Name: Prekshita vasudeo patil

registration No.: 20MAI0073

Assignment-4

Task-2 ¶

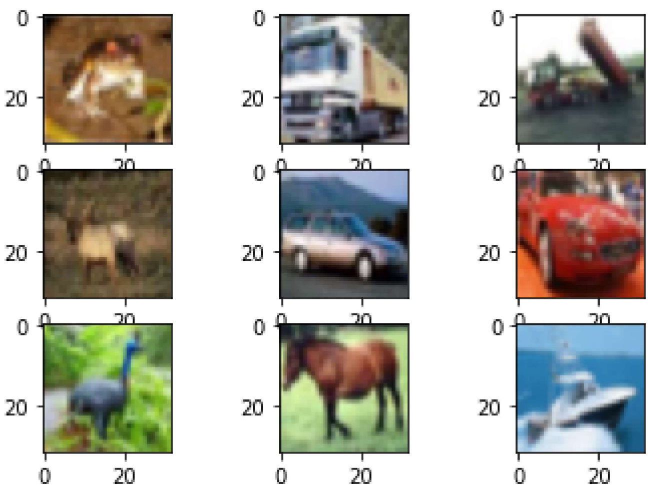
In [1]:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import keras as k
import tensorflow as tf
from keras.datasets import cifar10
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.model_selection import train_test_split
```

In [2]:

```
(trainX, trainy), (testX, testy) = cifar10.load_data()
print('Train: X=%s, y=%s' % (trainX.shape, trainy.shape))
print('Test: X=%s, y=%s' % (testX.shape, testy.shape))
for i in range(9):
    plt.subplot(330 + 1 + i)
    plt.imshow(trainX[i])
plt.show()
```

```
Train: X=(50000, 32, 32, 3), y=(50000, 1)
Test: X=(10000, 32, 32, 3), y=(10000, 1)
```



In [3]:

```
from keras.utils import to_categorical
trainy = to_categorical(trainy)
testy = to_categorical(testy)
```

In [4]:

```
from keras.layers import Conv2D,MaxPooling2D,Flatten,Dense,BatchNormalization
from keras.models import Sequential
from keras.optimizers import SGD
from keras.preprocessing.image import ImageDataGenerator
```

In [5]:

```
datagen = ImageDataGenerator(width_shift_range=0.1, height_shift_range=0.1, hori
```

In [6]:

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform',
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform',
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform',
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform',
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform',
model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform',
model.add(MaxPooling2D((2, 2)))
model.add(BatchNormalization()) # Adding Batch Normalization
model.add(Flatten())
model.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
model.add(Dense(10, activation='softmax'))
# compile model
opt = SGD(lr=0.001, momentum=0.9)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        896       
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 32)        9248      
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 64)        18496     
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 64)        36928     
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 64)          0         
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 128)         73856     
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 128)         147584    
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 128)         0         
_________________________________________________________________
batch_normalization (BatchNo (None, 4, 4, 128)         512       
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0         
_________________________________________________________________
dense (Dense)                (None, 128)               262272    
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290      
=================================================================
Total params: 551,082
Trainable params: 550,826
Non-trainable params: 256
_________________________________________________________________
```

In [7]:

```
it_train = datagen.flow(trainX, trainy, batch_size=64)
```

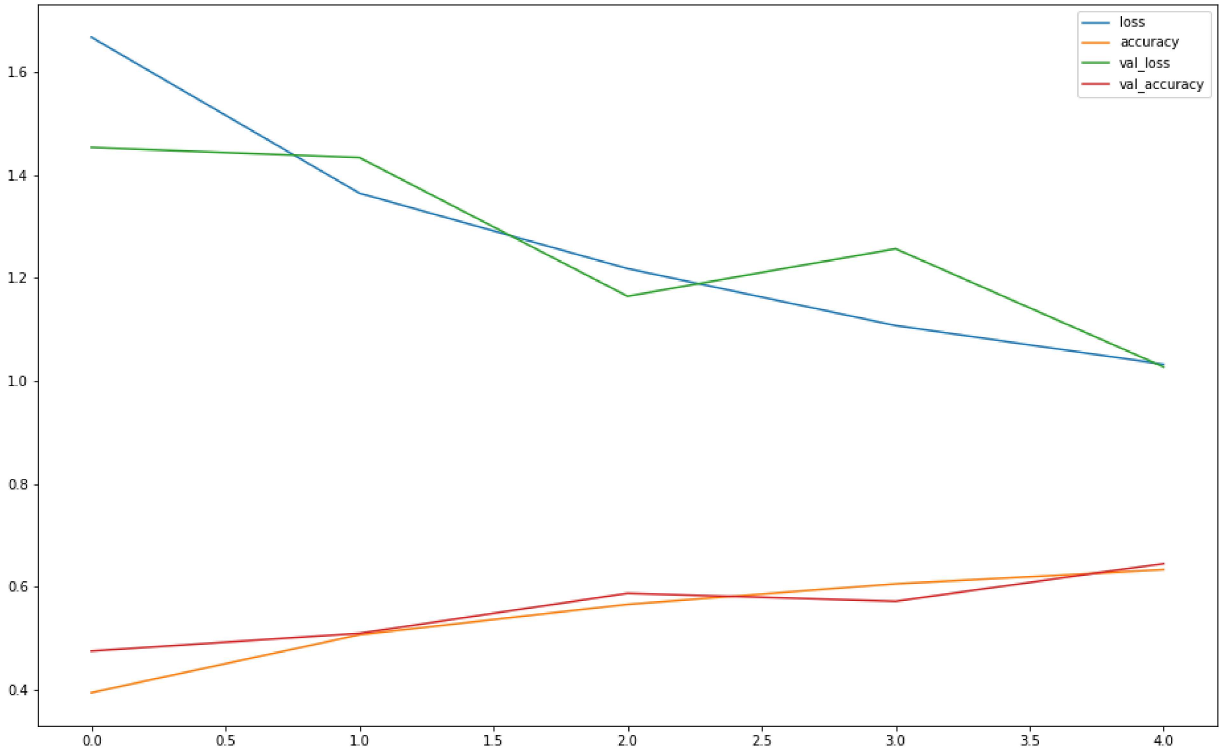
In [8]:
```
    steps = int(trainX.shape[0] / 64)
    history = model.fit_generator(it_train, steps_per_epoch=steps, epochs=5, vali
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.
py:1844: UserWarning: `Model.fit_generator` is deprecated and will be removed i
n a future version. Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
```

In [9]:
```
history = pd.DataFrame(history.history)
```

In [10]:
```
history.plot.line(figsize=(16,10))
```

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff56ba54410>
```



In [16]:
```
ypred = np.argmax(model.predict(testX),axis=1)
```

In [17]:
```
ypred
```

Out[17]:
```
array([3, 1, 8, ..., 5, 1, 7])
```

In [18]:
```
testty = np.argmax(testy,axis=1)
```

In [19]:
```
testty
```

Out[19]:
```
array([3, 8, 8, ..., 5, 1, 7])
```

In [20]:

```
accuracy_score(ypred,testty)
```

Out[20]:

```
0.6449
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Name: Prekshita vasudeo patil
registration No.: 20MAI0073
Assignment-4
Task-1
Link:https://github.com/prekshita19/DL-Assignments/tree/main/Assignment-4

In [ ]:

```
import pandas as pd
import numpy as np
import tensorflow as tf
import keras
import matplotlib.pyplot as plt
import seaborn as sns
import scipy
from sklearn.model_selection import train_test_split
```

In [2]:

```
(xtrain,ytrain),(xtest,ytest) = keras.datasets.cifar10.load_data()
classes = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse
```

In [3]:

```
plt.imshow(xtrain[0])
plt.title(classes[ytrain[0][0]],)
plt.axis(False)
plt.show()
```

In [4]:

```
#  selecting 50% less data from xtrain
(xtrain_new_50,xtest_new_50,ytrain_new_50,ytest_new_50) =train_test_split(xtrain,
xtrain_1,xtest_1,ytrain_1,ytest_1 = train_test_split(xtrain_new_50,ytrain_new_50,
print("Xtrain orignal :- ",xtrain.shape)
print("Xtrain 50% selected from xtrain:-",xtrain_new_50.shape)
print("70% selected from that 50% training :- ",xtrain_1.shape)
```

```
Xtrain orignal :-  (50000, 32, 32, 3)
Xtrain 50% selected from xtrain:- (25000, 32, 32, 3)
70% selected from that 50% training :-  (17500, 32, 32, 3)
```

In [5]:

```
from keras.models import Sequential
from keras.layers import Conv2D,Activation,BatchNormalization,MaxPooling2D,Dense,
```

In [6]:

```
from keras.utils import to_categorical

ytrain = to_categorical(ytrain,10)
ytrain_new_50 = to_categorical(ytrain_new_50,10)
ytrain_1 = to_categorical(ytrain_1,10)

ytest = to_categorical(ytest,10)
ytest_new_50 = to_categorical(ytest_new_50,10)
ytest_1 = to_categorical(ytest_1,10)
```

In [7]:

```
xtrain_1[0].shape
```

Out[7]:

```
(32, 32, 3)
```

In [8]:
```
AlexNet = Sequential()

#1st Convolutional Layer
AlexNet.add(Conv2D(filters=96, input_shape=xtrain_1[0].shape, kernel_size=(11,11)
AlexNet.add(BatchNormalization())
AlexNet.add(Activation('relu'))

# 1st Maxpooling Layer
AlexNet.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'))

#2nd Convolutional Layer
AlexNet.add(Conv2D(filters=256, kernel_size=(5, 5), strides=(1,1), padding='same'
AlexNet.add(BatchNormalization())
AlexNet.add(Activation('relu'))

# 2nd Maxpooling Layer
AlexNet.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'))

#3rd Convolutional Layer
AlexNet.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), padding='same')
AlexNet.add(BatchNormalization())
AlexNet.add(Activation('relu'))

#4th Convolutional Layer
AlexNet.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), padding='same')
AlexNet.add(BatchNormalization())
AlexNet.add(Activation('relu'))

#5th Convolutional Layer
AlexNet.add(Conv2D(filters=256, kernel_size=(3,3), strides=(1,1), padding='same')
AlexNet.add(BatchNormalization())
AlexNet.add(Activation('relu'))

# 3rd Maxpooling Layer
AlexNet.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'))

#Passing it to a Fully Connected layer
AlexNet.add(Flatten())

# 1st Fully Connected Layer
AlexNet.add(Dense(4096, input_shape=(32,32,3,)))
AlexNet.add(BatchNormalization())
AlexNet.add(Activation('relu'))

# Add Dropout to prevent overfitting
AlexNet.add(Dropout(0.4))

#2nd Fully Connected Layer
AlexNet.add(Dense(4096))
AlexNet.add(BatchNormalization())
AlexNet.add(Activation('relu'))
#Add Dropout
AlexNet.add(Dropout(0.4))

#3rd Fully Connected Layer
# AlexNet.add(Dense(1000))
```

```
AlexNet.add(BatchNormalization())
AlexNet.add(Activation('relu'))
#Add Dropout
AlexNet.add(Dropout(0.4))

#Output Layer
AlexNet.add(Dense(10))
AlexNet.add(BatchNormalization())
AlexNet.add(Activation('softmax'))

#Model Summary
AlexNet.summary()
# https://cs231n.github.io/transfer-learning/
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 8, 8, 96)          34944     
_________________________________________________________________
batch_normalization (BatchNo (None, 8, 8, 96)          384       
_________________________________________________________________
activation (Activation)      (None, 8, 8, 96)          0         
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 4, 4, 96)          0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 4, 4, 256)         614656    
_________________________________________________________________
batch_normalization_1 (Batch (None, 4, 4, 256)         1024      
_________________________________________________________________
activation_1 (Activation)    (None, 4, 4, 256)         0         
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 2, 2, 256)         0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 2, 2, 384)         885120    
_________________________________________________________________
batch_normalization_2 (Batch (None, 2, 2, 384)         1536      
_________________________________________________________________
activation_2 (Activation)    (None, 2, 2, 384)         0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 2, 2, 384)         1327488   
_________________________________________________________________
batch_normalization_3 (Batch (None, 2, 2, 384)         1536      
_________________________________________________________________
activation_3 (Activation)    (None, 2, 2, 384)         0         
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 2, 2, 256)         884992    
_________________________________________________________________
batch_normalization_4 (Batch (None, 2, 2, 256)         1024      
_________________________________________________________________
activation_4 (Activation)    (None, 2, 2, 256)         0         
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 1, 1, 256)         0         
_________________________________________________________________
flatten (Flatten)            (None, 256)               0         
_________________________________________________________________
dense (Dense)                (None, 4096)              1052672   
```

```
_________________________________________________________________
batch_normalization_5 (Batch (None, 4096)              16384     
_________________________________________________________________
activation_5 (Activation)    (None, 4096)              0         
_________________________________________________________________
dropout (Dropout)            (None, 4096)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              16781312  
_________________________________________________________________
batch_normalization_6 (Batch (None, 4096)              16384     
_________________________________________________________________
activation_6 (Activation)    (None, 4096)              0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 4096)              0         
_________________________________________________________________
batch_normalization_7 (Batch (None, 4096)              16384     
_________________________________________________________________
activation_7 (Activation)    (None, 4096)              0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 4096)              0         
_________________________________________________________________
dense_2 (Dense)              (None, 10)                40970     
_________________________________________________________________
batch_normalization_8 (Batch (None, 10)                40        
_________________________________________________________________
activation_8 (Activation)    (None, 10)                0         
=================================================================
Total params: 21,676,850
Trainable params: 21,649,502
Non-trainable params: 27,348
_________________________________________________________________
```

In [9]:

```
AlexNet.compile(loss = 'categorical_crossentropy', optimizer= 'adam', metrics=['a
```

In [10]:

```
# checkpoint = ModelCheckpoint('AlexNet.h5',save_best_only=True, monitor='val_acc
history =AlexNet.fit(xtrain_1,ytrain_1,epochs=10,batch_size=32)
```

```
Epoch 1/10
547/547 [==============================] - 11s 13ms/step - loss: 1.9141 - accur
acy: 0.3003
Epoch 2/10
547/547 [==============================] - 7s 12ms/step - loss: 1.5539 - accura
cy: 0.4400
Epoch 3/10
547/547 [==============================] - 7s 12ms/step - loss: 1.4151 - accura
cy: 0.5054
Epoch 4/10
547/547 [==============================] - 7s 12ms/step - loss: 1.3250 - accura
cy: 0.5396
Epoch 5/10
547/547 [==============================] - 7s 12ms/step - loss: 1.2156 - accura
cy: 0.5769
Epoch 6/10
547/547 [==============================] - 7s 12ms/step - loss: 1.1364 - accura
cy: 0.6088
Epoch 7/10
547/547 [==============================] - 7s 12ms/step - loss: 1.0408 - accura
cy: 0.6458
Epoch 8/10
547/547 [==============================] - 7s 12ms/step - loss: 0.9479 - accura
cy: 0.6789
Epoch 9/10
547/547 [==============================] - 7s 12ms/step - loss: 0.8661 - accura
cy: 0.7111
Epoch 10/10
547/547 [==============================] - 7s 12ms/step - loss: 0.7642 - accura
cy: 0.7450
```

In [11]:

```
history = pd.DataFrame(history.history)
```

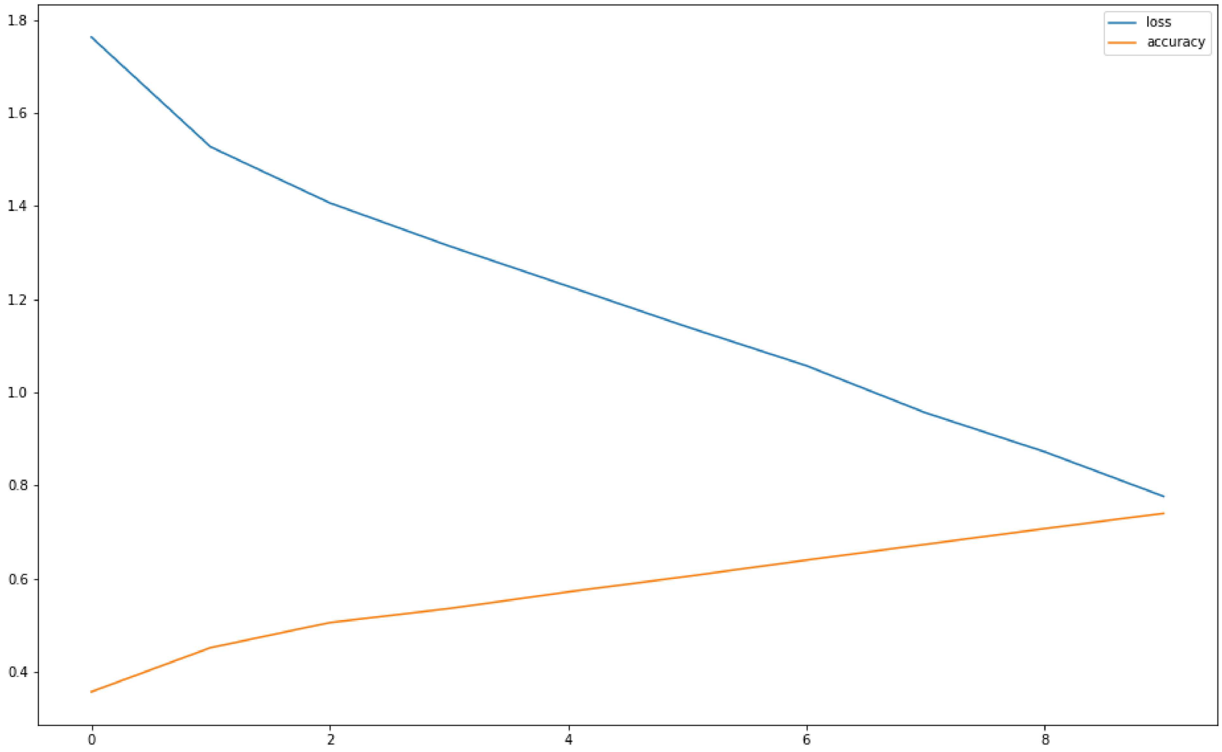
In [12]:
```
history
```

Out[12]:

	loss	accuracy
0	1.762730	0.356857
1	1.527228	0.451486
2	1.406960	0.505371
3	1.314531	0.535429
4	1.227925	0.571257
5	1.140598	0.604514
6	1.057282	0.639543
7	0.956235	0.673429
8	0.872383	0.707143
9	0.776495	0.740057

In [13]:
```
history.plot.line(figsize=(16,10),)
```

Out[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0bb612d510>
```



In [14]:

```
ypred = np.argmax(AlexNet.predict(xtest_1),axis=1)

ytrain = np.argmax(ytrain,axis=1)
ytrain_new_50 = np.argmax(ytrain_new_50,axis=1)
ytrain_1 = np.argmax(ytrain_1,axis=1)

ytest = np.argmax(ytest,axis=1)
ytest_new_50 = np.argmax(ytest_new_50,axis=1)
ytest_1 = np.argmax(ytest_1,axis=1)
```

In [15]:

```
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
accuracy_score(ypred,ytest_1)
```

Out[15]:

```
0.4696
```

In [16]:

```
ytrain = to_categorical(ytrain,10)
ytrain_new_50 = to_categorical(ytrain_new_50,10)
ytrain_1 = to_categorical(ytrain_1,10)

ytest = to_categorical(ytest,10)
ytest_new_50 = to_categorical(ytest_new_50,10)
ytest_1 = to_categorical(ytest_1,10)
```

In [17]:
```
history =AlexNet.fit(xtrain,ytrain,epochs=10,batch_size=32)
```

```
Epoch 1/10
1563/1563 [==============================] - 19s 12ms/step - loss: 1.0515 - acc
uracy: 0.6403
Epoch 2/10
1563/1563 [==============================] - 19s 12ms/step - loss: 0.9427 - acc
uracy: 0.6777
Epoch 3/10
1563/1563 [==============================] - 19s 12ms/step - loss: 0.8495 - acc
uracy: 0.7101
Epoch 4/10
1563/1563 [==============================] - 19s 12ms/step - loss: 0.7463 - acc
uracy: 0.7505
Epoch 5/10
1563/1563 [==============================] - 19s 12ms/step - loss: 0.6588 - acc
uracy: 0.7788
Epoch 6/10
1563/1563 [==============================] - 19s 12ms/step - loss: 0.5772 - acc
uracy: 0.8105
Epoch 7/10
1563/1563 [==============================] - 19s 12ms/step - loss: 0.4984 - acc
uracy: 0.8361
Epoch 8/10
1563/1563 [==============================] - 19s 12ms/step - loss: 0.4401 - acc
uracy: 0.8574
Epoch 9/10
1563/1563 [==============================] - 19s 12ms/step - loss: 0.3852 - acc
uracy: 0.8753
Epoch 10/10
1563/1563 [==============================] - 19s 12ms/step - loss: 0.3414 - acc
uracy: 0.8905
```

In [18]:
```
history = pd.DataFrame(history.history)
```

In [19]:
```
history
```

Out[19]:

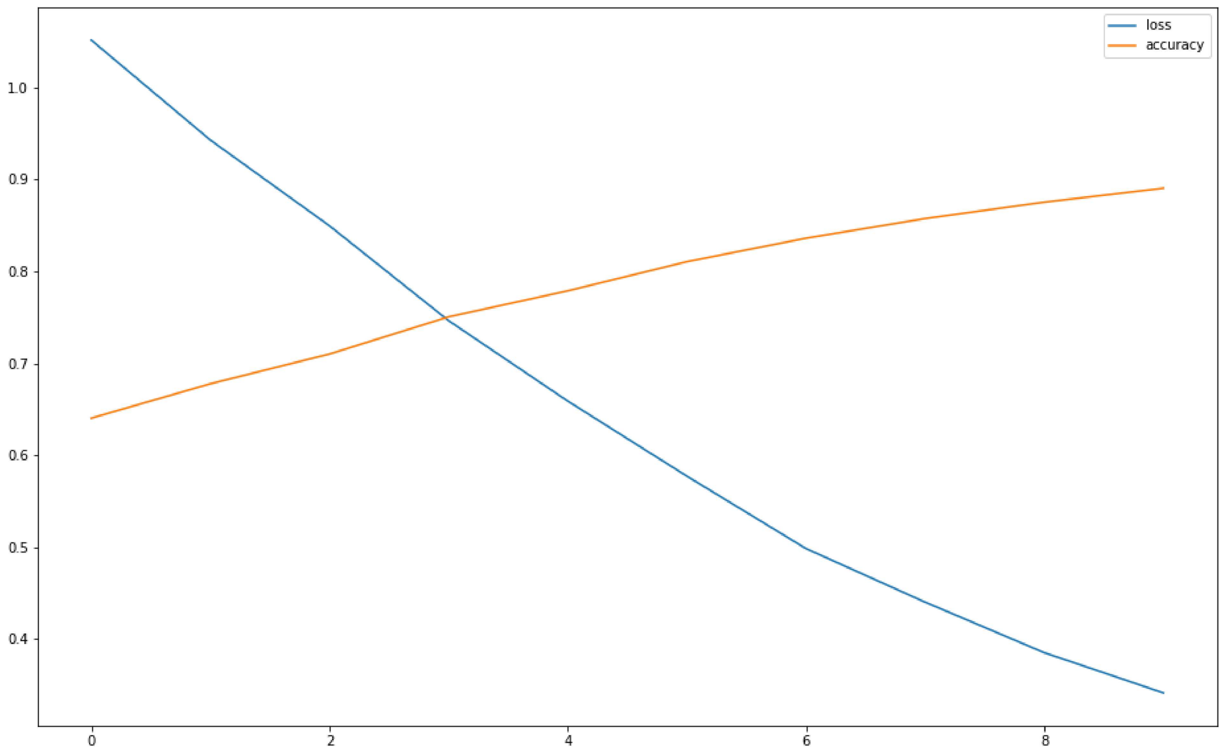
	loss	accuracy
0	1.051510	0.64032
1	0.942745	0.67774
2	0.849459	0.71012
3	0.746287	0.75052
4	0.658829	0.77884
5	0.577167	0.81052
6	0.498385	0.83610
7	0.440125	0.85744
8	0.385194	0.87528
9	0.341448	0.89046

In [20]:

```
history.plot.line(figsize=(16,10))
```

Out[20]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0baf790510>
```



In [21]:

```
ypred = np.argmax(AlexNet.predict(xtest),axis=1)

ytrain = np.argmax(ytrain,axis=1)
ytrain_new_50 = np.argmax(ytrain_new_50,axis=1)
ytrain_1 = np.argmax(ytrain_1,axis=1)

ytest = np.argmax(ytest,axis=1)
ytest_new_50 = np.argmax(ytest_new_50,axis=1)
ytest_1 = np.argmax(ytest_1,axis=1)
```

In [22]:

```
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
accuracy_score(ypred,ytest)
```

Out[22]:

```
0.6164
```