt.ylabel('Loss', fontsize=16)
plt.legend(['Training loss', 'Validation loss'], fontsize=14)
plt.show()



# Plot training and validation accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.xlabel('Epochs',fontsize=16)
plt.ylabel('Accuracy', fontsize=16)
plt.legend(['Training accuracy', 'Validation accuracy'], fontsize=14)
plt.show()
```

```
Epoch 1/20
  1/246 […] - ETA: 7s - loss: 3.7560 - accuracy:
0.0078WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared
to the batch time (batch time: 0.0169s vs `on_train_batch_end` time: 0.0310s).
Check your callbacks.
246/246 [==============================] - 10s 40ms/step - loss: 0.8855 -
accuracy: 0.7614 - val_loss: 0.1517 - val_accuracy: 0.9660
Epoch 2/20
246/246 [==============================] - 10s 40ms/step - loss: 0.0887 -
accuracy: 0.9778 - val_loss: 0.0695 - val_accuracy: 0.9820
Epoch 3/20
246/246 [==============================] - 10s 40ms/step - loss: 0.0397 -
accuracy: 0.9890 - val_loss: 0.0492 - val_accuracy: 0.9888
Epoch 4/20
246/246 [==============================] - 10s 40ms/step - loss: 0.0176 -
accuracy: 0.9954 - val_loss: 0.0448 - val_accuracy: 0.9878
Epoch 5/20
246/246 [==============================] - 10s 39ms/step - loss: 0.0161 -
accuracy: 0.9950 - val_loss: 0.1124 - val_accuracy: 0.9722
Epoch 6/20
246/246 [==============================] - 10s 39ms/step - loss: 0.0284 -
accuracy: 0.9922 - val_loss: 0.0331 - val_accuracy: 0.9920
Epoch 7/20
246/246 [==============================] - 10s 39ms/step - loss: 0.0062 -
accuracy: 0.9984 - val_loss: 0.0330 - val_accuracy: 0.9935
Epoch 8/20
246/246 [==============================] - 10s 39ms/step - loss: 0.0068 -
accuracy: 0.9979 - val_loss: 0.0511 - val_accuracy: 0.9889
Epoch 9/20
246/246 [==============================] - 10s 39ms/step - loss: 0.0238 -
accuracy: 0.9930 - val_loss: 0.0390 - val_accuracy: 0.9920
Epoch 10/20
246/246 [==============================] - 10s 39ms/step - loss: 0.0061 -
accuracy: 0.9979 - val_loss: 0.0463 - val_accuracy: 0.9906
Epoch 11/20
246/246 [==============================] - 10s 39ms/step - loss: 0.0150 -
accuracy: 0.9958 - val_loss: 0.0368 - val_accuracy: 0.9916
Epoch 12/20
246/246 [==============================] - 10s 39ms/step - loss: 0.0074 -
accuracy: 0.9980 - val_loss: 0.0373 - val_accuracy: 0.9926
Epoch 13/20
246/246 [==============================] - 10s 39ms/step - loss: 0.0127 -
accuracy: 0.9961 - val_loss: 0.0333 - val_accuracy: 0.9930
Epoch 14/20
246/246 [==============================] - 10s 39ms/step - loss: 0.0031 -
accuracy: 0.9991 - val_loss: 0.0348 - val_accuracy: 0.9934
Epoch 15/20
246/246 [==============================] - 10s 39ms/step - loss: 0.0195 -
```

```
accuracy: 0.9949 - val_loss: 0.0774 - val_accuracy: 0.9862
Epoch 16/20
246/246 [==============================] - 10s 39ms/step - loss: 0.0156 -
accuracy: 0.9955 - val_loss: 0.0535 - val_accuracy: 0.9909
Epoch 17/20
246/246 [==============================] - 10s 39ms/step - loss: 0.0043 -
accuracy: 0.9988 - val_loss: 0.0343 - val_accuracy: 0.9957
Epoch 18/20
246/246 [==============================] - 10s 39ms/step - loss: 1.2158e-04 -
accuracy: 1.0000 - val_loss: 0.0391 - val_accuracy: 0.9954
Epoch 19/20
246/246 [==============================] - 10s 39ms/step - loss: 2.1851e-05 -
accuracy: 1.0000 - val_loss: 0.0395 - val_accuracy: 0.9955
Epoch 20/20
246/246 [==============================] - 10s 39ms/step - loss: 1.1452e-05 -
accuracy: 1.0000 - val_loss: 0.0403 - val_accuracy: 0.9954
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_22 (Conv2D)           (None, 30, 30, 128)       3584
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 29, 29, 128)       0
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 27, 27, 128)       147584
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 26, 26, 128)       0
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 24, 24, 128)       147584
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 23, 23, 128)       0
_________________________________________________________________
flatten_6 (Flatten)          (None, 67712)             0
_________________________________________________________________
dense_14 (Dense)             (None, 256)               17334528
_________________________________________________________________
dense_15 (Dense)             (None, 43)                11051
=================================================================
Total params: 17,644,331
Trainable params: 17,644,331
Non-trainable params: 0
_________________________________________________________________
395/395 - 2s - loss: 0.2763 - accuracy: 0.9677

 Test accuracy: 0.967695951461792
```





CNN MODEL WITH CONV-CONV-POOL-CONV-CONV-POOL-FC-FC ARCHI-

TECTURE

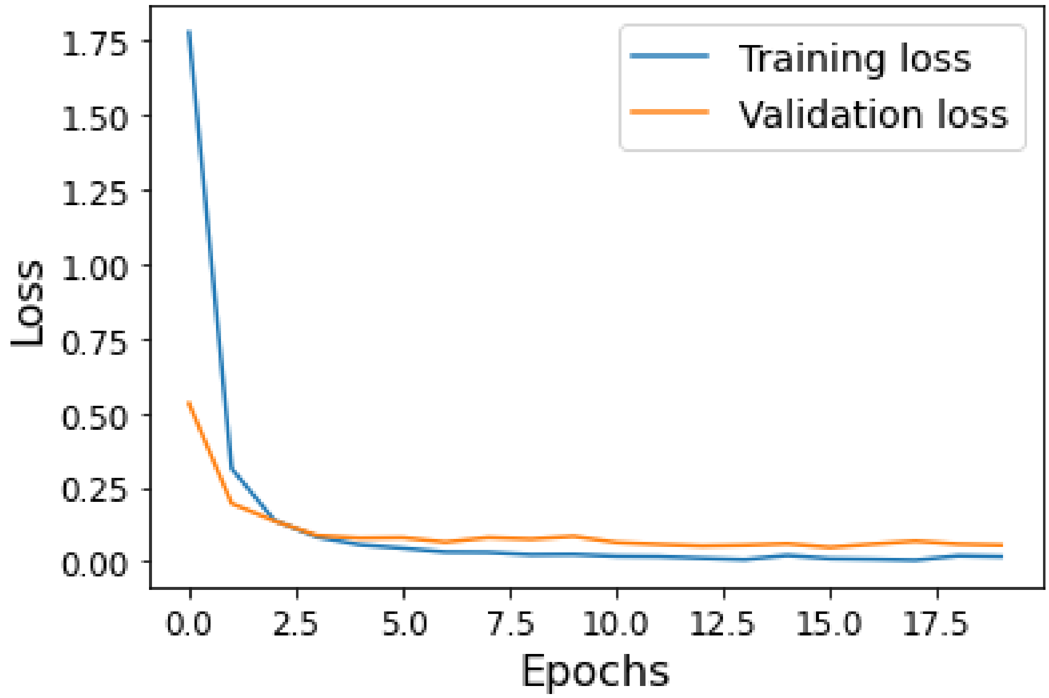
[15]:
```
model_mod1 = keras.models.Sequential(
    [keras.layers.Conv2D(filters=64, kernel_size=(3,3), activation='relu',␣
 ↪input_shape=(32,32,3)),
      keras.layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu',␣
 ↪strides=1),
      keras.layers.MaxPool2D(pool_size=(2, 2), strides=2),
      keras.layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu',␣
 ↪strides=1),
      keras.layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu',␣
 ↪strides=1),
      keras.layers.MaxPool2D(pool_size=(2, 2), strides=2),
      keras.layers.Flatten(),
      keras.layers.Dense(128, activation='relu'),
      keras.layers.Dense(43, activation='softmax')]
    )

# Compile model using adam optimizer
model_mod1.compile(optimizer='adam',loss='sparse_categorical_crossentropy',␣
 ↪metrics=['accuracy'])

# Train model
history_mod1=model_mod1.fit(x_train, y_train, batch_size=128,␣
 ↪epochs=20,validation_data=(x_val,y_val))

# Display various layers of CNN
model_mod1.summary()


# Evaluate model against test data
test_loss_mod1, test_acc_mod1 = model_mod1.evaluate(test_data, test_labels,␣
 ↪verbose=2)
print(f'\n Test accuracy: {test_acc_mod1}')


# Compare training and validation loss
plt.plot(history_mod1.history['loss'])
plt.plot(history_mod1.history['val_loss'])
plt.xlabel('Epochs',fontsize=16)
plt.ylabel('Loss', fontsize=16)
plt.legend(['Training loss', 'Validation loss'], fontsize=14)
plt.show()



# Plot training and validation accuracy
```

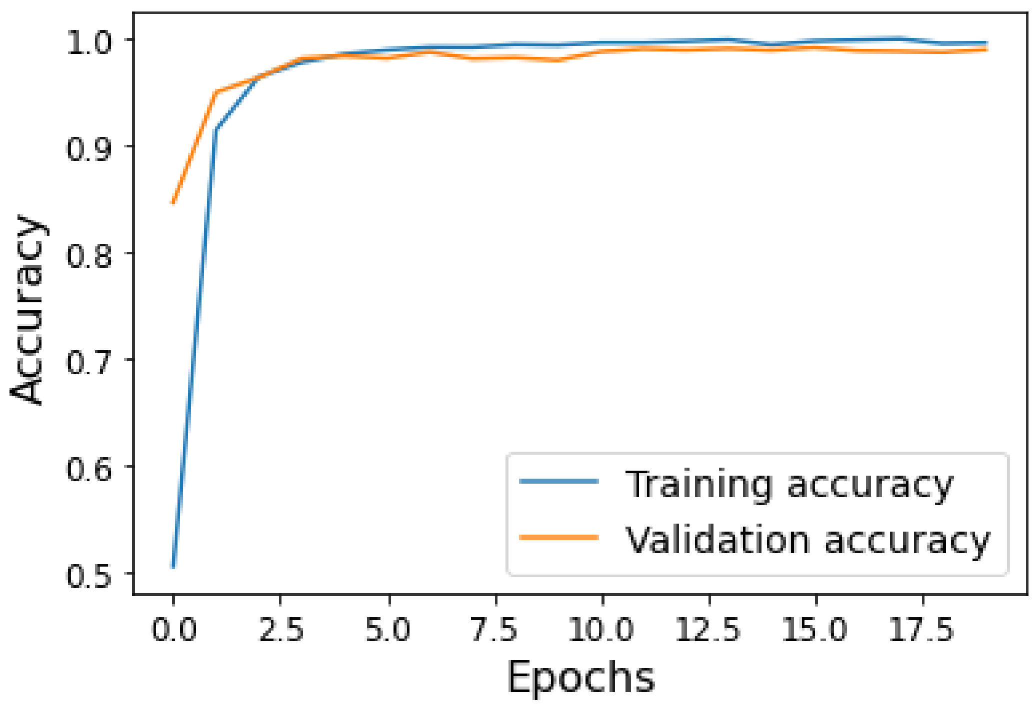
```python
plt.plot(history_mod1.history['accuracy'])
plt.plot(history_mod1.history['val_accuracy'])
plt.xlabel('Epochs',fontsize=16)
plt.ylabel('Accuracy', fontsize=16)
plt.legend(['Training accuracy', 'Validation accuracy'], fontsize=14)
plt.show()
```

```
Epoch 1/20
246/246 [==============================] - 3s 10ms/step - loss: 1.7775 -
accuracy: 0.5047 - val_loss: 0.5319 - val_accuracy: 0.8457
Epoch 2/20
246/246 [==============================] - 2s 9ms/step - loss: 0.3131 -
accuracy: 0.9132 - val_loss: 0.1957 - val_accuracy: 0.9486
Epoch 3/20
246/246 [==============================] - 2s 9ms/step - loss: 0.1389 -
accuracy: 0.9631 - val_loss: 0.1370 - val_accuracy: 0.9625
Epoch 4/20
246/246 [==============================] - 2s 9ms/step - loss: 0.0828 -
accuracy: 0.9767 - val_loss: 0.0883 - val_accuracy: 0.9804
Epoch 5/20
246/246 [==============================] - 2s 9ms/step - loss: 0.0570 -
accuracy: 0.9843 - val_loss: 0.0792 - val_accuracy: 0.9827
Epoch 6/20
246/246 [==============================] - 2s 9ms/step - loss: 0.0453 -
accuracy: 0.9881 - val_loss: 0.0810 - val_accuracy: 0.9805
Epoch 7/20
246/246 [==============================] - 2s 9ms/step - loss: 0.0323 -
accuracy: 0.9908 - val_loss: 0.0657 - val_accuracy: 0.9862
Epoch 8/20
246/246 [==============================] - 2s 9ms/step - loss: 0.0314 -
accuracy: 0.9907 - val_loss: 0.0813 - val_accuracy: 0.9801
Epoch 9/20
246/246 [==============================] - 2s 9ms/step - loss: 0.0235 -
accuracy: 0.9933 - val_loss: 0.0762 - val_accuracy: 0.9811
Epoch 10/20
246/246 [==============================] - 2s 9ms/step - loss: 0.0247 -
accuracy: 0.9926 - val_loss: 0.0854 - val_accuracy: 0.9787
Epoch 11/20
246/246 [==============================] - 2s 9ms/step - loss: 0.0184 -
accuracy: 0.9951 - val_loss: 0.0646 - val_accuracy: 0.9866
Epoch 12/20
246/246 [==============================] - 2s 9ms/step - loss: 0.0171 -
accuracy: 0.9951 - val_loss: 0.0583 - val_accuracy: 0.9892
Epoch 13/20
246/246 [==============================] - 2s 9ms/step - loss: 0.0116 -
accuracy: 0.9963 - val_loss: 0.0531 - val_accuracy: 0.9881
Epoch 14/20
```

```
246/246 [==============================] - 2s 9ms/step - loss: 0.0069 -
accuracy: 0.9978 - val_loss: 0.0556 - val_accuracy: 0.9894
Epoch 15/20
246/246 [==============================] - 2s 9ms/step - loss: 0.0214 -
accuracy: 0.9930 - val_loss: 0.0604 - val_accuracy: 0.9879
Epoch 16/20
246/246 [==============================] - 2s 9ms/step - loss: 0.0101 -
accuracy: 0.9965 - val_loss: 0.0483 - val_accuracy: 0.9904
Epoch 17/20
246/246 [==============================] - 2s 9ms/step - loss: 0.0082 -
accuracy: 0.9976 - val_loss: 0.0593 - val_accuracy: 0.9875
Epoch 18/20
246/246 [==============================] - 2s 9ms/step - loss: 0.0051 -
accuracy: 0.9986 - val_loss: 0.0690 - val_accuracy: 0.9869
Epoch 19/20
246/246 [==============================] - 2s 9ms/step - loss: 0.0199 -
accuracy: 0.9942 - val_loss: 0.0590 - val_accuracy: 0.9860
Epoch 20/20
246/246 [==============================] - 2s 9ms/step - loss: 0.0173 -
accuracy: 0.9949 - val_loss: 0.0559 - val_accuracy: 0.9885
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_25 (Conv2D)           (None, 30, 30, 64)        1792
_________________________________________________________________
conv2d_26 (Conv2D)           (None, 28, 28, 32)        18464
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_27 (Conv2D)           (None, 12, 12, 32)        9248
_________________________________________________________________
conv2d_28 (Conv2D)           (None, 10, 10, 16)        4624
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 5, 5, 16)          0
_________________________________________________________________
flatten_7 (Flatten)          (None, 400)               0
_________________________________________________________________
dense_16 (Dense)             (None, 128)               51328
_________________________________________________________________
dense_17 (Dense)             (None, 43)                5547
=================================================================
Total params: 91,003
Trainable params: 91,003
Non-trainable params: 0
_________________________________________________________________
395/395 - 1s - loss: 0.3570 - accuracy: 0.9481
```

```
Test accuracy: 0.948060154914856
```





DEFINE A MODEL WITH CONV-CONV-POOL-DROPOUT-CONV-CONV-POOL-DROPOUT-FC-DROPOUT-FC-DROPOUT-FC ARCHITECTURE

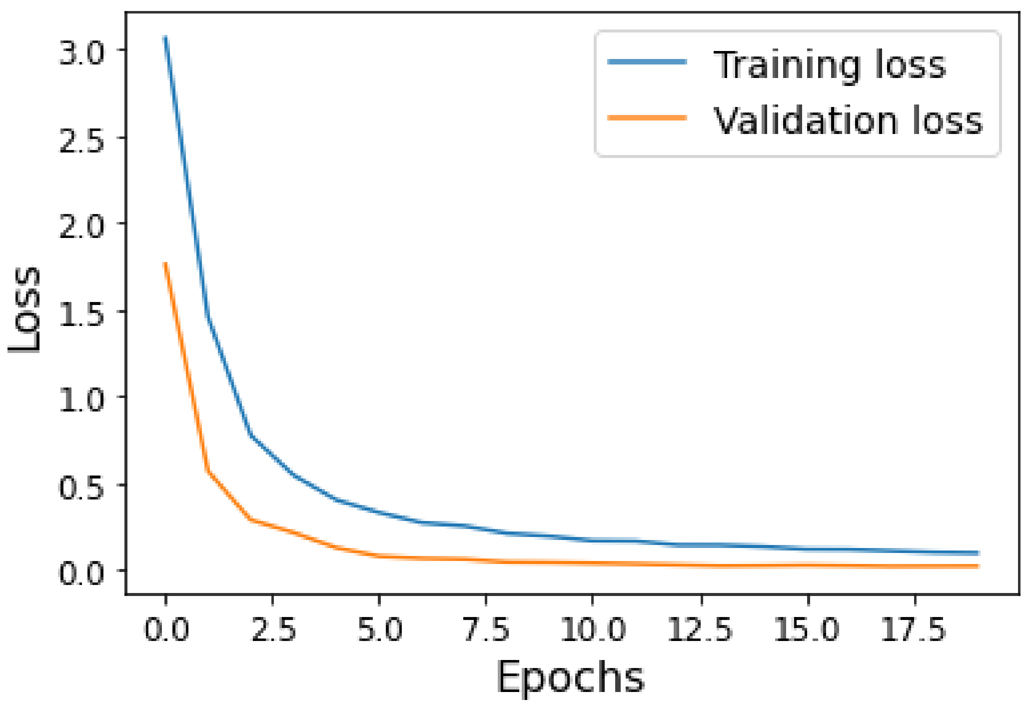
[16]:
```
model_mod2 = keras.models.Sequential(
    [keras.layers.Conv2D(filters=64, kernel_size=(3,3), activation='relu',␣
 ↪input_shape=(32,32,3), strides=1),
      keras.layers.Conv2D(filters=32, kernel_size=(3,3), activation='relu',␣
 ↪strides=1),
      keras.layers.MaxPool2D(pool_size=(2, 2), strides=2),
      keras.layers.Dropout(rate=0.25),
      keras.layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu',␣
 ↪strides=1),
      keras.layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu',␣
 ↪strides=1),
      keras.layers.MaxPool2D(pool_size=(2, 2), strides=2),
      keras.layers.Dropout(rate=0.25),
      keras.layers.Flatten(),
      keras.layers.Dense(256, activation='relu'),
      keras.layers.Dropout(rate=0.5),
      keras.layers.Dense(128, activation='relu'),
      keras.layers.Dropout(rate=0.5),
      keras.layers.Dense(43, activation='softmax')]
    )

# Compile model using adam optimizer
model_mod2.compile(optimizer='adam',loss='sparse_categorical_crossentropy',␣
 ↪metrics=['accuracy'])

# Train model
history_mod2=model_mod2.fit(x_train, y_train, batch_size=128,␣
 ↪epochs=20,validation_data=(x_val,y_val))

# Display various layers of CNN
model_mod2.summary()


# Evaluate model against test data
test_loss_mod2, test_acc_mod2 = model_mod2.evaluate(test_data, test_labels,␣
 ↪verbose=2)
print(f'\n Test accuracy: {test_acc_mod2}')


# Compare training and validation loss
plt.plot(history_mod2.history['loss'])
plt.plot(history_mod2.history['val_loss'])
plt.xlabel('Epochs',fontsize=16)
plt.ylabel('Loss', fontsize=16)
```

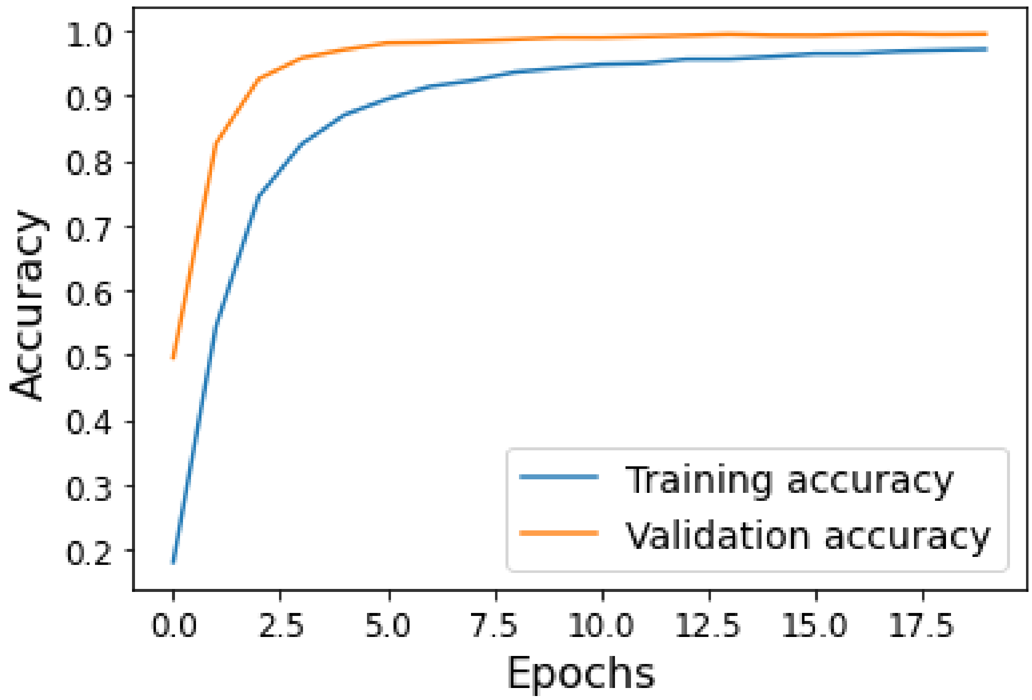
```python
plt.legend(['Training loss', 'Validation loss'], fontsize=14)
plt.show()



# Plot training and validation accuracy
plt.plot(history_mod2.history['accuracy'])
plt.plot(history_mod2.history['val_accuracy'])
plt.xlabel('Epochs',fontsize=16)
plt.ylabel('Accuracy', fontsize=16)
plt.legend(['Training accuracy', 'Validation accuracy'], fontsize=14)
plt.show()
```

```
Epoch 1/20
246/246 [==============================] - 3s 10ms/step - loss: 3.0621 -
accuracy: 0.1799 - val_loss: 1.7602 - val_accuracy: 0.4953
Epoch 2/20
246/246 [==============================] - 2s 10ms/step - loss: 1.4573 -
accuracy: 0.5433 - val_loss: 0.5699 - val_accuracy: 0.8257
Epoch 3/20
246/246 [==============================] - 2s 10ms/step - loss: 0.7758 -
accuracy: 0.7443 - val_loss: 0.2888 - val_accuracy: 0.9257
Epoch 4/20
246/246 [==============================] - 2s 10ms/step - loss: 0.5465 -
accuracy: 0.8247 - val_loss: 0.2156 - val_accuracy: 0.9578
Epoch 5/20
246/246 [==============================] - 2s 9ms/step - loss: 0.4038 -
accuracy: 0.8699 - val_loss: 0.1291 - val_accuracy: 0.9705
Epoch 6/20
246/246 [==============================] - 2s 10ms/step - loss: 0.3312 -
accuracy: 0.8944 - val_loss: 0.0799 - val_accuracy: 0.9807
Epoch 7/20
246/246 [==============================] - 2s 10ms/step - loss: 0.2727 -
accuracy: 0.9137 - val_loss: 0.0676 - val_accuracy: 0.9818
Epoch 8/20
246/246 [==============================] - 2s 10ms/step - loss: 0.2536 -
accuracy: 0.9232 - val_loss: 0.0631 - val_accuracy: 0.9838
Epoch 9/20
246/246 [==============================] - 2s 10ms/step - loss: 0.2114 -
accuracy: 0.9361 - val_loss: 0.0465 - val_accuracy: 0.9864
Epoch 10/20
246/246 [==============================] - 2s 9ms/step - loss: 0.1949 -
accuracy: 0.9421 - val_loss: 0.0445 - val_accuracy: 0.9892
Epoch 11/20
246/246 [==============================] - 2s 10ms/step - loss: 0.1695 -
accuracy: 0.9479 - val_loss: 0.0403 - val_accuracy: 0.9890
Epoch 12/20
```

```
246/246 [==============================] - 2s 10ms/step - loss: 0.1676 -
accuracy: 0.9497 - val_loss: 0.0355 - val_accuracy: 0.9906
Epoch 13/20
246/246 [==============================] - 2s 10ms/step - loss: 0.1450 -
accuracy: 0.9561 - val_loss: 0.0300 - val_accuracy: 0.9922
Epoch 14/20
246/246 [==============================] - 2s 10ms/step - loss: 0.1444 -
accuracy: 0.9562 - val_loss: 0.0247 - val_accuracy: 0.9943
Epoch 15/20
246/246 [==============================] - 2s 10ms/step - loss: 0.1352 -
accuracy: 0.9599 - val_loss: 0.0261 - val_accuracy: 0.9926
Epoch 16/20
246/246 [==============================] - 2s 10ms/step - loss: 0.1203 -
accuracy: 0.9644 - val_loss: 0.0284 - val_accuracy: 0.9923
Epoch 17/20
246/246 [==============================] - 2s 9ms/step - loss: 0.1191 -
accuracy: 0.9646 - val_loss: 0.0257 - val_accuracy: 0.9940
Epoch 18/20
246/246 [==============================] - 2s 10ms/step - loss: 0.1109 -
accuracy: 0.9680 - val_loss: 0.0221 - val_accuracy: 0.9949
Epoch 19/20
246/246 [==============================] - 2s 10ms/step - loss: 0.1030 -
accuracy: 0.9698 - val_loss: 0.0228 - val_accuracy: 0.9940
Epoch 20/20
246/246 [==============================] - 2s 10ms/step - loss: 0.0995 -
accuracy: 0.9712 - val_loss: 0.0233 - val_accuracy: 0.9948
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_29 (Conv2D)           (None, 30, 30, 64)        1792
_________________________________________________________________
conv2d_30 (Conv2D)           (None, 28, 28, 32)        18464
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 14, 14, 32)        0
_________________________________________________________________
dropout_8 (Dropout)          (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 12, 12, 32)        9248
_________________________________________________________________
conv2d_32 (Conv2D)           (None, 10, 10, 16)        4624
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 5, 5, 16)          0
_________________________________________________________________
dropout_9 (Dropout)          (None, 5, 5, 16)          0
_________________________________________________________________
flatten_8 (Flatten)          (None, 400)               0
_________________________________________________________________
```

```
dense_18 (Dense)             (None, 256)              102656
_________________________________________________________________
dropout_10 (Dropout)         (None, 256)              0
_________________________________________________________________
dense_19 (Dense)             (None, 128)              32896
_________________________________________________________________
dropout_11 (Dropout)         (None, 128)              0
_________________________________________________________________
dense_20 (Dense)             (None, 43)               5547
=================================================================
Total params: 175,227
Trainable params: 175,227
Non-trainable params: 0
_________________________________________________________________
395/395 - 1s - loss: 0.1177 - accuracy: 0.9720

 Test accuracy: 0.9719715118408203
```





[20]:
```
# Save model
# file_path='/content/drive/My Drive/NSU/Traffic_data'
# model_mod2.save(f'{file_path}/final_model.h5')
```

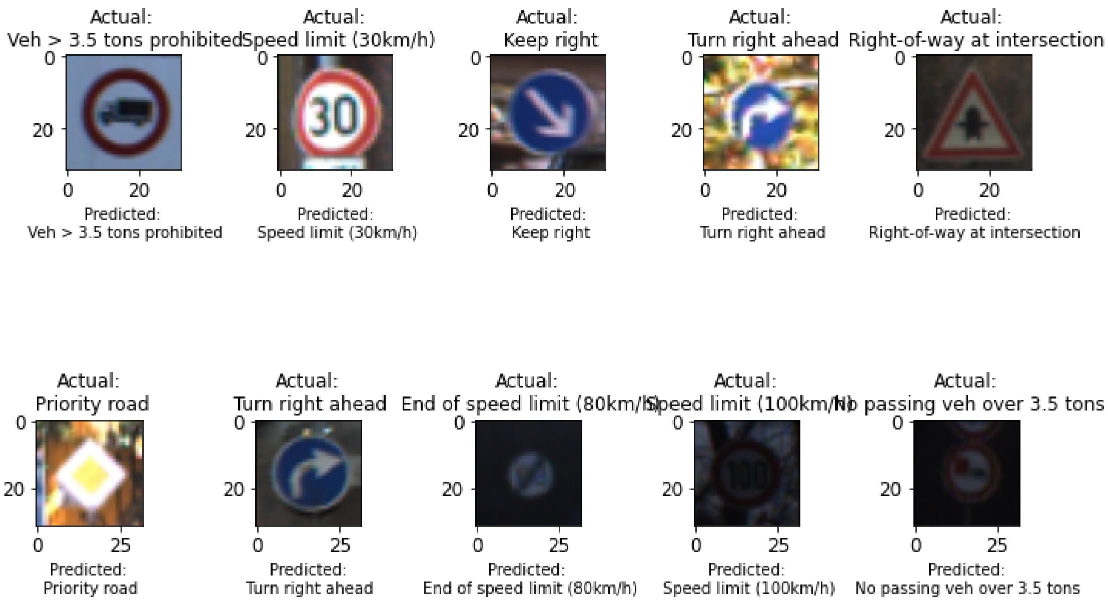
TESTING THE MODEL

[17]:
```
# Define the 43 classes of the road signs

sign_classes={0: 'Speed limit (20km/h)',
 1: 'Speed limit (30km/h)',
 2: 'Speed limit (50km/h)',
 3: 'Speed limit (60km/h)',
 4: 'Speed limit (70km/h)',
 5: 'Speed limit (80km/h)',
 6: 'End of speed limit (80km/h)',
 7: 'Speed limit (100km/h)',
 8: 'Speed limit (120km/h)',
 9: 'No passing',
 10: 'No passing veh over 3.5 tons',
 11: 'Right-of-way at intersection',
 12: 'Priority road',
 13: 'Yield',
 14: 'Stop',
 15: 'No vehicles',
```

```
 16: 'Veh > 3.5 tons prohibited',
 17: 'No entry',
 18: 'General caution',
 19: 'Dangerous curve left',
 20: 'Dangerous curve right',
 21: 'Double curve',
 22: 'Bumpy road',
 23: 'Slippery road',
 24: 'Road narrows on the right',
 25: 'Road work',
 26: 'Traffic signals',
 27: 'Pedestrians',
 28: 'Children crossing',
 29: 'Bicycles crossing',
 30: 'Beware of ice/snow',
 31: 'Wild animals crossing',
 32: 'End speed + passing limits',
 33: 'Turn right ahead',
 34: 'Turn left ahead',
 35: 'Ahead only',
 36: 'Go straight or right',
 37: 'Go straight or left',
 38: 'Keep right',
 39: 'Keep left',
 40: 'Roundabout mandatory',
 41: 'End of no passing',
 42: 'End no passing veh > 3.5 tons'}
```

[18]:
```
# Predict the classes for the first 5 images in the test dataset
pred_first=[sign_classes[i] for i in (np.argmax(i) for i in model_mod2.
 ↪predict(test_data[:5]))]
pred_last=[sign_classes[i] for i in (np.argmax(i) for i in model_mod2.
 ↪predict(test_data[-5:]))]

# Display the first 5 images of the test data and show their actual and␣
 ↪predicted class

fig=plt.figure(figsize=(10,20))
for i in range(5):
  fig.add_subplot(1,5,i+1)
  plt.imshow(test_data[i])
  plt.title(f"Actual:\n {sign_classes[test_labels[i]]}")
  plt.xlabel(f"Predicted:\n {pred_first[i]}")
fig.tight_layout()

# Display the last 5 images of the test data and show their actual and␣
 ↪predicted class
```

```python
fig=plt.figure(figsize=(10,25))
for i in range(5):
  fig.add_subplot(1,5,i+1)
  plt.imshow(test_data[i-5])
  plt.title(f"Actual:\n {sign_classes[test_labels[i-5]]}")
  plt.xlabel(f"Predicted:\n {pred_last[i]}")
fig.tight_layout()
```



[ ]: