Mujtaba Shahid Faizi

BSCS-5A

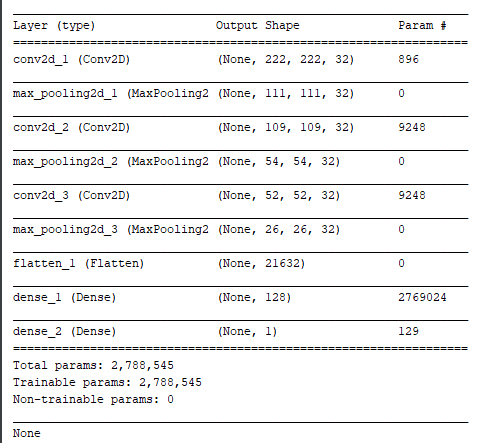
#131818

Lab 8 of Computer Vision

**CODE:**

**from** keras.preprocessing.image **import** ImageDataGenerator  
**from** keras.models **import** Sequential  
**from** keras.layers **import** Dense, Dropout, Flatten  
**from** keras.layers **import** Conv2D, MaxPooling2D  
**import** keras  
**import** matplotlib.pyplot **as** plt  
**from** keras.models **import** model\_from\_json  
**import** numpy  
**import** cv2  
  
*# Initialising the CNN*model = Sequential()  
model.add(Conv2D(32, (3, 3), input\_shape = (224, 224, 3), activation = **'relu'**))  
model.add(MaxPooling2D(pool\_size = (2, 2)))  
model.add(Conv2D(32, (3, 3), activation = **'relu'**))  
model.add(MaxPooling2D(pool\_size = (2, 2)))  
model.add(Conv2D(32, (3, 3), activation = **'relu'**))  
model.add(MaxPooling2D(pool\_size = (2, 2)))  
model.add(Flatten())  
model.add(Dense(units = 128, activation = **'relu'**))  
model.add(Dense(units = 1, activation = **'sigmoid'**))  
  
*# Compiling the CNN*model.compile(optimizer = **'adam'**, loss = **'binary\_crossentropy'**, metrics = [**'accuracy'**])  
  
print(model.summary())  
  
*#image augmentations*train\_datagen = ImageDataGenerator(rescale = 1./255,  
 shear\_range = 0.2,  
 zoom\_range = 0.2,  
 horizontal\_flip = **True**,  
 validation\_split=0.2)  
  
test\_datagen = ImageDataGenerator(rescale = 1./255)  
  
training\_set = train\_datagen.flow\_from\_directory(**'data/train'**,  
 target\_size = (224, 224),  
 batch\_size = 16,  
 class\_mode = **'binary'**,  
 subset = **'training'**)  
testing\_set = train\_datagen.flow\_from\_directory(**'data/train'**,  
 target\_size = (224, 224),  
 batch\_size = 16,  
 class\_mode = **'binary'**,  
 subset=**'validation'**)  
validation\_set = test\_datagen.flow\_from\_directory(**'data/validation'**,  
 target\_size = (224, 224),  
 batch\_size = 16,  
 class\_mode = **'binary'**)  
  
model.compile(loss=keras.losses.binary\_crossentropy,  
 optimizer=**'rmsprop'**,  
 metrics=[**'accuracy'**])  
  
model.fit\_generator(training\_set,  
 epochs = 12,  
 validation\_data = testing\_set,  
 verbose=1)  
  
*# Save the weights*model.save\_weights(**'my\_model\_weights.h5'**)  
  
*# Save the model architecture***with** open(**'my\_model.json'**, **'w'**) **as** f:  
 f.write(model.to\_json())  
  
score = model.evaluate\_generator(testing\_set)  
  
*# # Model reconstruction from JSON file  
# with open('my\_model.json', 'r') as f:  
# model = model\_from\_json(f.read())  
#  
# # Load weights into the new model  
# model.load\_weights('my\_model\_weights.h5')  
#  
# model.compile(loss=keras.losses.binary\_crossentropy,  
# optimizer='rmsprop',  
# metrics=['accuracy'])*print(**'Test loss:'**, score[0])  
print(**'Test accuracy:'**, score[1])  
  
img1 = cv2.imread(**'dog1.jpg'**)  
img1 = cv2.resize(img1, (224,224))  
*# img1 = img1.reshape(1,224,224,3)*img2 = cv2.imread(**'cat1.jpg'**)  
img2 = cv2.resize(img2, (224,224))  
img3 = cv2.imread(**'dog2.jpg'**)  
img3 = cv2.resize(img3, (224,224))  
img4 = cv2.imread(**'cat2.jpg'**)  
img4 = cv2.resize(img4, (224,224))  
img=[img1,img2,img3,img4]  
img=numpy.array(img)  
  
pred = model.predict(img)  
list=[]  
**for** a **in** pred:  
 **if** a>0.5:  
 list.append(**"dog"**)  
 **else**:  
 list.append(**"cat"**)  
  
f, axarr = plt.subplots(2,2)  
axarr[0,0].imshow(img1)  
axarr[0,0].set\_title(**"This is a "**+list[0])  
axarr[0,1].imshow(img2)  
axarr[0,1].set\_title(**"This is a "**+list[1])  
axarr[1,0].imshow(img3)  
axarr[1,0].set\_title(**"This is a "**+list[2])  
axarr[1,1].imshow(img4)  
axarr[1,1].set\_title(**"This is a "**+list[3])  
plt.show()

**My CNN Structure:**

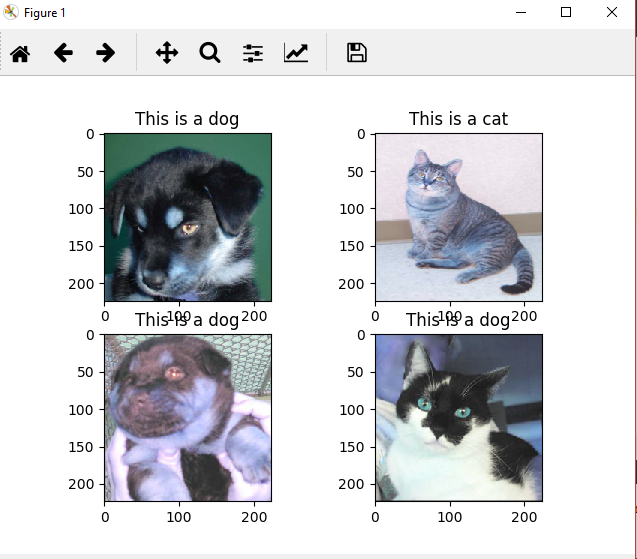


**Validation Accuracy: 76%**

**Testing Accuracy: 73%**

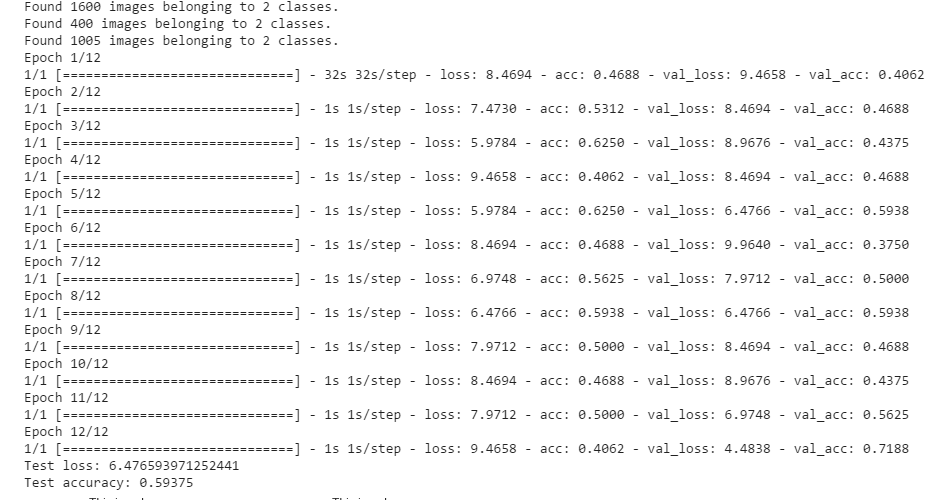


**Predictions:**



**CODE (Training VGG16 with imagenet weights) 🡪 Done on Colab:**

**from** keras.applications.vgg16 **import** VGG16  
**from** keras.preprocessing.image **import** ImageDataGenerator  
**from** keras.layers **import** Dense, Flatten  
**from** keras **import** Model  
**import** keras  
**import** matplotlib.pyplot **as** plt  
**import** cv2  
**import** numpy  
  
model = VGG16(weights=**"imagenet"**, include\_top=**True**, input\_shape=(224,224,3))  
last = model.output  
x = Flatten(name=**'flatten'**)(last)  
x = Dense(1000, activation=**'relu'**)(x)  
preds = Dense(1, activation=**'softmax'**)(x)  
model = Model(model.input, preds)  
  
print(model.summary())  
  
*#image augmentations*train\_datagen = ImageDataGenerator(rescale = 1./255,  
 shear\_range = 0.2,  
 zoom\_range = 0.2,  
 horizontal\_flip = **True**,  
 validation\_split=0.2)  
  
test\_datagen = ImageDataGenerator(rescale = 1./255)  
  
training\_set = train\_datagen.flow\_from\_directory(**"/content/drive/My Drive/Colab Notebooks/data/train"**,  
 target\_size = (224, 224),  
 batch\_size = 32,  
 class\_mode = **'binary'**,  
 subset = **'training'**)  
testing\_set = train\_datagen.flow\_from\_directory(**"/content/drive/My Drive/Colab Notebooks/data/train"**,  
 target\_size = (224, 224),  
 batch\_size = 32,  
 class\_mode = **'binary'**,  
 subset=**'validation'**)  
  
validation\_set = test\_datagen.flow\_from\_directory(**"/content/drive/My Drive/Colab Notebooks/data/validation"**,  
 target\_size = (224, 224),  
 batch\_size = 32,  
 class\_mode = **'binary'**)  
  
  
model.compile(loss=keras.losses.binary\_crossentropy,  
 optimizer=**'rmsprop'**,  
 metrics=[**'accuracy'**])  
  
model.fit\_generator(training\_set,  
 epochs = 12,  
 validation\_data = testing\_set,  
 steps\_per\_epoch=1,  
 validation\_steps=1,  
 verbose=1)  
  
*# Save the weights*model.save\_weights(**'/content/drive/My Drive/Colab Notebooks/VGG\_model\_weights.h5'**)  
  
*# Save the model architecture***with** open(**'/content/drive/My Drive/Colab Notebooks/VGG\_model.json'**, **'w'**) **as** f:  
 f.write(model.to\_json())  
  
score = model.evaluate\_generator(testing\_set, steps=1)  
  
*# # Model reconstruction from JSON file  
# with open('LSTM\_model.json', 'r') as f:  
# model = model\_from\_json(f.read())  
#  
# # Load weights into the new model  
# model.load\_weights('LSTM\_model\_weights.h5')*print(**'Test loss:'**, score[0])  
print(**'Test accuracy:'**, score[1])  
  
img1 = cv2.imread(**'/content/drive/My Drive/Colab Notebooks/dog1.jpg'**)  
img1 = cv2.resize(img1, (224,224))  
*# img1 = img1.reshape(1,224,224,3)*img2 = cv2.imread(**'/content/drive/My Drive/Colab Notebooks/cat1.jpg'**)  
img2 = cv2.resize(img2, (224,224))  
img3 = cv2.imread(**'/content/drive/My Drive/Colab Notebooks/dog2.jpg'**)  
img3 = cv2.resize(img3, (224,224))  
img4 = cv2.imread(**'/content/drive/My Drive/Colab Notebooks/cat2.jpg'**)  
img4 = cv2.resize(img4, (224,224))  
img=[img1,img2,img3,img4]  
img=numpy.array(img)  
  
pred = model.predict(img)  
list=[]  
**for** a **in** pred:  
 **if** a>0.5:  
 list.append(**"dog"**)  
 **else**:  
 list.append(**"cat"**)  
  
f, axarr = plt.subplots(2,2)  
axarr[0,0].imshow(img1)  
axarr[0,0].set\_title(**"This is a "**+list[0])  
axarr[0,1].imshow(img2)  
axarr[0,1].set\_title(**"This is a "**+list[1])  
axarr[1,0].imshow(img3)  
axarr[1,0].set\_title(**"This is a "**+list[2])  
axarr[1,1].imshow(img4)  
axarr[1,1].set\_title(**"This is a "**+list[3])  
plt.show()



**Validation Accuracy: 71%**

**Testing Accuracy: 59%**

**Predictions:**

