AD18511 – DEEP LEARNING LABORATORY

DATE:

EX.NO: 11 **IMPLEMENTATION OF GOOGLENET ARCHITECTURE.**

AIM:

To implement the application using Google-net architecture.

DESCRIPTION:

GoogLeNet is a convolutional neural network that is 22 layers deep.

GoogLeNet, also known as Inception v1, is a deep convolutional neural network architecture that was introduced in 2014. It won the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC-2014) with a classification performance of 92.3%, which was a significant improvement over previous state-of-the-art models.

The success of GoogLeNet was attributed to a number of factors, including:

The use of inception modules, which are a type of building block that combines convolutions of different kernel sizes in parallel. This allows the network to learn features at different scales, which is important for image classification.

The use of 1x1 convolutions to reduce the number of filter channels in the network, which makes it more efficient and reduces the risk of overfitting.

The use of global average pooling at the end of the network, which allows the network to learn features that are invariant to translation and scale.

PROGRAM:

```
import tensorflow as tf
from tensorflow.keras.layers import Input,
Conv2D,MaxPooling2D,AveragePooling2D,concatenate,Flatten,Dense,Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical

(x_train,y_train),(x_test,y_test)=cifar10.load_data()
x_train=x_train/255.0
x_test=x_test/255.0
y_train=to_categorical(y_train,num_classes=10)
y_test=to_categorical(y_test,num_classes=10)

def inception_module(x,filters):
    conv1x1=Conv2D(filters[0],(1,1),padding='same',activation='relu')(x)
    conv3x3=Conv2D(filters[1],(3,3),padding='same',activation='relu')(x)
    conv5x5=Conv2D(filters[2],(5,5),padding='same',activation='relu')(x)
    maxpool=MaxPooling2D((3,3),strides=(1,1),padding='same')(x)
    pool_conv=Conv2D(filters[3],(1,1),padding='same',activation='relu')(maxpool)
    return concatenate([conv1x1,conv3x3,conv5x5,pool_conv],axis=-1)
```

```
input_layer=Input(shape=(32,32,3))
x=Conv2D(64,(7,7),padding='same',activation='relu',strides=(2,2))(input_layer)
x=MaxPooling2D((3,3),strides=(2,2))(x)
x=Conv2D(64,(1,1),padding='same',activation='relu')(x)
x=Conv2D(192,(3,3),padding='same',activation='relu')(x)
x=MaxPooling2D((3,3),strides=(2,2))(x)
x=inception_module(x,[64,128,32,32])
x=inception_module(x,[128,192,96,64])
x=MaxPooling2D((3,3),strides=(2,2))(x)

x=Flatten()(x)
x=Dropout(0.4)(x)
x=Dense(256,activation='relu')(x)
output_layer=Dense(10,activation='softmax')(x)

model=Model(inputs=input_layer,outputs=output_layer)

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

history=model.fit(x_train,y_train,epochs=10,batch_size=128,validation_data=(x_test,y_test))
```

OUTPUT:

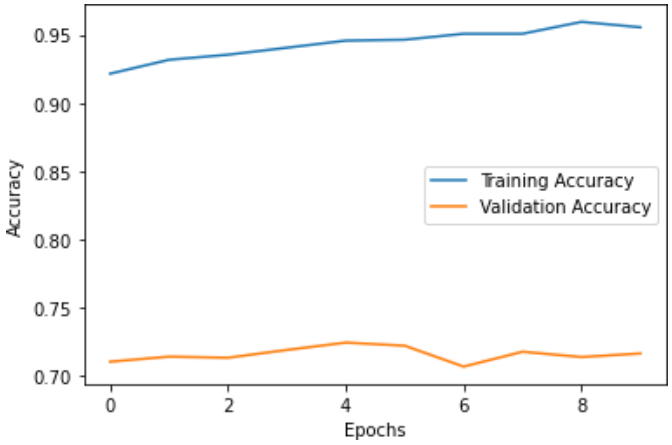
```
l_accuracy: 0.7068
Epoch 8/10
391/391 [==============================] - 24s 61ms/step - loss: 0.1421 - accuracy: 0.9507 - val_loss: 1.3139 -
val_accuracy: 0.7177
Epoch 9/10
391/391 [==============================] - 23s 59ms/step - loss: 0.1206 - accuracy: 0.9594 - val_loss: 1.5549 -
val_accuracy: 0.7138
Epoch 10/10
391/391 [==============================] - 23s 60ms/step - loss: 0.1277 - accuracy: 0.9554 - val_loss: 1.5585 -
val_accuracy: 0.7164
```

```
#Plot the training accuracy graph
import matplotlib.pyplot as plt
import numpy as np
plt.plot(history.history['accuracy'],label='Training Accuracy')
plt.plot(history.history['val_accuracy'],label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

#Generate predictions
predictions=model.predict(x_test)
predicted_labels=np.argmax(predictions,axis=1)
true_labels=np.argmax(y_test,axis=1)

#Print classification report
from sklearn.metrics import classification_report
print(classification_report(true_labels,predicted_labels))
```

OUTPUT:



313/313 [==============================] - 1s 4ms/step

	precision	recall	f1-score	support
0	0.76	0.79	0.78	1000
1	0.85	0.78	0.81	1000
2	0.60	0.61	0.61	1000
3	0.54	0.53	0.53	1000
4	0.63	0.74	0.68	1000
5	0.67	0.55	0.61	1000
6	0.77	0.77	0.77	1000
7	0.73	0.80	0.76	1000
8	0.85	0.81	0.83	1000
9	0.78	0.79	0.78	1000
accuracy			0.72	10000
macro avg	0.72	0.72	0.72	10000
weighted avg	0.72	0.72	0.72	10000

RESULT:

Thus the application using googlenet architecture has been implemented successfully.