Cats vs Dogs

-by M.Kaushik Sai

ABSTRACT:

This project focuses on classifying images into two categories: cats and dogs. We employ convolutional neural networks (CNNs) which are widely used for image classification. The dataset comprises images of cats and dogs, which are split into training and test sets. The model is trained on the training set and evaluated on the test set to assess its performance. The Model consists of various layers such as Convolution layer, Batch Normalization layer ,Dropout layer ,Dense layer with Relu(Rectified Linear Unit) activation function ,Adam optimizer, sigmoid function at output layer .Each Layer helps the model to become more robust. After the model achieves a high accuracy rate it is evaluated by making it predict from new dataset. Dataset is taken from Kaggle.

OBJECTIVE:

The objective of this project is to build a CNN model capable of accurately distinguishing between images of cats and dogs. By utilizing deep learning techniques along with required preprocessing , we aim to achieve a high level of classification accuracy on unseen data.

INTRODUCTION:

Image classification is a fundamental task in computer vision, with applications ranging from medical diagnosis to autonomous driving. In this project, we tackle the task of classifying images of cats and dogs using a CNN architecture. CNNs have proven to be highly effective in image classification tasks due to their ability to learn hierarchical features directly from raw pixel data.

This task holds practical significance in various domains, including pet identification, wildlife monitoring, and security surveillance. By accurately distinguishing between cats and dogs in images, we can automate tasks such as pet registration, animal counting in wildlife reserves, and identifying potential security threats .So let us dive deeper into neural network used in the project.

1. **Convolutional Layers**: (Feature Extraction)
    - Each layer applies a set of learnable filters (kernels) to the input image, performing convolution operations to extract various features such as edges, textures, and shapes.
2. **Pooling Layers**: (Reduce Dimension of input)
    - Common pooling operations include max pooling and average pooling, which help in reducing computational complexity, preventing overfitting, and increasing translation invariance.
3. **Activation Functions**: (Introduce non-linearity)
    - Activation functions enable CNNs to learn complex relationships and make non-linear predictions.

- Common activation functions include ReLU (Rectified Linear Unit), sigmoid, and tanh, which introduce non-linear transformations to the output of convolutional and fully connected layers.

4. **Batch Normalization**:
   - Batch normalization is a technique used to stabilize and accelerate the training of deep neural networks.
   - It normalizes the activations of each layer by adjusting and scaling them to have zero mean and unit variance, which helps in reducing internal covariate shift, improving gradient flow, and allowing for higher learning rates.

5. **Dropout Regularization**:
   - Dropout regularization is a regularization technique used to prevent overfitting in deep neural networks.
   - It randomly selects a subset of neurons in a layer and sets their outputs to zero during training, effectively dropping them out of the network. This forces the network to learn redundant representations and improves its generalization capability.

METHODOLOGY:

1. **Data Preparation**:
   - The training and validation data are prepared using **ImageDataGenerator** from Keras. Data augmentation techniques such as rescaling, shearing, zooming, and horizontal flipping are applied to the training data to increase its diversity and improve the model's generalization.
   - The training and validation data are loaded using **flow_from_directory** method, specifying the target size, batch size, and class mode.

2. **Model Definition**:
   - The CNN model is defined using **Sequential** from TensorFlow/Keras.
   - Convolutional layers with batch normalization, max pooling, and dropout are added to extract features from the input images.
   - Flatten layer is used to flatten the 2D feature maps to 1D.
   - Fully connected dense layers with batch normalization and dropout are added for classification.
   - The output layer with sigmoid activation function is used for binary classification (cat or dog).

- The model is compiled with the Adam optimizer, binary crossentropy loss, and accuracy metric.
- Total number of layers in this model are **22+1** output layer.

3. **Model Training**:
   - The model is trained using the **fit** method. Training data (**train_data**) and validation data (**validation_generator**) are passed as input.
   - The training process runs for 30 epochs, and the model's performance is evaluated on the validation set after each epoch.
4. **Prediction**:
   - After training, the model is used to make predictions on a set of test images (**Pred1.jpg** to **Pred12.jpg**).
   - Each test image is loaded, pre-processed(resizing), and passed through the trained model to predict whether it belongs to the "cat" or "dog" class.
   - The predictions are printed, and the corresponding images are displayed using **matplotlib**.
5. **Visualization**:
   - Finally, the training history (accuracy and loss) is visualized using **matplotlib** to understand the training progress and model performance over epochs.

CODE:

I have used jupyter notebook to explain each line. I have taken screenshots for the following outputs(code in text is present after screenshots)

Cats vs Dogs
-by M.Kaushik Sai

For this project we import tensorflow,keras,Matplotlib,numpy

tensorflow (imported as tf):

TensorFlow is an open-source machine learning framework developed by Google.
It provides tools and libraries for building and training deep learning models, including neural networks.
keras.preprocessing.image:

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow.
The image module from keras.preprocessing provides utilities for working with image data, including loading, preprocessing, and augmenting images.

matplotlib.pyplot (imported as plt):

Matplotlib is a plotting library for Python.
The pyplot module provides a MATLAB-like plotting interface, allowing users to create and customize plots easily.
matplotlib.image (imported as mpimg):

Matplotlib's image module provides functions for working with image data within Matplotlib.
It includes utilities for loading and displaying images in various formats.

numpy (imported as np):

NumPy is a fundamental package for scientific computing with Python.

numpy (imported as np):

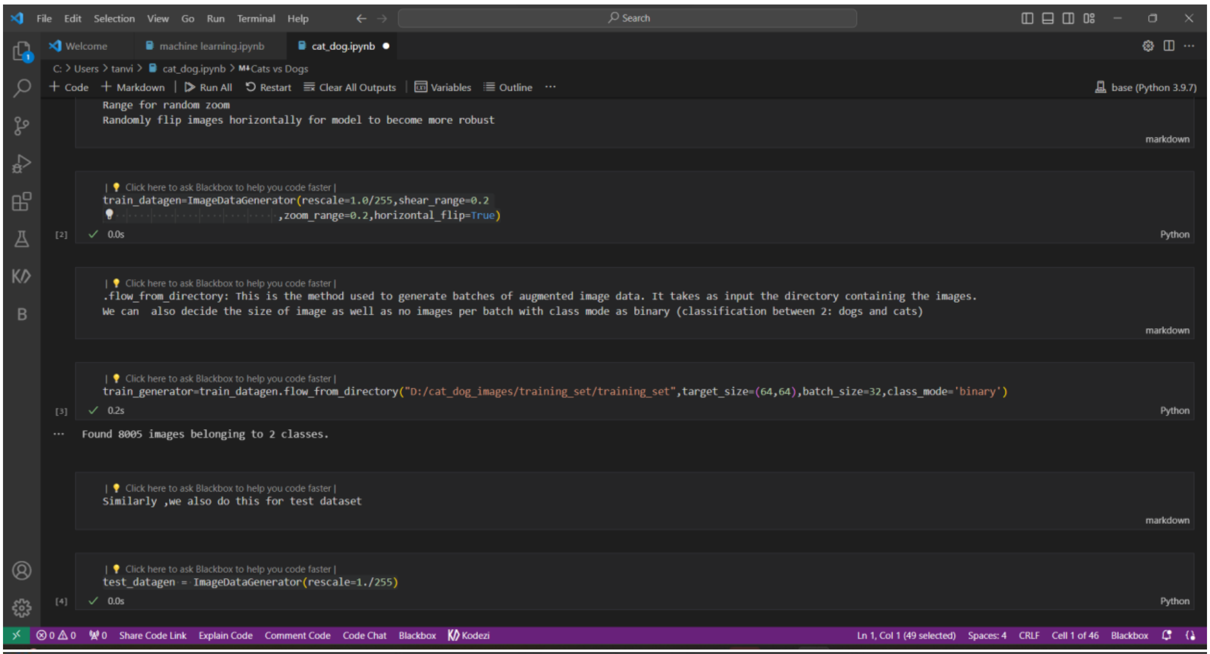
NumPy is a fundamental package for scientific computing with Python.
It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

```python
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
```

```
c:\Users\tanvi\anaconda3\lib\site-packages\scipy\__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.26.3
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
WARNING:tensorflow:From c:\Users\tanvi\anaconda3\lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.loss
```

data preprocessing

Data Augmentation
We use a image data generator with the following parameters
Rescale pixel values to the range [0, 1]
Shear intensity (in radians)
Range for random zoom
Randomly flip images horizontally for model to become more robust



Range for random zoom
Randomly flip images horizontally for model to become more robust

```python
train_datagen=ImageDataGenerator(rescale=1.0/255,shear_range=0.2
                                 ,zoom_range=0.2,horizontal_flip=True)
```

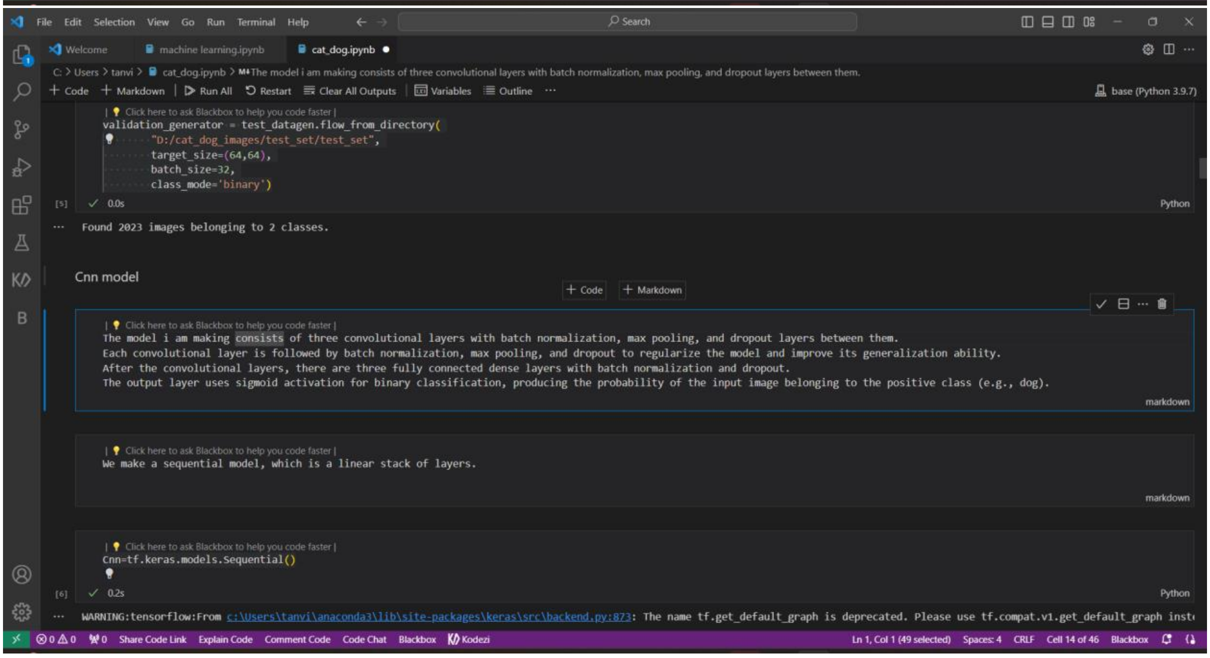
.flow_from_directory: This is the method used to generate batches of augmented image data. It takes as input the directory containing the images.
We can also decide the size of image as well as no images per batch with class mode as binary (classification between 2: dogs and cats)

```python
train_generator=train_datagen.flow_from_directory("D:/cat_dog_images/training_set/training_set",target_size=(64,64),batch_size=32,class_mode='binary')
```

```
Found 8005 images belonging to 2 classes.
```

Similarly ,we also do this for test dataset

```python
test_datagen = ImageDataGenerator(rescale=1./255)
```



```python
validation_generator = test_datagen.flow_from_directory(
        "D:/cat_dog_images/test_set/test_set",
        target_size=(64,64),
        batch_size=32,
        class_mode='binary')
```

```
Found 2023 images belonging to 2 classes.
```

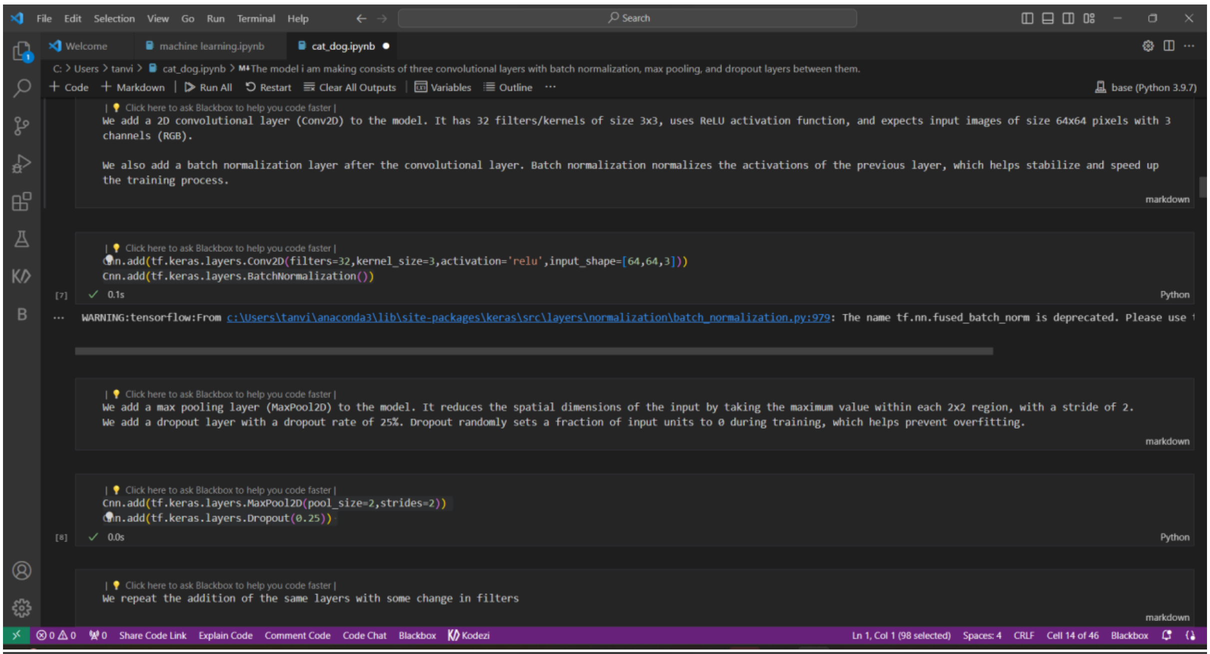
Cnn model

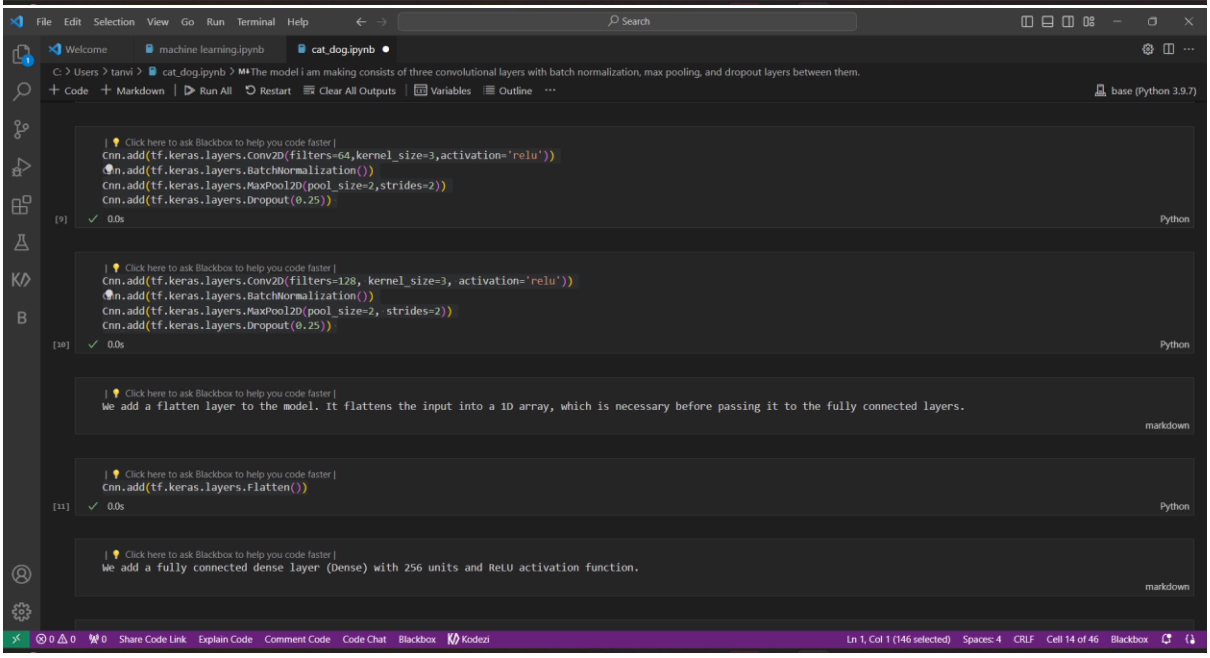
The model i am making consists of three convolutional layers with batch normalization, max pooling, and dropout layers between them.
Each convolutional layer is followed by batch normalization, max pooling, and dropout to regularize the model and improve its generalization ability.
After the convolutional layers, there are three fully connected dense layers with batch normalization and dropout.
The output layer uses sigmoid activation for binary classification, producing the probability of the input image belonging to the positive class (e.g., dog).

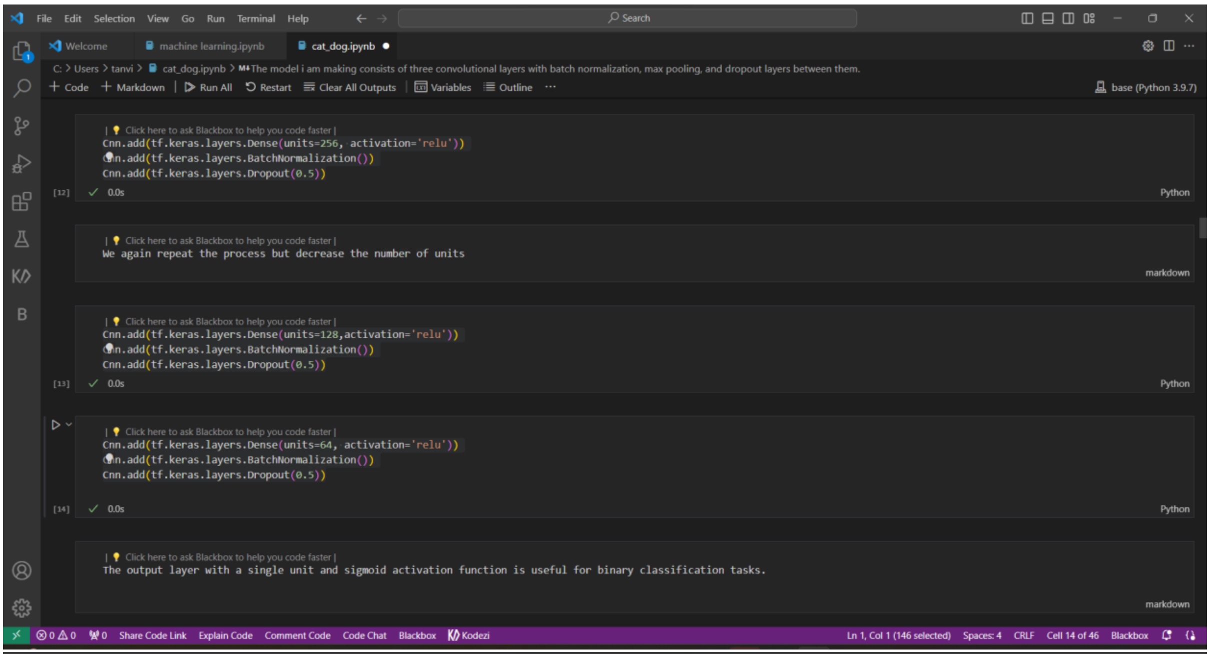
We make a sequential model, which is a linear stack of layers.

```python
Cnn=tf.keras.models.Sequential()
```

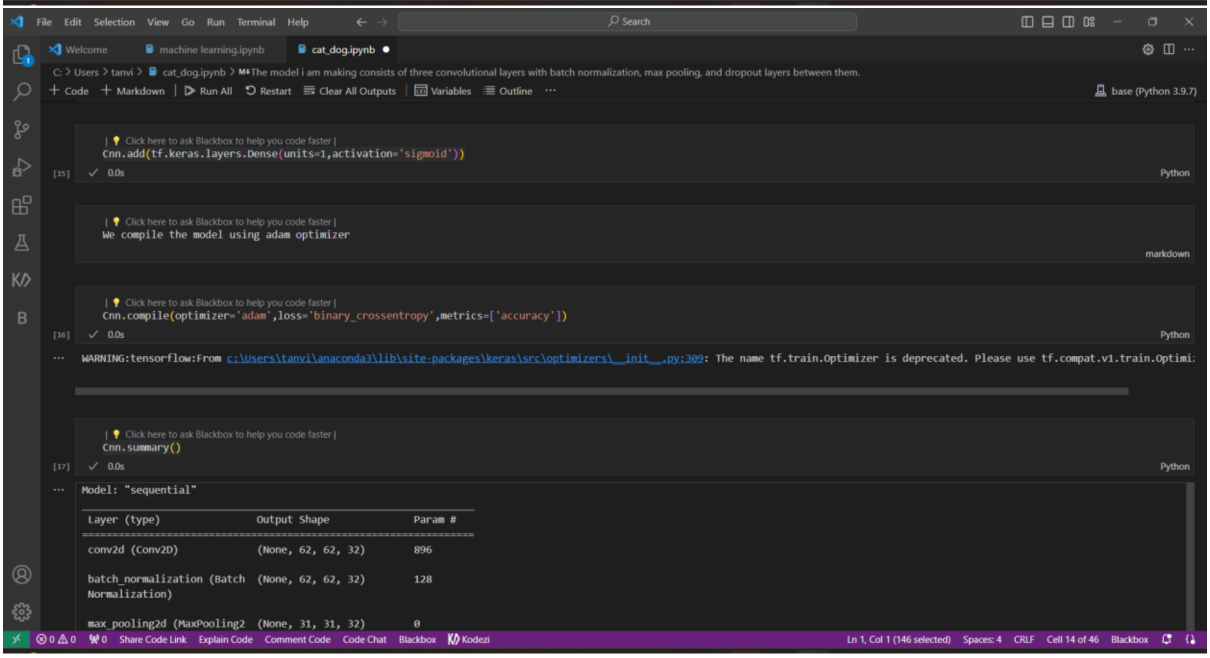
```
WARNING:tensorflow:From c:\Users\tanvi\anaconda3\lib\site-packages\keras\src\backend.py:873: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph inst
```


File Edit Selection View Go Run Terminal Help
Search
Welcome
machine learning.ipynb
cat_dog.ipynb
C: > Users > tanvi > cat_dog.ipynb > M↓The model i am making consists of three convolutional layers with batch normalization, max pooling, and dropout layers between them.
+ Code + Markdown | Run All Restart Clear All Outputs Variables Outline ...
base (Python 3.9.7)
| Click here to ask Blackbox to help you code faster |
We add a 2D convolutional layer (Conv2D) to the model. It has 32 filters/kernels of size 3x3, uses ReLU activation function, and expects input images of size 64x64 pixels with 3 channels (RGB).
We also add a batch normalization layer after the convolutional layer. Batch normalization normalizes the activations of the previous layer, which helps stabilize and speed up the training process.
markdown
| Click here to ask Blackbox to help you code faster |
Cnn.add(tf.keras.layers.Conv2D(filters=32,kernel_size=3,activation='relu',input_shape=[64,64,3]))
Cnn.add(tf.keras.layers.BatchNormalization())
[7] ✓ 0.1s
Python
WARNING:tensorflow:From c:\Users\tanvi\anaconda3\lib\site-packages\keras\src\layers\normalization\batch_normalization.py:979: The name tf.nn.fused_batch_norm is deprecated. Please use t
| Click here to ask Blackbox to help you code faster |
We add a max pooling layer (MaxPool2D) to the model. It reduces the spatial dimensions of the input by taking the maximum value within each 2x2 region, with a stride of 2.
We add a dropout layer with a dropout rate of 25%. Dropout randomly sets a fraction of input units to 0 during training, which helps prevent overfitting.
markdown
| Click here to ask Blackbox to help you code faster |
Cnn.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))
Cnn.add(tf.keras.layers.Dropout(0.25))
[8] ✓ 0.0s
Python
| Click here to ask Blackbox to help you code faster |
We repeat the addition of the same layers with some change in filters
markdown
Share Code Link Explain Code Comment Code Code Chat Blackbox Kodezi
Ln 1, Col 1 (98 selected) Spaces: 4 CRLF Cell 14 of 46 Blackbox

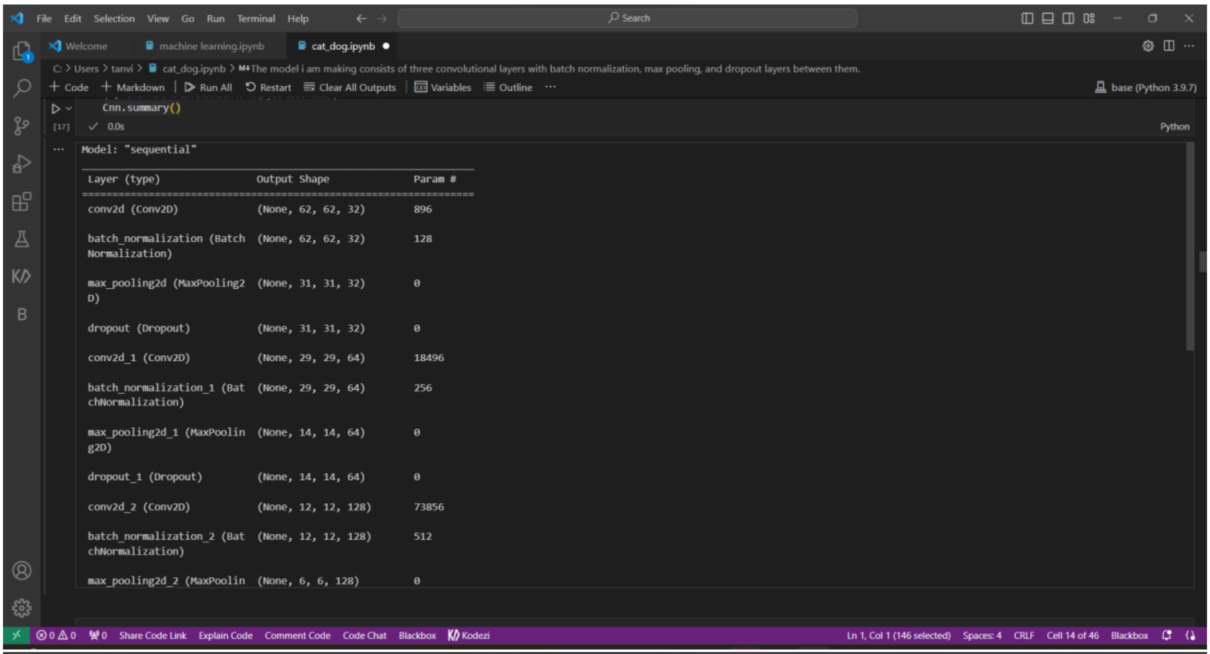

File Edit Selection View Go Run Terminal Help
Search
Welcome
machine learning.ipynb
cat_dog.ipynb
C: > Users > tanvi > cat_dog.ipynb > M↓The model i am making consists of three convolutional layers with batch normalization, max pooling, and dropout layers between them.
+ Code + Markdown | Run All Restart Clear All Outputs Variables Outline ...
base (Python 3.9.7)
| Click here to ask Blackbox to help you code faster |
Cnn.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,activation='relu'))
Cnn.add(tf.keras.layers.BatchNormalization())
Cnn.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))
Cnn.add(tf.keras.layers.Dropout(0.25))
[9] ✓ 0.0s
Python
| Click here to ask Blackbox to help you code faster |
Cnn.add(tf.keras.layers.Conv2D(filters=128, kernel_size=3, activation='relu'))
Cnn.add(tf.keras.layers.BatchNormalization())
Cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
Cnn.add(tf.keras.layers.Dropout(0.25))
[10] ✓ 0.0s
Python
| Click here to ask Blackbox to help you code faster |
We add a flatten layer to the model. It flattens the input into a 1D array, which is necessary before passing it to the fully connected layers.
markdown
| Click here to ask Blackbox to help you code faster |
Cnn.add(tf.keras.layers.Flatten())
[11] ✓ 0.0s
Python
| Click here to ask Blackbox to help you code faster |
We add a fully connected dense layer (Dense) with 256 units and ReLU activation function.
markdown
Share Code Link Explain Code Comment Code Code Chat Blackbox Kodezi
Ln 1, Col 1 (146 selected) Spaces: 4 CRLF Cell 14 of 46 Blackbox



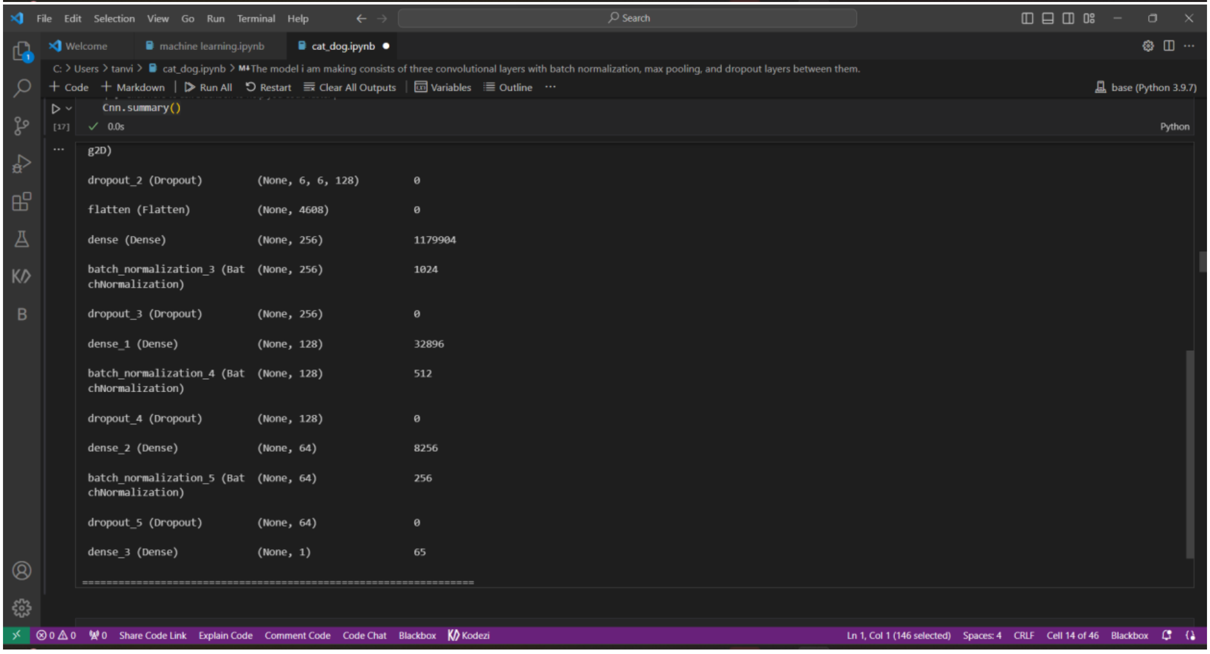
C: > Users > tanvi > cat_dog.ipynb > M↓The model i am making consists of three convolutional layers with batch normalization, max pooling, and dropout layers between them.
Cnn.add(tf.keras.layers.Dense(units=256, activation='relu'))
Cnn.add(tf.keras.layers.BatchNormalization())
Cnn.add(tf.keras.layers.Dropout(0.5))
[12] 0.0s
We again repeat the process but decrease the number of units
Cnn.add(tf.keras.layers.Dense(units=128,activation='relu'))
Cnn.add(tf.keras.layers.BatchNormalization())
Cnn.add(tf.keras.layers.Dropout(0.5))
[13] 0.0s
Cnn.add(tf.keras.layers.Dense(units=64, activation='relu'))
Cnn.add(tf.keras.layers.BatchNormalization())
Cnn.add(tf.keras.layers.Dropout(0.5))
[14] 0.0s
The output layer with a single unit and sigmoid activation function is useful for binary classification tasks.



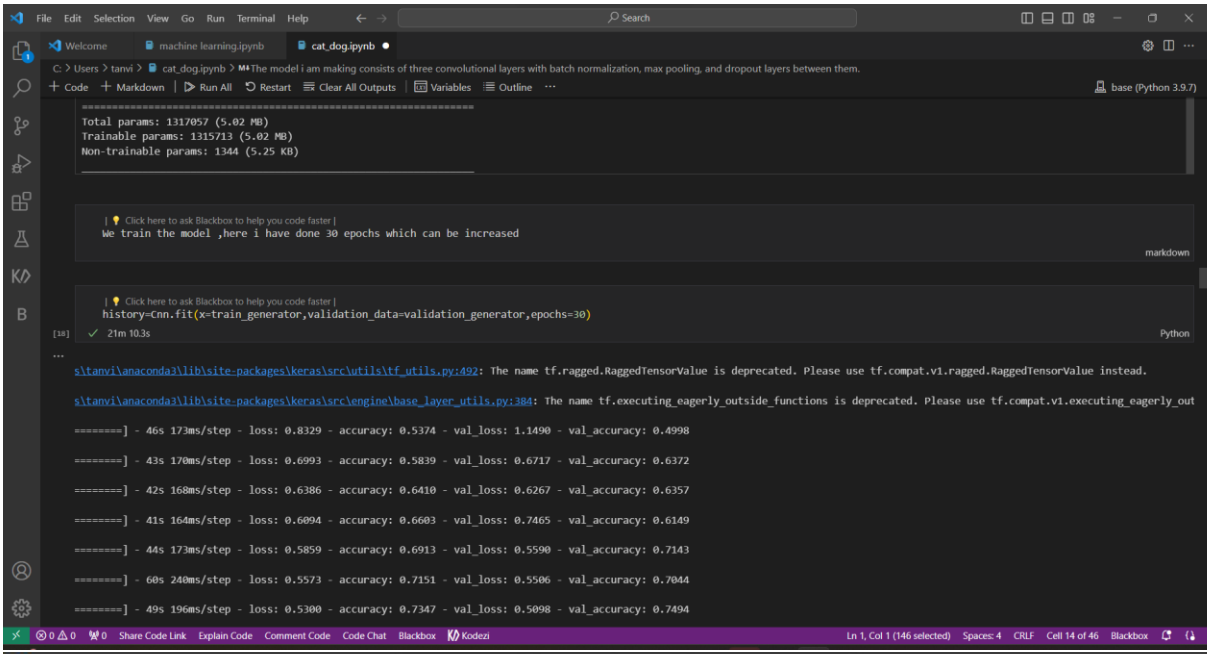
C: > Users > tanvi > cat_dog.ipynb > M↓The model i am making consists of three convolutional layers with batch normalization, max pooling, and dropout layers between them.
Cnn.add(tf.keras.layers.Dense(units=1,activation='sigmoid'))
[15] 0.0s
We compile the model using adam optimizer
Cnn.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
[16] 0.0s
WARNING:tensorflow:From c:\Users\tanvi\anaconda3\lib\site-packages\keras\src\optimizers__init__.py:309: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimi:
Cnn.summary()
[17] 0.0s
Model: "sequential"
Layer (type) Output Shape Param #
conv2d (Conv2D) (None, 62, 62, 32) 896
batch_normalization (Batch Normalization) (None, 62, 62, 32) 128
max_pooling2d (MaxPooling2 (None, 31, 31, 32) 0

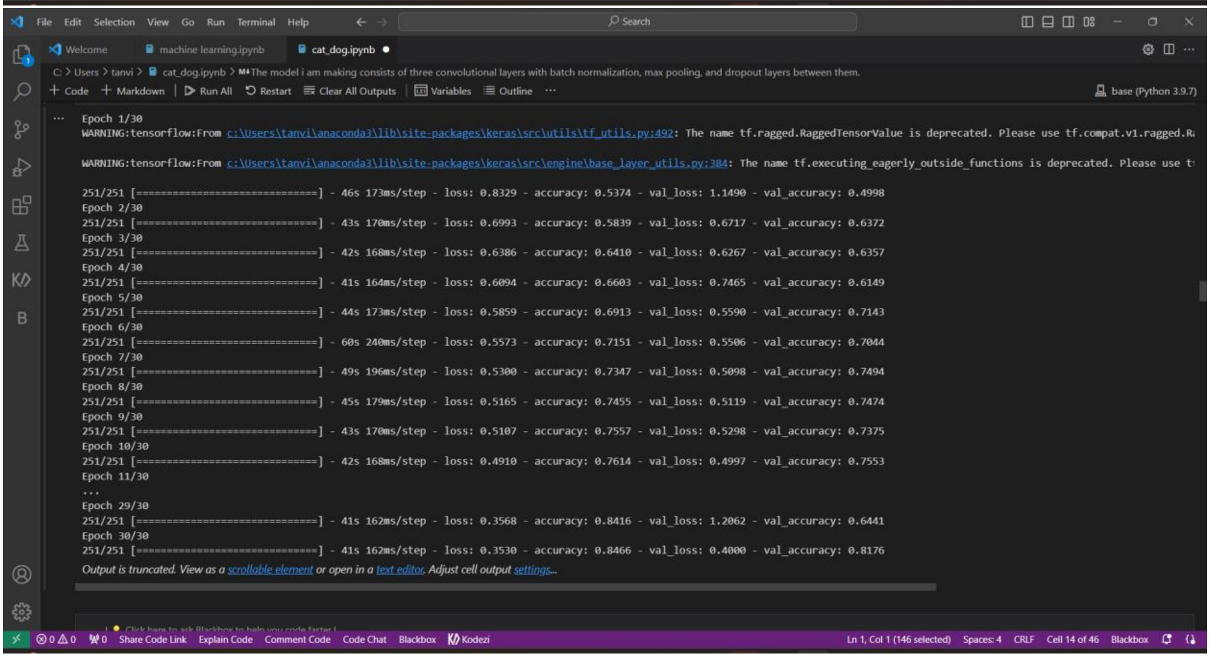


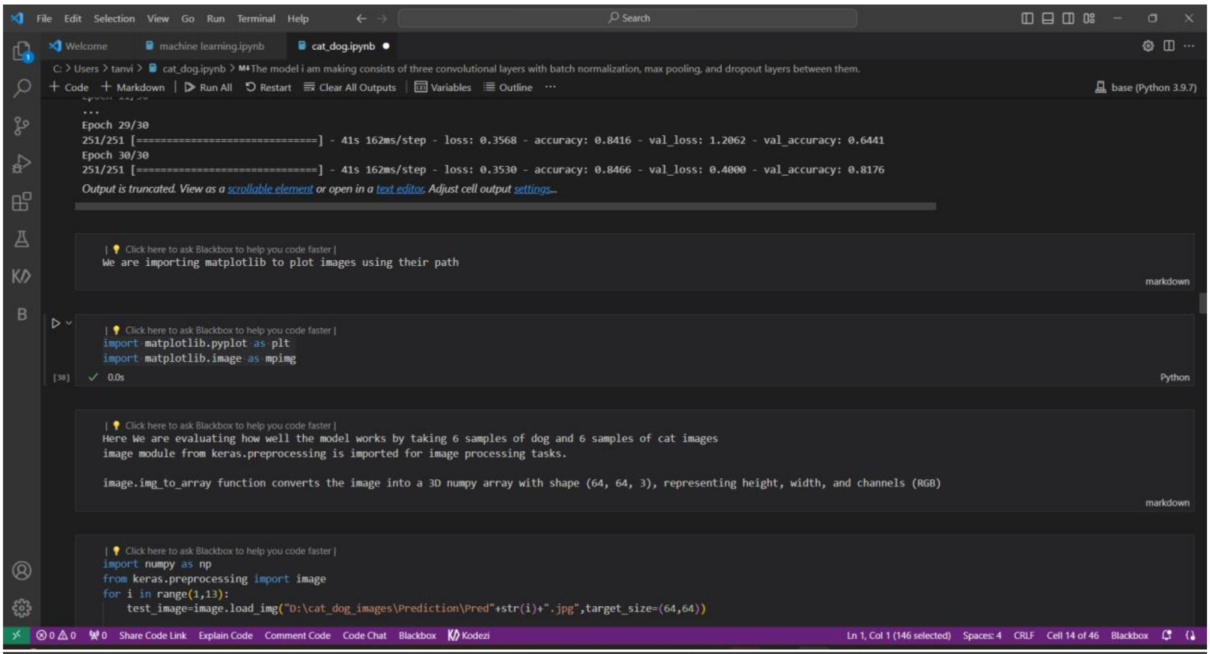
cat_dog.ipynb
The model i am making consists of three convolutional layers with batch normalization, max pooling, and dropout layers between them.
cnn.summary()
Model: "sequential"
Layer (type) Output Shape Param #
conv2d (Conv2D) (None, 62, 62, 32) 896
batch_normalization (BatchNormalization) (None, 62, 62, 32) 128
max_pooling2d (MaxPooling2D) (None, 31, 31, 32) 0
dropout (Dropout) (None, 31, 31, 32) 0
conv2d_1 (Conv2D) (None, 29, 29, 64) 18496
batch_normalization_1 (BatchNormalization) (None, 29, 29, 64) 256
max_pooling2d_1 (MaxPooling2D) (None, 14, 14, 64) 0
dropout_1 (Dropout) (None, 14, 14, 64) 0
conv2d_2 (Conv2D) (None, 12, 12, 128) 73856
batch_normalization_2 (BatchNormalization) (None, 12, 12, 128) 512
max_pooling2d_2 (MaxPoolin (None, 6, 6, 128) 0

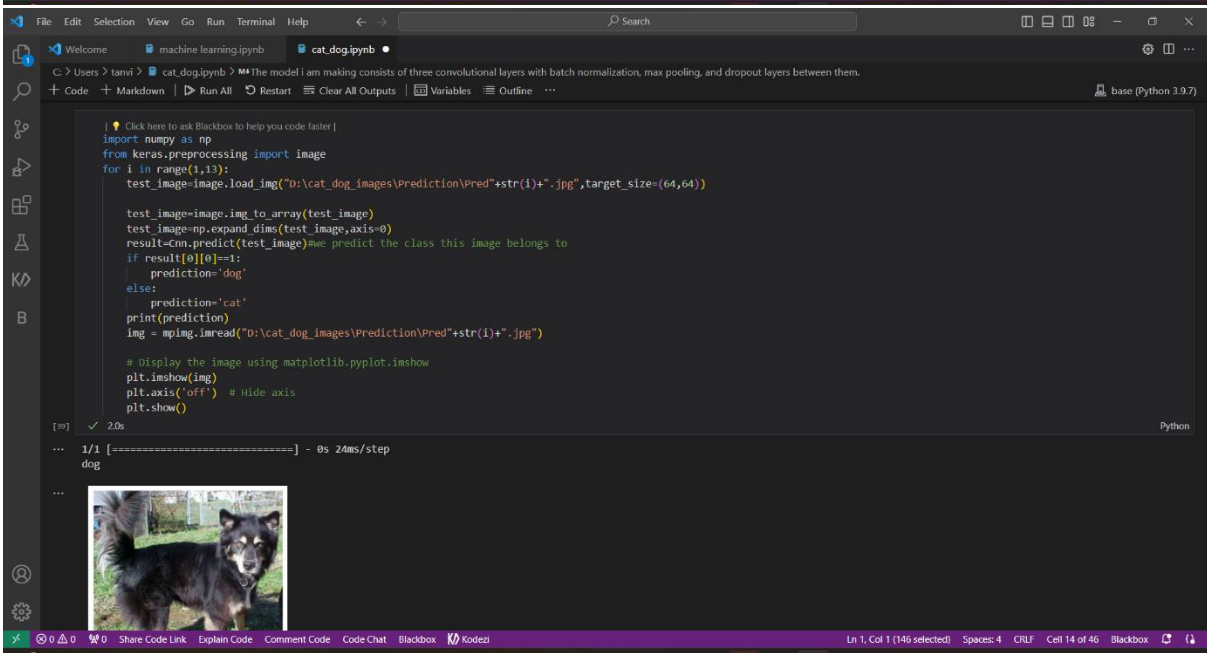


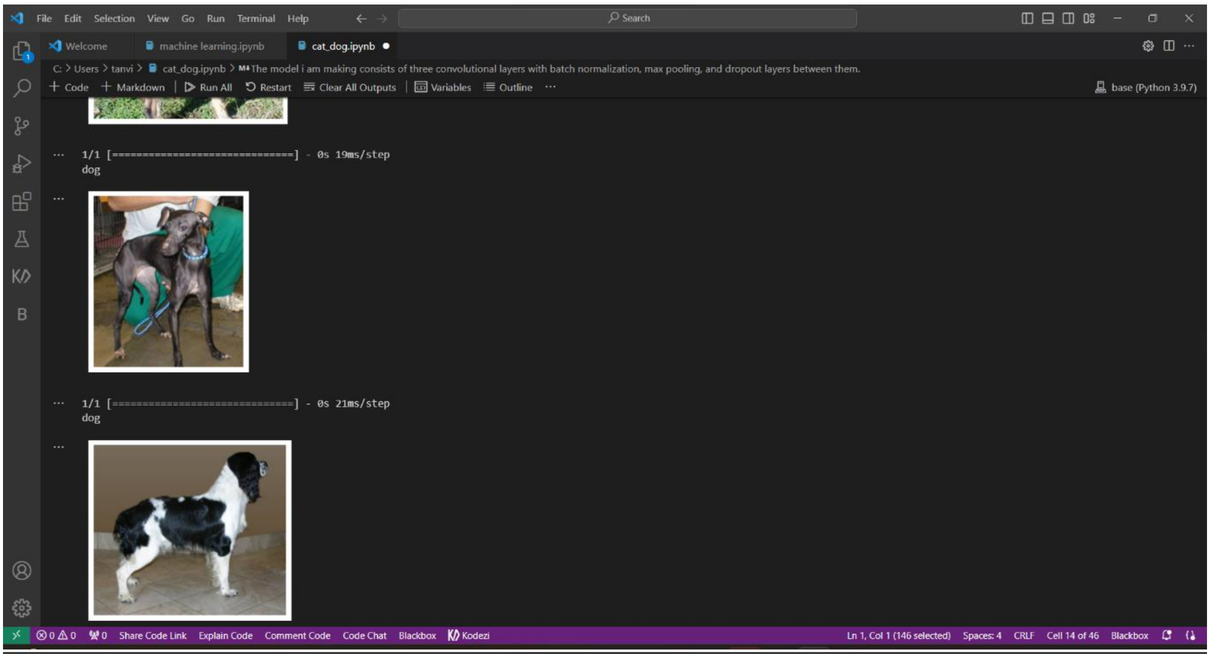
cat_dog.ipynb
The model i am making consists of three convolutional layers with batch normalization, max pooling, and dropout layers between them.
cnn.summary()
g2D)
dropout_2 (Dropout) (None, 6, 6, 128) 0
flatten (Flatten) (None, 4608) 0
dense (Dense) (None, 256) 1179904
batch_normalization_3 (BatchNormalization) (None, 256) 1024
dropout_3 (Dropout) (None, 256) 0
dense_1 (Dense) (None, 128) 32896
batch_normalization_4 (BatchNormalization) (None, 128) 512
dropout_4 (Dropout) (None, 128) 0
dense_2 (Dense) (None, 64) 8256
batch_normalization_5 (BatchNormalization) (None, 64) 256
dropout_5 (Dropout) (None, 64) 0
dense_3 (Dense) (None, 1) 65

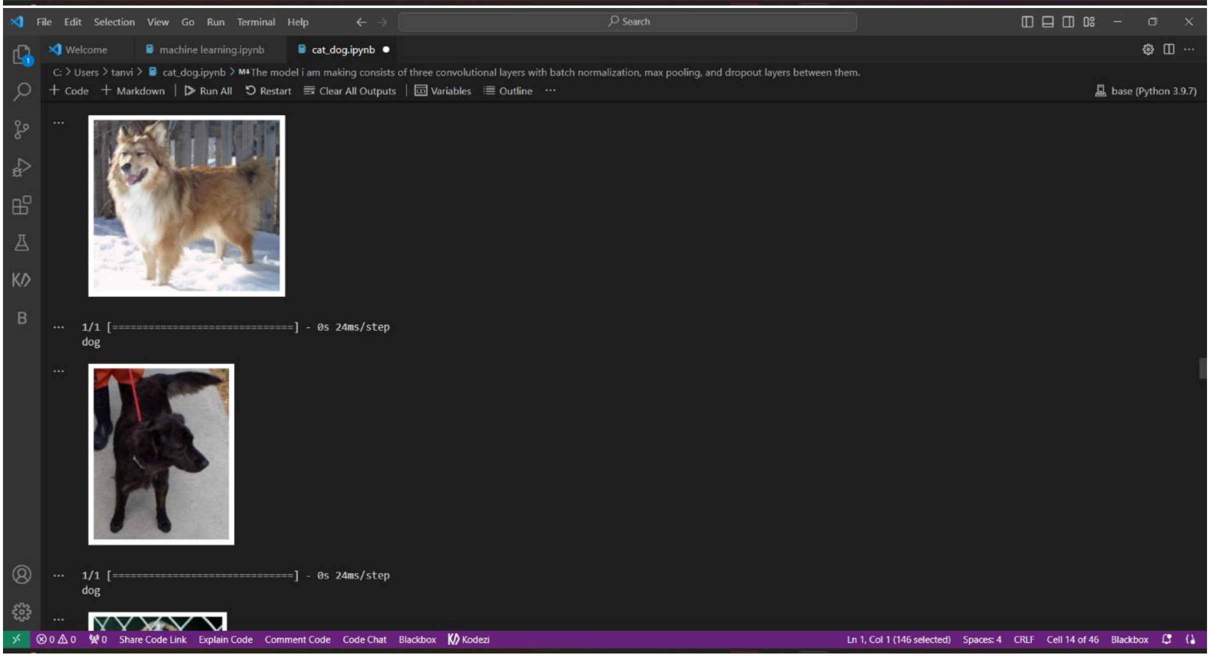

File Edit Selection View Go Run Terminal Help
Search
Welcome
machine learning.ipynb
cat_dog.ipynb
C: > Users > tanvi > cat_dog.ipynb > M↓The model i am making consists of three convolutional layers with batch normalization, max pooling, and dropout layers between them.
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
base (Python 3.9.7)
Total params: 1317057 (5.02 MB)
Trainable params: 1315713 (5.02 MB)
Non-trainable params: 1344 (5.25 KB)
| Click here to ask Blackbox to help you code faster |
We train the model ,here i have done 30 epochs which can be increased
markdown
| Click here to ask Blackbox to help you code faster |
history=Cnn.fit(x=train_generator,validation_data=validation_generator,epochs=30)
[18]
21m 10.3s
Python
s\tanvi\anaconda3\lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.
s\tanvi\anaconda3\lib\site-packages\keras\src\engine\base_layer_utils.py:384: The name tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_out
=======] - 46s 173ms/step - loss: 0.8329 - accuracy: 0.5374 - val_loss: 1.1490 - val_accuracy: 0.4998
=======] - 43s 170ms/step - loss: 0.6993 - accuracy: 0.5839 - val_loss: 0.6717 - val_accuracy: 0.6372
=======] - 42s 168ms/step - loss: 0.6386 - accuracy: 0.6410 - val_loss: 0.6267 - val_accuracy: 0.6357
=======] - 41s 164ms/step - loss: 0.6094 - accuracy: 0.6603 - val_loss: 0.7465 - val_accuracy: 0.6149
=======] - 44s 173ms/step - loss: 0.5859 - accuracy: 0.6913 - val_loss: 0.5590 - val_accuracy: 0.7143
=======] - 60s 240ms/step - loss: 0.5573 - accuracy: 0.7151 - val_loss: 0.5506 - val_accuracy: 0.7044
=======] - 49s 196ms/step - loss: 0.5300 - accuracy: 0.7347 - val_loss: 0.5098 - val_accuracy: 0.7494
Share Code Link
Explain Code
Comment Code
Code Chat
Blackbox
Kodezi
Ln 1, Col 1 (146 selected)
Spaces: 4
CRLF
Cell 14 of 46
Blackbox

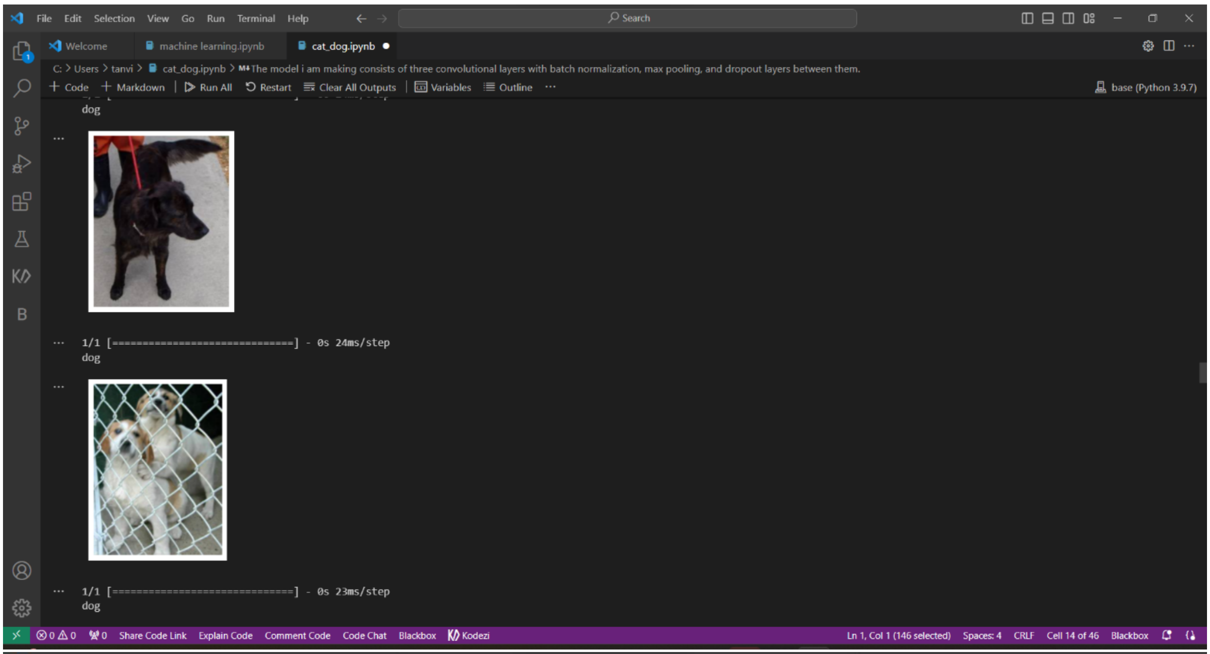

File Edit Selection View Go Run Terminal Help
Search
Welcome
machine learning.ipynb
cat_dog.ipynb
C: > Users > tanvi > cat_dog.ipynb > M↓The model i am making consists of three convolutional layers with batch normalization, max pooling, and dropout layers between them.
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
base (Python 3.9.7)
Epoch 1/30
WARNING:tensorflow:From c:\Users\tanvi\anaconda3\lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.R
WARNING:tensorflow:From c:\Users\tanvi\anaconda3\lib\site-packages\keras\src\engine\base_layer_utils.py:384: The name tf.executing_eagerly_outside_functions is deprecated. Please use t
251/251 [==============================] - 46s 173ms/step - loss: 0.8329 - accuracy: 0.5374 - val_loss: 1.1490 - val_accuracy: 0.4998
Epoch 2/30
251/251 [==============================] - 43s 170ms/step - loss: 0.6993 - accuracy: 0.5839 - val_loss: 0.6717 - val_accuracy: 0.6372
Epoch 3/30
251/251 [==============================] - 42s 168ms/step - loss: 0.6386 - accuracy: 0.6410 - val_loss: 0.6267 - val_accuracy: 0.6357
Epoch 4/30
251/251 [==============================] - 41s 164ms/step - loss: 0.6094 - accuracy: 0.6603 - val_loss: 0.7465 - val_accuracy: 0.6149
Epoch 5/30
251/251 [==============================] - 44s 173ms/step - loss: 0.5859 - accuracy: 0.6913 - val_loss: 0.5590 - val_accuracy: 0.7143
Epoch 6/30
251/251 [==============================] - 60s 240ms/step - loss: 0.5573 - accuracy: 0.7151 - val_loss: 0.5506 - val_accuracy: 0.7044
Epoch 7/30
251/251 [==============================] - 49s 196ms/step - loss: 0.5300 - accuracy: 0.7347 - val_loss: 0.5098 - val_accuracy: 0.7494
Epoch 8/30
251/251 [==============================] - 45s 179ms/step - loss: 0.5165 - accuracy: 0.7455 - val_loss: 0.5119 - val_accuracy: 0.7474
Epoch 9/30
251/251 [==============================] - 43s 170ms/step - loss: 0.5107 - accuracy: 0.7557 - val_loss: 0.5298 - val_accuracy: 0.7375
Epoch 10/30
251/251 [==============================] - 42s 168ms/step - loss: 0.4910 - accuracy: 0.7614 - val_loss: 0.4997 - val_accuracy: 0.7553
Epoch 11/30
...
Epoch 29/30
251/251 [==============================] - 41s 162ms/step - loss: 0.3568 - accuracy: 0.8416 - val_loss: 1.2062 - val_accuracy: 0.6441
Epoch 30/30
251/251 [==============================] - 41s 162ms/step - loss: 0.3530 - accuracy: 0.8466 - val_loss: 0.4000 - val_accuracy: 0.8176
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
Share Code Link
Explain Code
Comment Code
Code Chat
Blackbox
Kodezi
Ln 1, Col 1 (146 selected)
Spaces: 4
CRLF
Cell 14 of 46
Blackbox

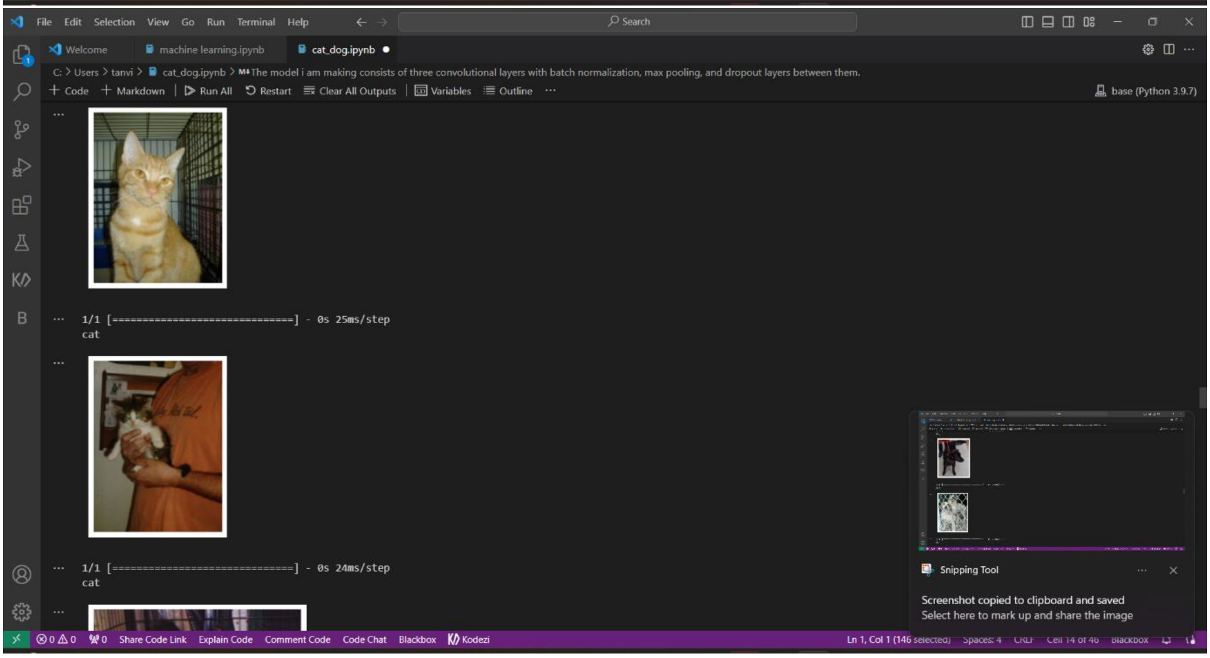

cat_dog.ipynb
C: > Users > tanvi > cat_dog.ipynb > M↓The model i am making consists of three convolutional layers with batch normalization, max pooling, and dropout layers between them.
Epoch 29/30
251/251 [==============================] - 41s 162ms/step - loss: 0.3568 - accuracy: 0.8416 - val_loss: 1.2062 - val_accuracy: 0.6441
Epoch 30/30
251/251 [==============================] - 41s 162ms/step - loss: 0.3530 - accuracy: 0.8466 - val_loss: 0.4000 - val_accuracy: 0.8176
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
We are importing matplotlib to plot images using their path
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
Here We are evaluating how well the model works by taking 6 samples of dog and 6 samples of cat images
image module from keras.preprocessing is imported for image processing tasks.
image.img_to_array function converts the image into a 3D numpy array with shape (64, 64, 3), representing height, width, and channels (RGB)
import numpy as np
from keras.preprocessing import image
for i in range(1,13):
test_image=image.load_img("D:\cat_dog_images\Prediction\Pred"+str(i)+".jpg",target_size=(64,64))

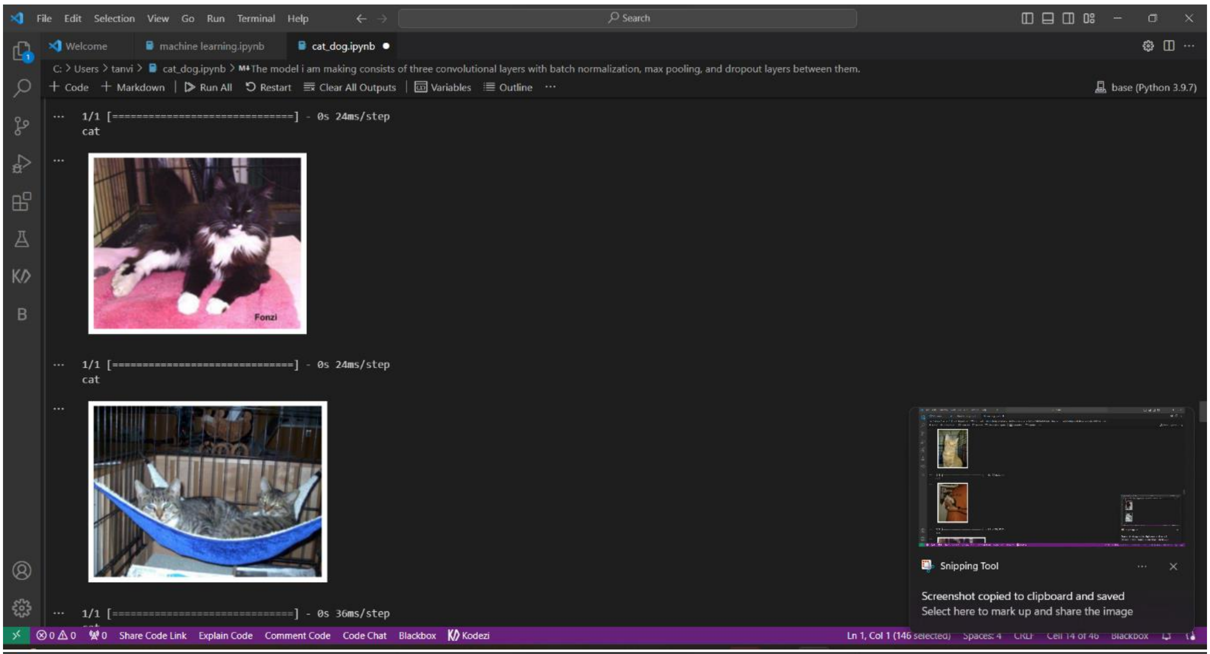

cat_dog.ipynb
import numpy as np
from keras.preprocessing import image
for i in range(1,13):
test_image=image.load_img("D:\cat_dog_images\Prediction\Pred"+str(i)+".jpg",target_size=(64,64))
test_image=image.img_to_array(test_image)
test_image=np.expand_dims(test_image,axis=0)
result=Cnn.predict(test_image)#we predict the class this image belongs to
if result[0][0]==1:
prediction='dog'
else:
prediction='cat'
print(prediction)
img = mpimg.imread("D:\cat_dog_images\Prediction\Pred"+str(i)+".jpg")
# Display the image using matplotlib.pyplot.imshow
plt.imshow(img)
plt.axis('off') # Hide axis
plt.show()
1/1 [==============================] - 0s 24ms/step
dog

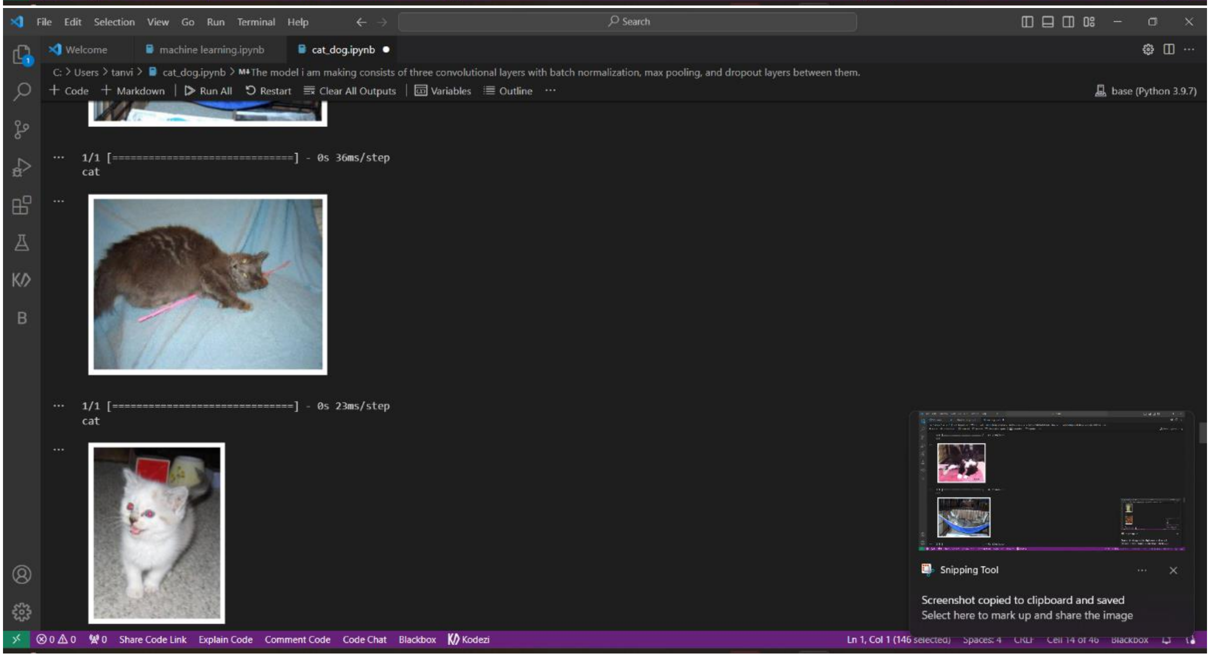

File Edit Selection View Go Run Terminal Help
Search
Welcome
machine learning.ipynb
cat_dog.ipynb
C: > Users > tanvi > cat_dog.ipynb > M↓The model i am making consists of three convolutional layers with batch normalization, max pooling, and dropout layers between them.
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
base (Python 3.9.7)
1/1 [==============================] - 0s 19ms/step
dog
1/1 [==============================] - 0s 21ms/step
dog
Share Code Link
Explain Code
Comment Code
Code Chat
Blackbox
Kodezi
Ln 1, Col 1 (146 selected)
Spaces: 4
CRLF
Cell 14 of 46
Blackbox

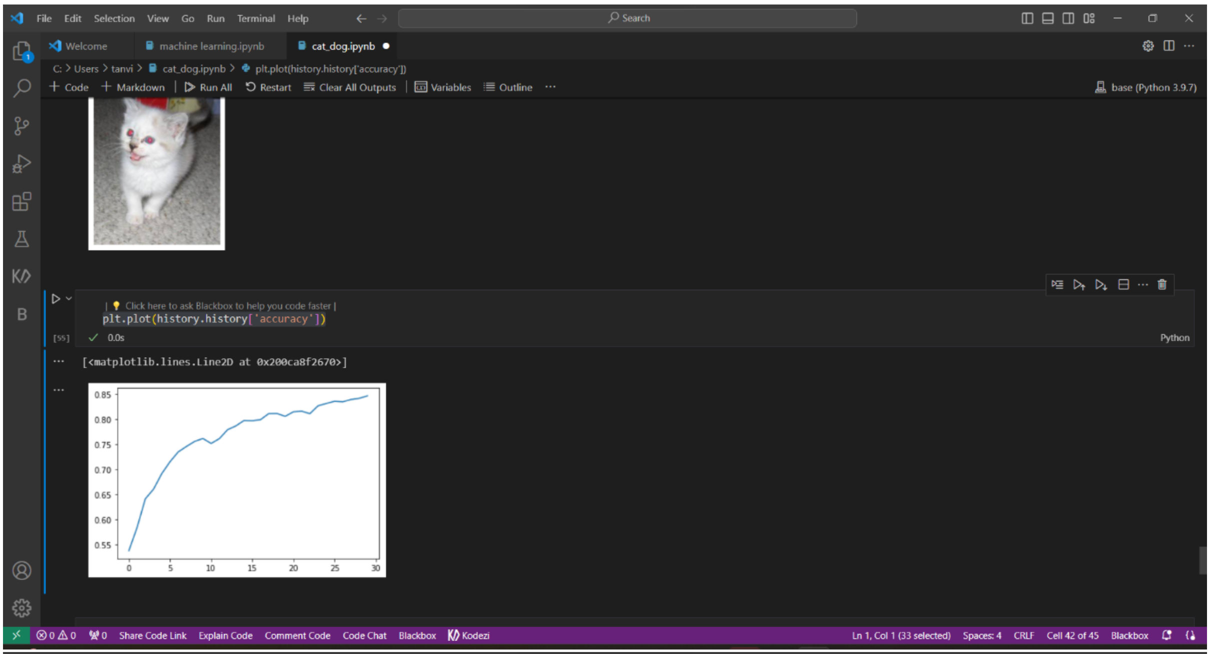

File Edit Selection View Go Run Terminal Help
Search
Welcome
machine learning.ipynb
cat_dog.ipynb
C: > Users > tanvi > cat_dog.ipynb > M↓The model i am making consists of three convolutional layers with batch normalization, max pooling, and dropout layers between them.
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
base (Python 3.9.7)
1/1 [==============================] - 0s 24ms/step
dog
1/1 [==============================] - 0s 24ms/step
dog
Share Code Link
Explain Code
Comment Code
Code Chat
Blackbox
Kodezi
Ln 1, Col 1 (146 selected)
Spaces: 4
CRLF
Cell 14 of 46
Blackbox

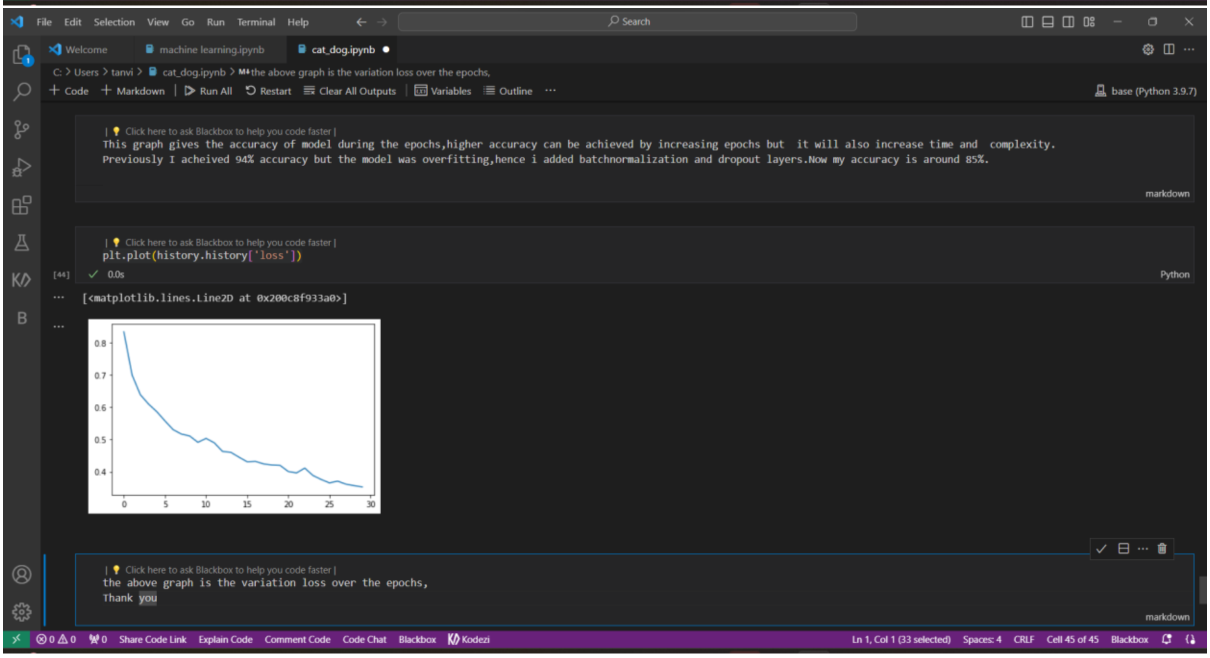

File Edit Selection View Go Run Terminal Help
Search
Welcome
machine learning.ipynb
cat_dog.ipynb
C: > Users > tanvi > cat_dog.ipynb > M↓The model i am making consists of three convolutional layers with batch normalization, max pooling, and dropout layers between them.
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
base (Python 3.9.7)
dog
1/1 [==============================] - 0s 24ms/step
dog
1/1 [==============================] - 0s 23ms/step
dog
Share Code Link
Explain Code
Comment Code
Code Chat
Blackbox
Kodezi
Ln 1, Col 1 (146 selected)
Spaces: 4
CRLF
Cell 14 of 46
Blackbox


File Edit Selection View Go Run Terminal Help
Search
Welcome
machine learning.ipynb
cat_dog.ipynb
C: > Users > tanvi > cat_dog.ipynb > M↓The model i am making consists of three convolutional layers with batch normalization, max pooling, and dropout layers between them.
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
base (Python 3.9.7)
1/1 [==============================] - 0s 25ms/step
cat
1/1 [==============================] - 0s 24ms/step
cat
Snipping Tool
Screenshot copied to clipboard and saved
Select here to mark up and share the image
Share Code Link
Explain Code
Comment Code
Code Chat
Blackbox
Kodezi
Ln 1, Col 1 (146 selected)


File Edit Selection View Go Run Terminal Help
Search
Welcome
machine learning.ipynb
cat_dog.ipynb
C: > Users > tanvi > cat_dog.ipynb > M↓The model i am making consists of three convolutional layers with batch normalization, max pooling, and dropout layers between them.
+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ...
base (Python 3.9.7)
1/1 [==============================] - 0s 24ms/step
cat
Fonzi
1/1 [==============================] - 0s 24ms/step
cat
1/1 [==============================] - 0s 36ms/step
Snipping Tool
Screenshot copied to clipboard and saved
Select here to mark up and share the image
Share Code Link Explain Code Comment Code Code Chat Blackbox Kodezi
Ln 1, Col 1 (146 selected) Spaces: 4 CRLF Cell 14 of 46 Blackbox


File Edit Selection View Go Run Terminal Help
Search
Welcome
machine learning.ipynb
cat_dog.ipynb
C: > Users > tanvi > cat_dog.ipynb > M↓The model i am making consists of three convolutional layers with batch normalization, max pooling, and dropout layers between them.
+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ...
base (Python 3.9.7)
1/1 [==============================] - 0s 36ms/step
cat
1/1 [==============================] - 0s 23ms/step
cat
Snipping Tool
Screenshot copied to clipboard and saved
Select here to mark up and share the image
Share Code Link Explain Code Comment Code Code Chat Blackbox Kodezi
Ln 1, Col 1 (146 selected) Spaces: 4 CRLF Cell 14 of 46 Blackbox





Code (in text):

import tensorflow as tf

from keras.preprocessing.image import ImageDataGenerator

train_gen=ImageDataGenerator(rescale=1.0/255,shear_range=0.2

,zoom_range=0.2,horizontal_flip=True)

train_data=train_gen.flow_from_directory("D:/cat_dog_images/training_set/training_set",target_size=(64,64),batch_size=32,class_mode='binary')

test_datagen = ImageDataGenerator(rescale=1./255)

validation_generator = test_datagen.flow_from_directory(

```
        "D:/cat_dog_images/test_set/test_set",
        target_size=(64,64),
        batch_size=32,
        class_mode='binary')
Cnn=tf.keras.models.Sequential()

Cnn.add(tf.keras.layers.Conv2D(filters=32,kernel_size=3,activation='relu',input_shape=[64,64,3]))
Cnn.add(tf.keras.layers.BatchNormalization())
Cnn.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))
Cnn.add(tf.keras.layers.Dropout(0.25))

Cnn.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,activation='relu'))
Cnn.add(tf.keras.layers.BatchNormalization())
Cnn.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))
Cnn.add(tf.keras.layers.Dropout(0.25))

Cnn.add(tf.keras.layers.Conv2D(filters=128, kernel_size=3, activation='relu'))
Cnn.add(tf.keras.layers.BatchNormalization())
Cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
Cnn.add(tf.keras.layers.Dropout(0.25))

Cnn.add(tf.keras.layers.Flatten())

Cnn.add(tf.keras.layers.Dense(units=256, activation='relu'))
Cnn.add(tf.keras.layers.BatchNormalization())
Cnn.add(tf.keras.layers.Dropout(0.5))

Cnn.add(tf.keras.layers.Dense(units=128,activation='relu'))
Cnn.add(tf.keras.layers.BatchNormalization())
Cnn.add(tf.keras.layers.Dropout(0.5))
```

```
Cnn.add(tf.keras.layers.Dense(units=64, activation='relu'))
Cnn.add(tf.keras.layers.BatchNormalization())
Cnn.add(tf.keras.layers.Dropout(0.5))

Cnn.add(tf.keras.layers.Dense(units=1,activation='sigmoid'))

Cnn.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
Cnn.summary()

history=Cnn.fit(x=train_data,validation_data=validation_generator,epochs=30)

import matplotlib.pyplot as plt
import matplotlib.image as mpimg

import numpy as np
from keras.preprocessing import image
for i in range(1,13):

test_image=image.load_img("D:\cat_dog_images\Prediction\Pred"+str(i)+".jpg",target_size=(64,64))

  test_image=image.img_to_array(test_image)
  test_image=np.expand_dims(test_image,axis=0)
  result=Cnn.predict(test_image)#we predict the class this image belongs to
  if result[0][0]==1:
    prediction='dog'
  else:
    prediction='cat'
  print(prediction)
  img = mpimg.imread("D:\cat_dog_images\Prediction\Pred"+str(i)+".jpg")

  # Display the image using matplotlib.pyplot.imshow
```

```
    plt.imshow(img)
    plt.axis('off')  # Hide axis
    plt.show()
plt.plot(history.history['accuracy'])
plt.plot(history.history['loss'])
```

CONCLUSION:

We have successfully implemented a CNN model for classifying images of cats and dogs. The model demonstrates good performance, achieving high accuracy(85%,can increase but must take note of overfitting and optimal number of epochs, increasing amount of data for each class i.e cat and dog)on both the training and validation sets. Through this project, we have showcased the effectiveness of deep learning techniques, particularly CNNs, in image classification tasks. Further optimizations and fine-tuning could potentially enhance the model's performance for real-world applications.

The successful development of the CNN model for classifying cats and dogs has broader implications in various domains, including pet identification, wildlife monitoring, and security surveillance. The model can be deployed in real-world scenarios to automate tasks such as pet registration, animal counting, thereby enhancing operational efficiency and accuracy.