Recognition of Sign Language Digits using ConvNets

Assignment No 3

Waqar Kaleem Khan

01-249191-013

Class MS (DS)

Version 2.0(After Changes)

Data Overview:

We will use "sign language digits data set" for this assignment size of this dataset is 32 megabits source of dataset is kaggale. In this dataset there is 2062 sign language digit images. As we all know digit are from 0 to 9 so this dataset have 10 unique signs. This dataset have two folder one name X.npy and other is Y.npy after compiling this example of code we can find that what's in X folder and in Y folder "X = np.load ("X.npy") print(X.shape)"

Y = np.load("Y.npy") print(Y.shape) as in output we can see in the X folder contain 2062 images of 64 x 64 and y folder contain the corresponding labels for the images in X folder.

Key Words:

Convolutional Neural Networks (CNNs), Fully connected layer(FCN), Accuracy(Acc), Layer(Lay)

First implementation:

in the first implementation we create a multilayer CNNs using keras library to classify the gestures of sign language below the code is attached

input_shape=(64,64,1)

model= tensorflow.keras.Sequential()

model.add(Conv2D(32, (3,3),activation='relu', padding='same' , name="conv_1", input_shape=input_shape))

model.add(MaxPooling2D(pool_size=(2,2),name= 'max_pool1'))

model.add(Conv2D(16, (3,3),activation='relu', padding='same' , name="conv_2", input_shape=input_shape))

model.add(MaxPooling2D((2,2),name= 'max_pool2'))

model.add(Flatten())

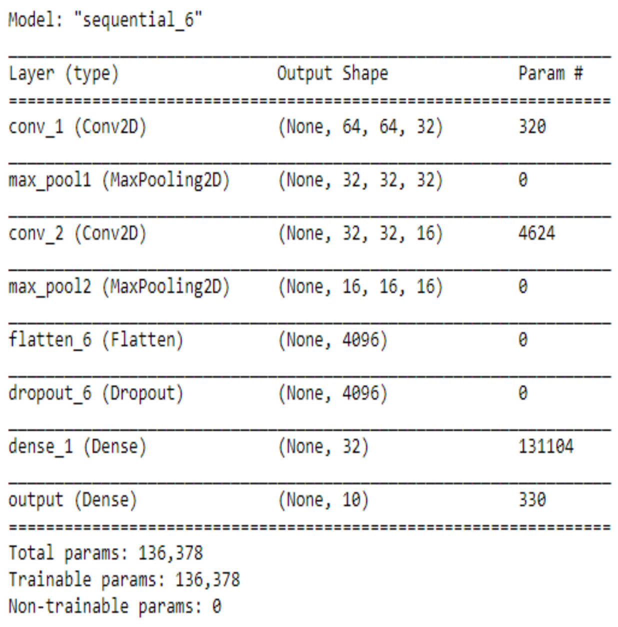
model.add(Dropout(0.5))

model.add(Dense(32,activation='relu' ,name='dense_1'))

model.add(Dense(10,activation='softmax' ,name='output'))

model.summary()

summary of the first implementation is below a screenshot take after compiling the model



```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv_1 (Conv2D)              (None, 64, 64, 32)        320
_________________________________________________________________
max_pool1 (MaxPooling2D)     (None, 32, 32, 32)        0
_________________________________________________________________
conv_2 (Conv2D)              (None, 32, 32, 16)        4624
_________________________________________________________________
max_pool2 (MaxPooling2D)     (None, 16, 16, 16)        0
_________________________________________________________________
flatten_6 (Flatten)          (None, 4096)              0
_________________________________________________________________
dropout_6 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                131104
_________________________________________________________________
output (Dense)               (None, 10)                330
=================================================================
Total params: 136,378
Trainable params: 136,378
Non-trainable params: 0
```

TABLE1
Other parameters use in model

Activation function	Relu
Loss function	categorical_crossentropy
Optimizer	adam

TABLE2

Training and Testing Dataset

Test Size	Train Size	Random State
30	70	30

Table3

Results after First Implementation

Train Acc	Test Acc
71.52	71.74

Second Implantation (Changing Neuron):

In the second implantation we change neuron randomly and add some layers of neuron the below to see how the neuron affect the accuracy or performance of the model in the first implantation the number of neurons are 32 and 16 in CNNs and 32 in Fully connected layers. Accuracy of model changes when the number of neuron change in every iteration are given below

Table4

Train Test Accuracy with Different Layers

CNN Lay 1	CNN Lay 2	CNN Lay 3	FCL Lay 1	FCL Lay 2	FCL Lay 3	Train Acc	Test Acc
32	16	0	32	0	0	71.5	71.7
64	64	32	64	32	0	78.7	78.6
128	64	64	128	64	32	83.9	82.5
256	128	62	128	64	64	75.8	79.8
512	128	64	512	128	64	86.2	85.1

Conclusion (Changing Numbers Neuron):

Testing the model with different neuron the highest test accuracy is on the (512, 128, 64) CNNs neuron and (512, 128, 64) Dense which is 85.1 as shown in the above tables and the lowest testing accuracy of the model is on (32, 16) CNNs neuron and (32) dense neuron

Which is 71.7 as show in the above table.

Third Implementation (Changing Batch Size):

First batch size was 100 then we change batch size to 150,256,350, and 500 with the model which has high test accuracy to see the results that how the batch size impact the accuracy of the model as shown in the below table.

Table5

Train Test Accuracy with Different Batchsize

Batch size	Train Acc	Test Acc
100	86.2	85.1
150	82.4	81.7
250	79.6	76.9
300	74.8	72.7
50	95.5	92.4

Conclusion (Batch Size Testing):

Best number of batch size for our model is 50 as we increase the batch size of the model that affect the accuracy of the model as we can see in the above table that the accuracy of our model on batch size 100 is 85.1 and on 300 it decreases to the 72.7 as in first implantation our batch size is 100 and when further more decrease it to 50 the accuracy to 92.4 of our increases. With all the above different batch size the activation function used is relu.

Fourth Implementation (Training and Test Size):

In the first implementation we set our training size to 70 percent and test size to 30 percent we get the highest Accuracy of 91.4 with the batch size of 50 as we increase the training size by 10 percent and decrease the testing size by 10 percent which make our tanning size 80 percent and testing size 20 percent we get the accuracy of 89.6 training accuracy and 88.3 testing accuracy.

While using test size 30 and train size 70 I have restart the kernel two times and re compile the model again but the accuracy is more than when we have test size 20% and train size 80 %.

Fifth Implementation (Changing Optimizer):

We use two commonly use optimizers in our model to see the impact of optimizers on the model Accuracy first one is adam and the second one is sgd.

Adam optimizer maintain a learning rate for each network weight.

Sgd is an optimizer which maintain a single learning rate for all weights and updates and the learning rate does not change during training,

The batch size use with both optimizers are 50.

Table6
Train Test Accuracy with Different Optimizer

Optimizer	Test size	Train size	Train Acc	Test Acc
Adam	70%	30%	95.5	92.4
Adam	80%	20%	89.6	88.3
Sgd	70%	30%	11.4	6.8
Sgd	80%	20%	12.6	7.0

Sixth Implantation (Changing Activation Function):

In the first iteration we use relu activation function in both CNNs layers and Fully connected layers the accuracy of testing and training dataset are below in the table. In the sixth iteration we change the activation function to tanh with all layers and after that we change CNNs layer to relu and the fully connected layer to tanh the accuracy of the model is below.

The test size is 30 percent and train size is 70 percent in this iteration

TABLE7
Test Train Accuracy with Different Activation Function

CNNs	FCL	Traint Acc	Test Acc
relu	relu	95.5	92.4
Tanh	tanh	88.3	79.6
Relu	tanh	83.0	81.0

Conclusion (Changing Activation Functions):

Using relu function in CNNs layers and fully connected layers will give us the highest accuracy.

Seventh Implementation (Testing on Different Number of Layers):

For checking the impact of the layers on the accuracy we have use different combination of layers in the CNNs and Fully connected layers as you can see how the combination of different layer effect the accuracy of the model

TABLE8
Test Train Accuracy with Different Number of CNNs and FC Layers

CNNs layers	FC layers	Training Acc	Testing Acc
1	1	81.2	73.5
2	2	76.9	78.3
3	3	95	92
3	2	92.5	91.4
4	2	86.7	90.6
4	3	87.9	89.8

Conclusion (Changing Layers):

Using different combination of layers as we can see in the above table that a combination of 3x3 layers highest accuracy on the test data set with 2 max polling activation function used is relu the training dataset is 70% and test dataset is 30 % with the 3 fully connected layers

Eight Implementation (Channing Dropout):

Dropout in the NN is a regularization technique use to prevent neural networks from over fitting in the dropout randomly selected neuron are ignored during training they are dropout randomly. This means that the selected neuron contribution to the network is temporarily removed while doing forward and in the back propagation not any weight updation are apply to the selected neuron.
In the first implantation our dropout value is 0.5 After that in the dropout changing iteration we change drop out from 0.1 to 0.5 as we can see after the results the heights Test Accuracy are given on 0.5 dropout size and after that the good testing accuracy result is on 0.2 but as we can see that the best training accuracy is on the 0.2 dropout size which is 96.6 and the lowest test accuracy is on 0.1 which is 87.5 so as we can see that changing drop out size we will get different results so there is no fixed dropout size which will give a best results it matters from problem to problem and when testing your model change it in some iteration and find a best results for your model

TABLE9

Train Test Accuracy with Different Dropout Size

Dropout	Training Acc	Testing Acc
0.5	95.5	92.4
0.4	93.9	88.8
0.3	95.7	88.6
0.2	96.6	91.2
0.1	95.2	87.5

Ninth Implementation (Changing Filters Size):

In the first iteration we use 3x3 filters with the 3 layers of CNNs and 3 layers of fully connected layers now In the iteration we will change the filters layer to 1x1 and 4x4 to see how this affect the accuracy of the model

TABLE10

Train Test Accuracy with Different Filter Size

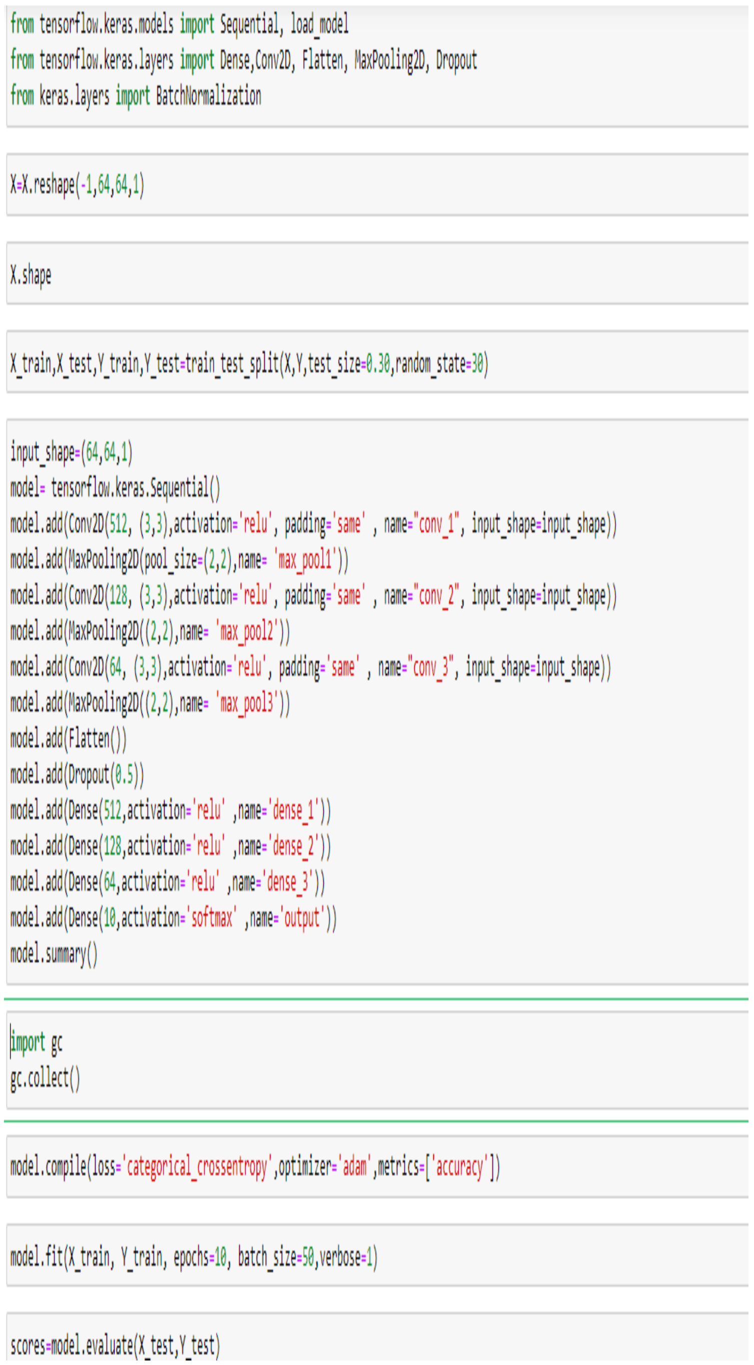
Filters	Layer1	Layer2	Layer3	Train Acc	Test Acc
3x3	3x3	3x3	3x3	95.5	92.4
2x2	2x2	2x2	2x2	86.2	84.6
4x4 3x3 2x2	4x4	3x3	2x2	93.2	89.9

Conclusion (Changing Filter Size):

In the Eight iteration we use different combination of filters in the first two iteration we use same filters in all three layers and in the third iteration we use 3 different filters but as we can see that the highest test accuracy we get on the 3x3 filters use in all layers and also we have seen that how the filters can affect the accuracy of a model all the filers use in the above model is taken randomly.

Selecting Final Model with Best Results:

After doing all above eight implementation the model which give highest test accuracy code is below.



```
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Dense,Conv2D, Flatten, MaxPooling2D, Dropout
from keras.layers import BatchNormalization
```

```
X=X.reshape(-1,64,64,1)
```

```
X.shape
```

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.30,random_state=30)
```

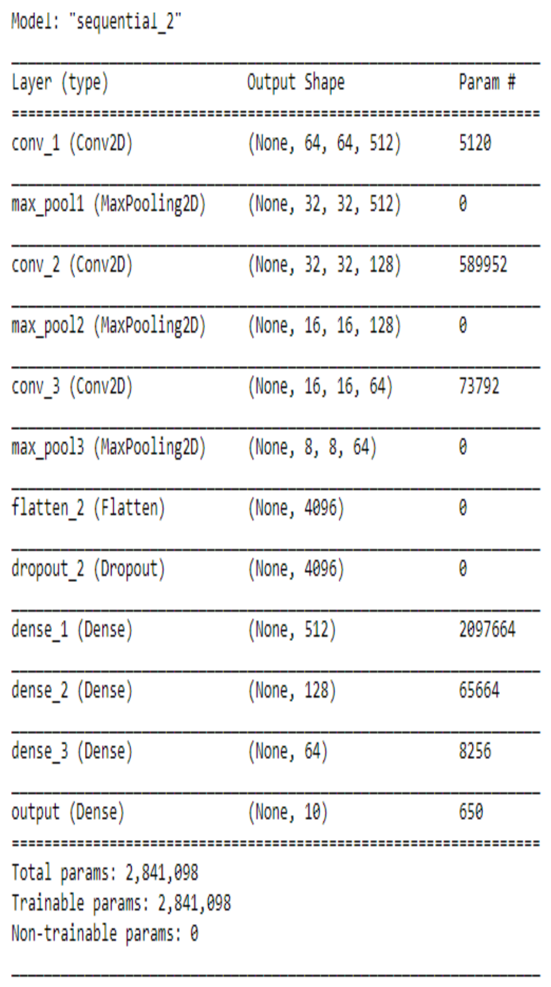
```
input_shape=(64,64,1)
model= tensorflow.keras.Sequential()
model.add(Conv2D(512, (3,3),activation='relu', padding='same' , name="conv_1", input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2,2),name= 'max_pool1'))
model.add(Conv2D(128, (3,3),activation='relu', padding='same' , name="conv_2", input_shape=input_shape))
model.add(MaxPooling2D((2,2),name= 'max_pool2'))
model.add(Conv2D(64, (3,3),activation='relu', padding='same' , name="conv_3", input_shape=input_shape))
model.add(MaxPooling2D((2,2),name= 'max_pool3'))
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(512,activation='relu' ,name='dense_1'))
model.add(Dense(128,activation='relu' ,name='dense_2'))
model.add(Dense(64,activation='relu' ,name='dense_3'))
model.add(Dense(10,activation='softmax' ,name='output'))
model.summary()
```

```
import gc
gc.collect()
```

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

```
model.fit(X_train, Y_train, epochs=10, batch_size=50,verbose=1)
```

```
scores=model.evaluate(X_test,Y_test)
```



```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv_1 (Conv2D)              (None, 64, 64, 512)       5120
_________________________________________________________________
max_pool1 (MaxPooling2D)     (None, 32, 32, 512)       0
_________________________________________________________________
conv_2 (Conv2D)              (None, 32, 32, 128)       589952
_________________________________________________________________
max_pool2 (MaxPooling2D)     (None, 16, 16, 128)       0
_________________________________________________________________
conv_3 (Conv2D)              (None, 16, 16, 64)        73792
_________________________________________________________________
max_pool3 (MaxPooling2D)     (None, 8, 8, 64)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 4096)              0
_________________________________________________________________
dropout_2 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               2097664
_________________________________________________________________
dense_2 (Dense)              (None, 128)               65664
_________________________________________________________________
dense_3 (Dense)              (None, 64)                8256
_________________________________________________________________
output (Dense)               (None, 10)                650
=================================================================
Total params: 2,841,098
Trainable params: 2,841,098
Non-trainable params: 0
_________________________________________________________________
```

TABLE11
Final parameters and Results of the model

Activation Function	Relu
CNNs Layers	3
FC Layers	3
Epochs	10
Batch size	50
Drop out	0.5
Test Acc	92.4
Train Acc	95.5

Conclusion:

After running all the test changing the hyper parameters the highest accuracy given by the model the final code and the results are above the highest accuracy we can achieve is 92.4 also we have seen the results and impact of the accuracy while changing the different hyper parameters one by one we have seen different results. As we can see that the test we run on the model with different parameters include activation functions, layers, neuron, and filters of convolutional networks, dropout size, train test dataset and batch size. As we know from the above experiments that if we increase the layers and number of neuron in both CNNs and FCL layers and decrease the size of the batch size and use 'Relu' activation function and keep the size of 'dropout' to 0.5 we can achieve the highest accuracy of the model.

Refrences:

1 https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/

2 https://keras.io/optimizers/

3 https://www.kaggle.com/kanncaa1/deep-learning-tutorial-for-beginners

4 https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

5https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/