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

```
model = Sequential()
```

```
model.add(Conv2D(64, (3, 3), activation='relu', input_shape=(64, 64, 3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 64)	1,792
max_pooling2d (MaxPooling2D)	(None, 31, 31, 64)	0
flatten (Flatten)	(None, 61504)	0
dense (Dense)	(None, 128)	7,872,640
dense_1 (Dense)	(None, 1)	129

Total params: 7,874,561 (30.04 MB)
Trainable params: 7,874,561 (30.04 MB)
Non-trainable params: 0 (0.00 B)

Generated code may be subject to a license | 39xdgy/Self_study | 4322vipul/face_detection_with_cnn

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale =1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   rotation_range=0.2,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   fill_mode='nearest',
                                   vertical_flip=True,
                                   horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1./255)
```

Generated code may be subject to a license | ravikiran-ds/faceRec-CNN

```
Train_path='/content/drive/MyDrive/images/Train'
Test_path='/content/drive/MyDrive/images/Test'
Train_generator = train_datagen.flow_from_directory(Train_path,
                                                    target_size=(64, 64),
                                                    batch_size=32,
                                                    class_mode='binary')
Test_generator = test_datagen.flow_from_directory(Test_path,
                                                  target_size=(64, 64),
                                                  batch_size=32,
                                                  class_mode='binary')
```

```
Found 6 images belonging to 2 classes.
Found 2 images belonging to 2 classes.
```

```
Train_generator.class_indices
```

```
{'Cat': 0, 'Dog': 1}
```

Start coding or generate with AI.

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

```
model.fit(Train_generator, epochs=100, validation_data=Test_generator)
```

```
Epoch 1/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 400ms/step - accuracy: 0.8333 - loss: 0.3586 - val_accuracy: 0.5000 - val_loss: 4.42
Epoch 2/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 375ms/step - accuracy: 1.0000 - loss: 0.2298 - val_accuracy: 0.5000 - val_loss: 3.95
Epoch 3/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 332ms/step - accuracy: 0.8333 - loss: 0.3161 - val_accuracy: 0.5000 - val_loss: 3.92
Epoch 4/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 693ms/step - accuracy: 0.8333 - loss: 0.6804 - val_accuracy: 0.5000 - val_loss: 4.31
Epoch 5/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 331ms/step - accuracy: 0.8333 - loss: 0.3296 - val_accuracy: 0.5000 - val_loss: 4.85
Epoch 6/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 348ms/step - accuracy: 0.6667 - loss: 0.3584 - val_accuracy: 0.5000 - val_loss: 5.23
Epoch 7/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 358ms/step - accuracy: 1.0000 - loss: 0.3075 - val_accuracy: 0.5000 - val_loss: 5.44
Epoch 8/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 346ms/step - accuracy: 1.0000 - loss: 0.1880 - val_accuracy: 0.5000 - val_loss: 5.45
Epoch 9/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 395ms/step - accuracy: 0.8333 - loss: 0.2389 - val_accuracy: 0.5000 - val_loss: 5.28
Epoch 10/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 543ms/step - accuracy: 1.0000 - loss: 0.2041 - val_accuracy: 0.5000 - val_loss: 5.07
Epoch 11/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 809ms/step - accuracy: 0.8333 - loss: 0.2849 - val_accuracy: 0.5000 - val_loss: 4.80
Epoch 12/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 454ms/step - accuracy: 1.0000 - loss: 0.0853 - val_accuracy: 0.5000 - val_loss: 4.64
Epoch 13/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 611ms/step - accuracy: 1.0000 - loss: 0.1253 - val_accuracy: 0.5000 - val_loss: 4.60
Epoch 14/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 447ms/step - accuracy: 0.8333 - loss: 0.1828 - val_accuracy: 0.5000 - val_loss: 4.66
Epoch 15/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 402ms/step - accuracy: 0.8333 - loss: 0.3956 - val_accuracy: 0.5000 - val_loss: 4.88
Epoch 16/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 778ms/step - accuracy: 1.0000 - loss: 0.1618 - val_accuracy: 0.5000 - val_loss: 5.18
Epoch 17/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 430ms/step - accuracy: 1.0000 - loss: 0.1060 - val_accuracy: 0.5000 - val_loss: 5.33
Epoch 18/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 324ms/step - accuracy: 0.8333 - loss: 0.3311 - val_accuracy: 0.5000 - val_loss: 5.24
Epoch 19/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 319ms/step - accuracy: 0.8333 - loss: 0.1707 - val_accuracy: 0.5000 - val_loss: 5.00
Epoch 20/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 331ms/step - accuracy: 1.0000 - loss: 0.0759 - val_accuracy: 0.5000 - val_loss: 4.78
Epoch 21/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 323ms/step - accuracy: 1.0000 - loss: 0.0811 - val_accuracy: 0.5000 - val_loss: 4.62
Epoch 22/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 328ms/step - accuracy: 0.8333 - loss: 0.1625 - val_accuracy: 0.5000 - val_loss: 4.60
Epoch 23/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 339ms/step - accuracy: 1.0000 - loss: 0.0540 - val_accuracy: 0.5000 - val_loss: 4.66
Epoch 24/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 328ms/step - accuracy: 0.8333 - loss: 0.1725 - val_accuracy: 0.5000 - val_loss: 4.84
Epoch 25/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 326ms/step - accuracy: 0.8333 - loss: 0.1578 - val_accuracy: 0.5000 - val_loss: 5.14
Epoch 26/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 324ms/step - accuracy: 1.0000 - loss: 0.0997 - val_accuracy: 0.5000 - val_loss: 5.41
Epoch 27/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 610ms/step - accuracy: 1.0000 - loss: 0.2059 - val_accuracy: 0.5000 - val_loss: 5.66
Epoch 28/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 637ms/step - accuracy: 1.0000 - loss: 0.1707 - val_accuracy: 0.5000 - val_loss: 5.66
Epoch 29/100
```

```
model.save('dog-cat-classifier.h5')
```

```
nstead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.ke
```

Start coding or generate with AI.