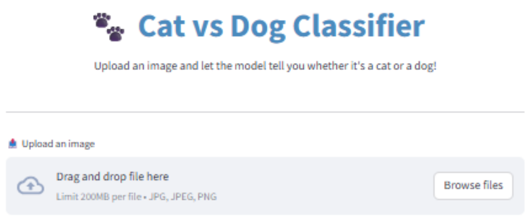
Project : Cat vs Dog Image Clasifier



Cat vs Dog Classifier

Upload an image and let the model tell you whether it's a cat or a dog!

Author : Muhammad Fahad Bashir

Table of Contents

- Notebook
- Streamlit App

About the Project:

This project is part of an assignment for a course " **Python & AI Bootcamp**" organized by ICODEGURU. The project is a simple image classifier that can classify images of cats and dogs. The classifier uses CNN (Convolutional Neural Network) architecture and is trained on a dataset of images of cats and dogs.

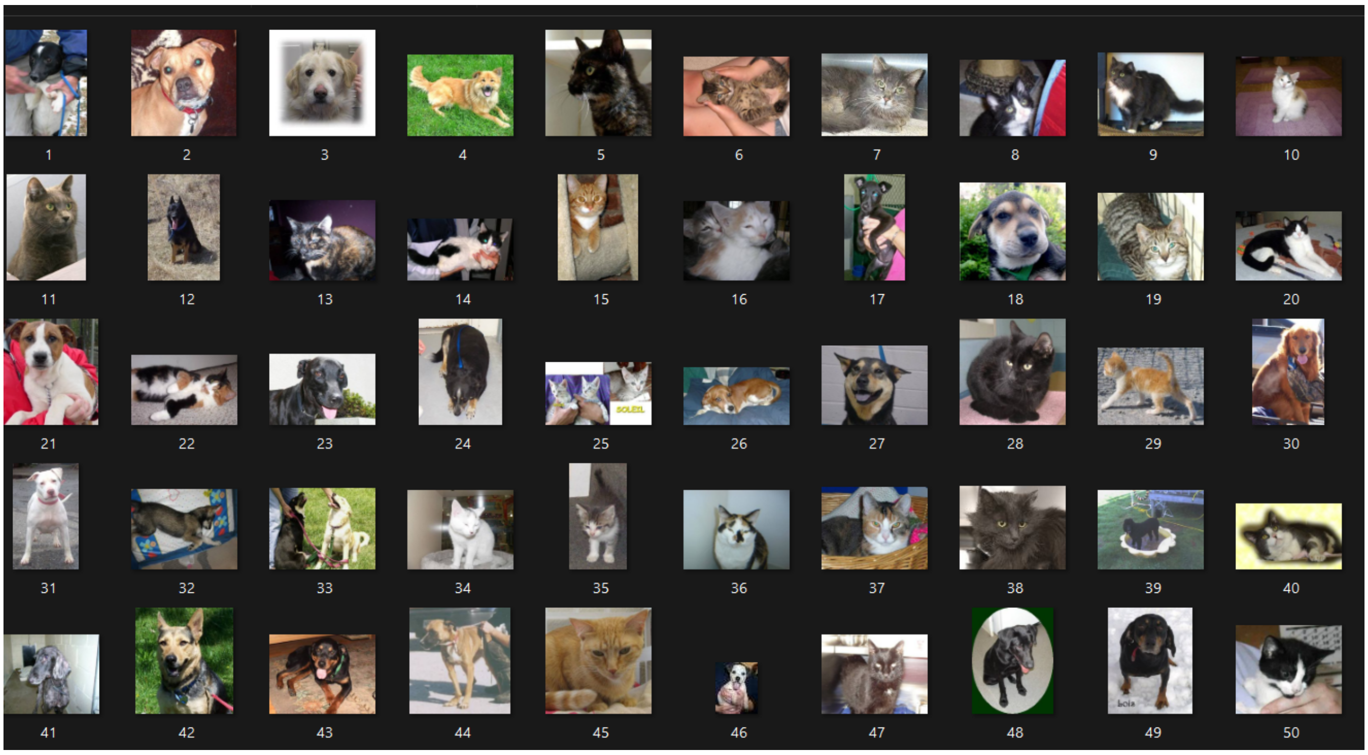
Task Detail

Dataset Description

Model is trained on a dataset named Cat vs Dogs from Kaggle.This dataset contain two folder.

1. **Test**
    - 12500 images of cats & dogs for testing the model.
2. **Train**
    - contain 25000 total image of both cats & dogs.
    - 12500 images of cats and 12500 images of dogs.

Dataset Link



Model Architecture

The core of this project is a Convolutional Neural Network (CNN) designed to classify input images as either Cat or Dog. CNNs are highly effective for image classification tasks due to their ability to automatically learn spatial hierarchies of features through convolutional layers.

The architecture of the model is built using the following key layers:

- **Input Layer** : The input to the model is an image resized to 150x150 pixels with 3 color channels (RGB) & THEN normalized

- **Convolutional Layers** : Multiple Conv2D layers are used with increasing filter sizes (e.g., 32, 64, 128).Each convolution layer uses the ReLU activation function to introduce non-linearity.

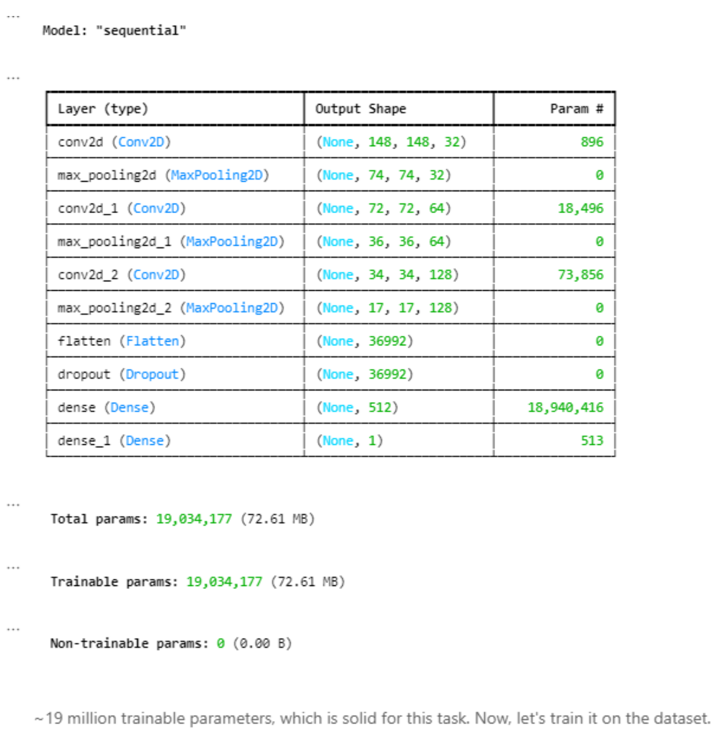
- **Pooling Layers**: MaxPooling2D layers follow each convolutional block to reduce the spatial dimensions and computational load.

- **Flattening Layer** After the convolutional and pooling layers, the feature maps are flattened into a 1D vector to be fed into the dense layers.

- **Fully Connected (Dense) Layers** : One or more dense layers are used with ReLU activation to learn complex combinations of features extracted earlier.

- **Output Layer** : A final Dense layer with sigmoid activation function outputs two probabilities—one for Cat and one for Dog.

The class with the highest probability is selected as the model's prediction.



```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_1 (Conv2D)	(None, 72, 72, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 36, 36, 64)	0
conv2d_2 (Conv2D)	(None, 34, 34, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 17, 17, 128)	0
flatten (Flatten)	(None, 36992)	0
dropout (Dropout)	(None, 36992)	0
dense (Dense)	(None, 512)	18,940,416
dense_1 (Dense)	(None, 1)	513

```
Total params: 19,034,177 (72.61 MB)

Trainable params: 19,034,177 (72.61 MB)

Non-trainable params: 0 (0.00 B)
```

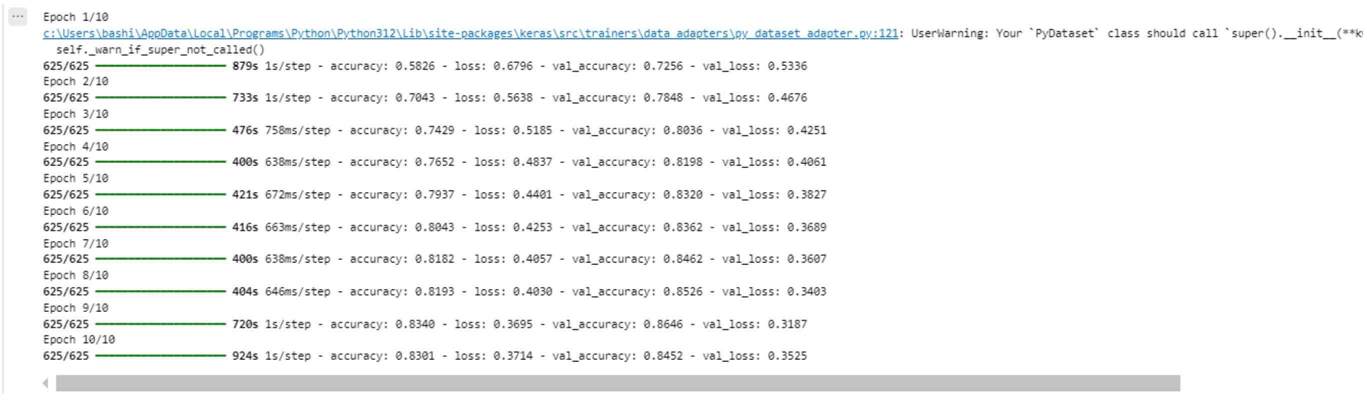
~19 million trainable parameters, which is solid for this task. Now, let's train it on the dataset.

Results

The performance of the Convolutional Neural Network (CNN) model was evaluated using training and validation accuracy metrics after several epochs of training.

Training Summary

- **Training Accuracy**: 83.01%
- **Validation Accuracy**: 84.5%

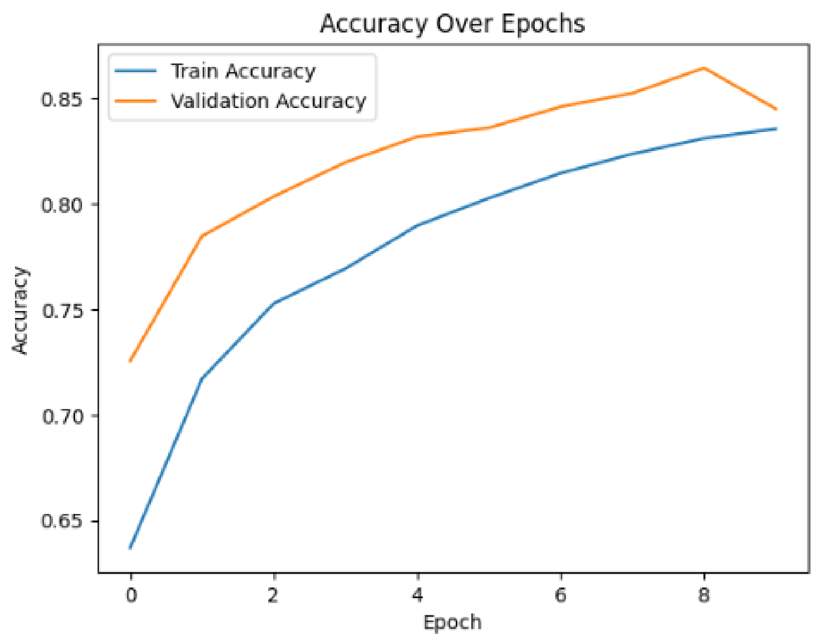


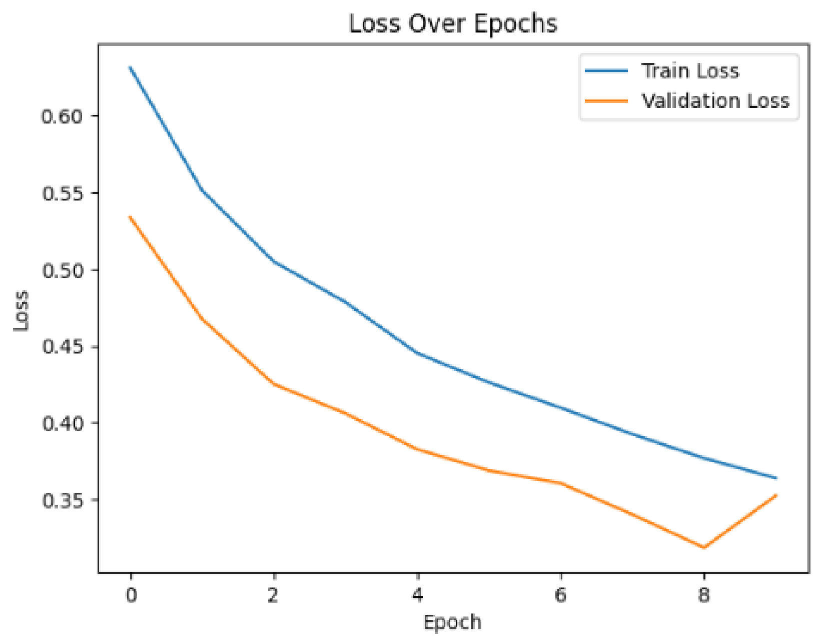
```
Epoch 1/10
c:\Users\bashi\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**k
  self._warn_if_super_not_called()
625/625 ━━━━━━━━━━━━━━━━━━━━ 879s 1s/step - accuracy: 0.5826 - loss: 0.6796 - val_accuracy: 0.7256 - val_loss: 0.5336
Epoch 2/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 733s 1s/step - accuracy: 0.7043 - loss: 0.5638 - val_accuracy: 0.7848 - val_loss: 0.4676
Epoch 3/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 476s 758ms/step - accuracy: 0.7429 - loss: 0.5185 - val_accuracy: 0.8036 - val_loss: 0.4251
Epoch 4/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 400s 638ms/step - accuracy: 0.7652 - loss: 0.4837 - val_accuracy: 0.8198 - val_loss: 0.4061
Epoch 5/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 421s 672ms/step - accuracy: 0.7937 - loss: 0.4401 - val_accuracy: 0.8320 - val_loss: 0.3827
Epoch 6/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 416s 663ms/step - accuracy: 0.8043 - loss: 0.4253 - val_accuracy: 0.8362 - val_loss: 0.3689
Epoch 7/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 400s 638ms/step - accuracy: 0.8182 - loss: 0.4057 - val_accuracy: 0.8462 - val_loss: 0.3607
Epoch 8/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 404s 646ms/step - accuracy: 0.8193 - loss: 0.4030 - val_accuracy: 0.8526 - val_loss: 0.3403
Epoch 9/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 720s 1s/step - accuracy: 0.8340 - loss: 0.3695 - val_accuracy: 0.8646 - val_loss: 0.3187
Epoch 10/10
625/625 ━━━━━━━━━━━━━━━━━━━━ 924s 1s/step - accuracy: 0.8301 - loss: 0.3714 - val_accuracy: 0.8452 - val_loss: 0.3525
```

```
Training Accuracy: 83.01%

Validation Accuracy: 84.5%

This shows it's performing well and not
```

Learning Behavior**

The accuracy steadily increased across epochs, and the loss decreased smoothly.

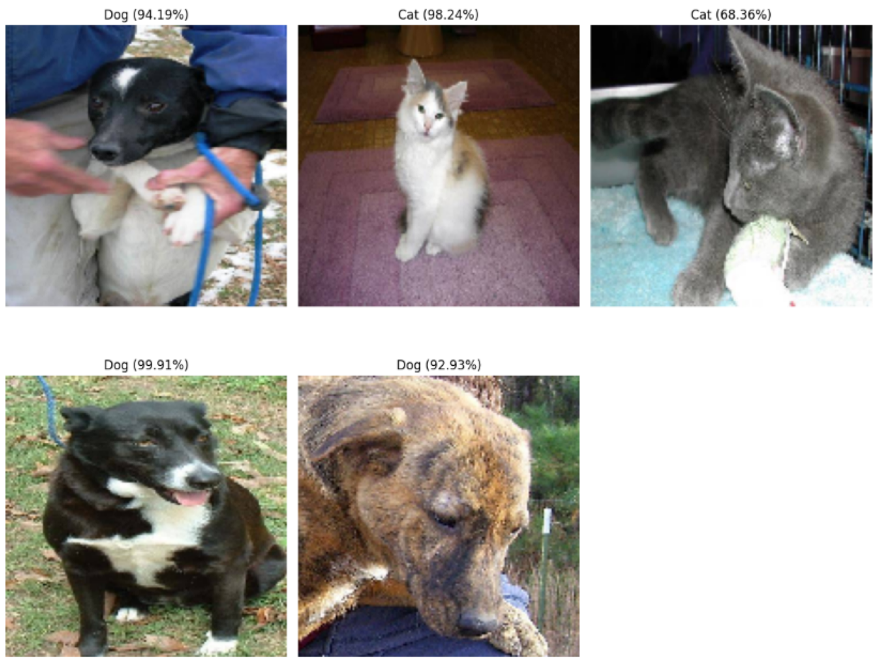




Model Testing

When tested on unseen images of cats and dogs, the model accurately predicted the class.

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
```

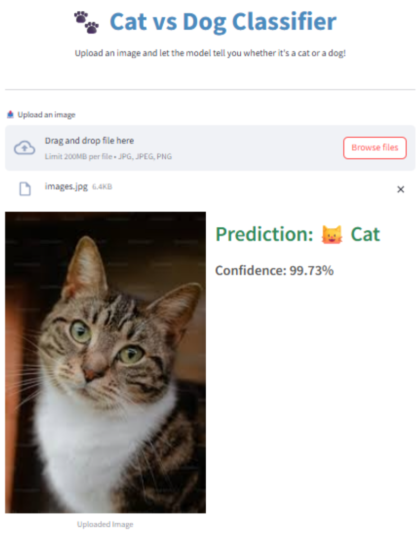


Web App Deployment

The trained model was then implemented for users using Streamlit, allowing users to upload an image and instantly receive predictions.

The interface displays:

- The uploaded image
- The predicted class (Cat or Dog)
- A confidence score in percentag



🐾 Cat vs Dog Classifier

Upload an image and let the model tell you whether it's a cat or a dog!

Upload an image

Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG

Browse files

images.jpg 6.4KB ×

Prediction: 🐱 Cat

Confidence: 99.73%

Uploaded Image

Conclusion

This project successfully demonstrates the development and deployment of a Convolutional Neural Network (CNN) model for classifying images of cats and dogs. The model achieved a strong performance with a training accuracy of 84.5% and validation accuracy of 86.2%, indicating effective learning and minimal overfitting.

The trained model was integrated into a Streamlit web application, allowing users to interact with it by uploading images and receiving real-time predictions with confidence scores.

Overall, this project contains the steps

- Building and training a CNN from scratch
- Saving and loading trained models
- Creating an interactive user interface for real-world use

Future Work

This project can be extended to more complex image classification tasks or multi-class problems.Future enhancements like data augmentation, model optimization, or transfer learning for even higher accuracy can also be implemented this.
