

Machine Learning Project

Car Images Classification By Training Model

Deep Learning Steps

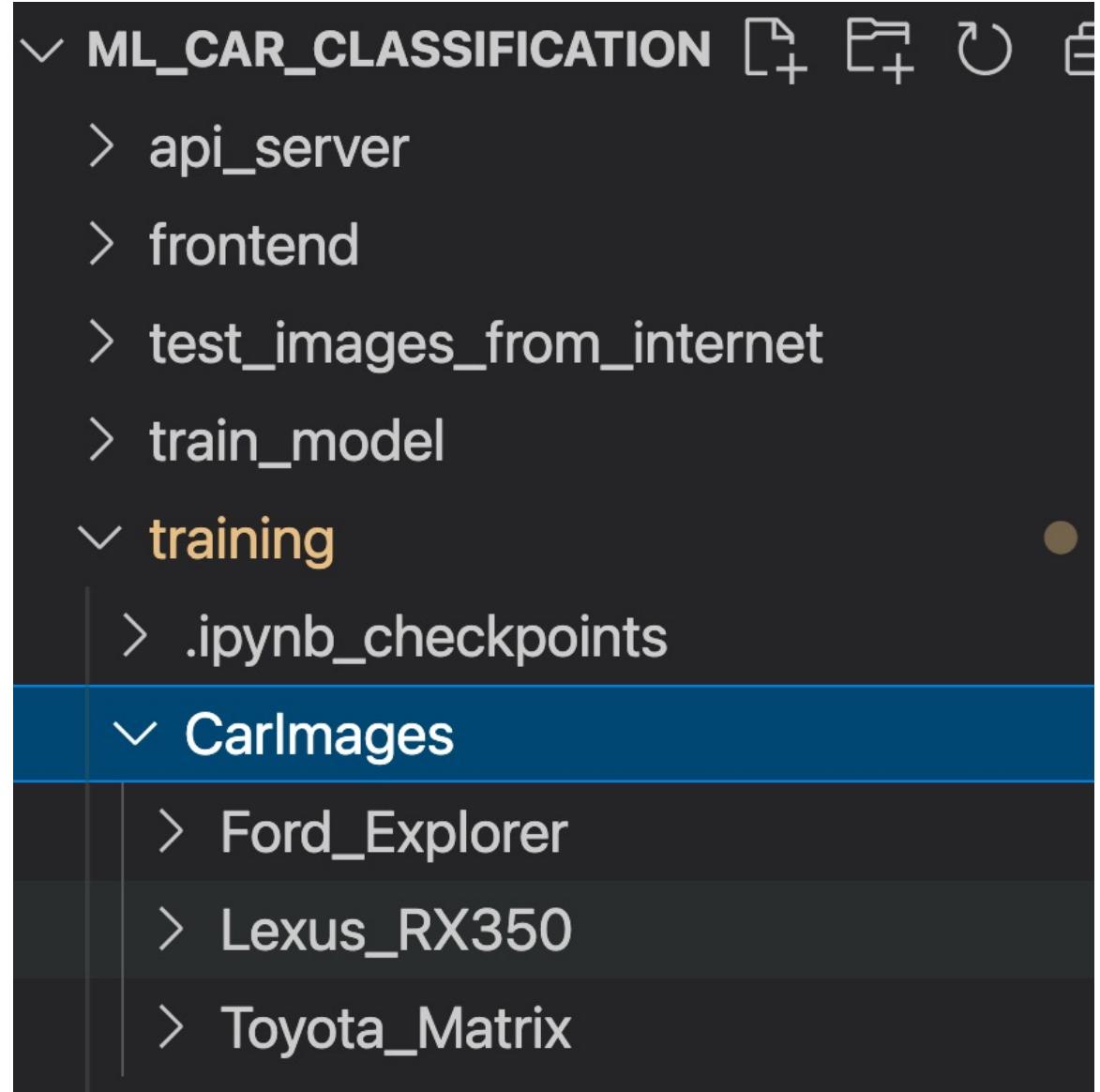
1. Data collection
2. Data Preparation
3. Training the Model
4. Serving Trained Model with API Server
5. Consuming the Model for Prediction

1. Data Collection

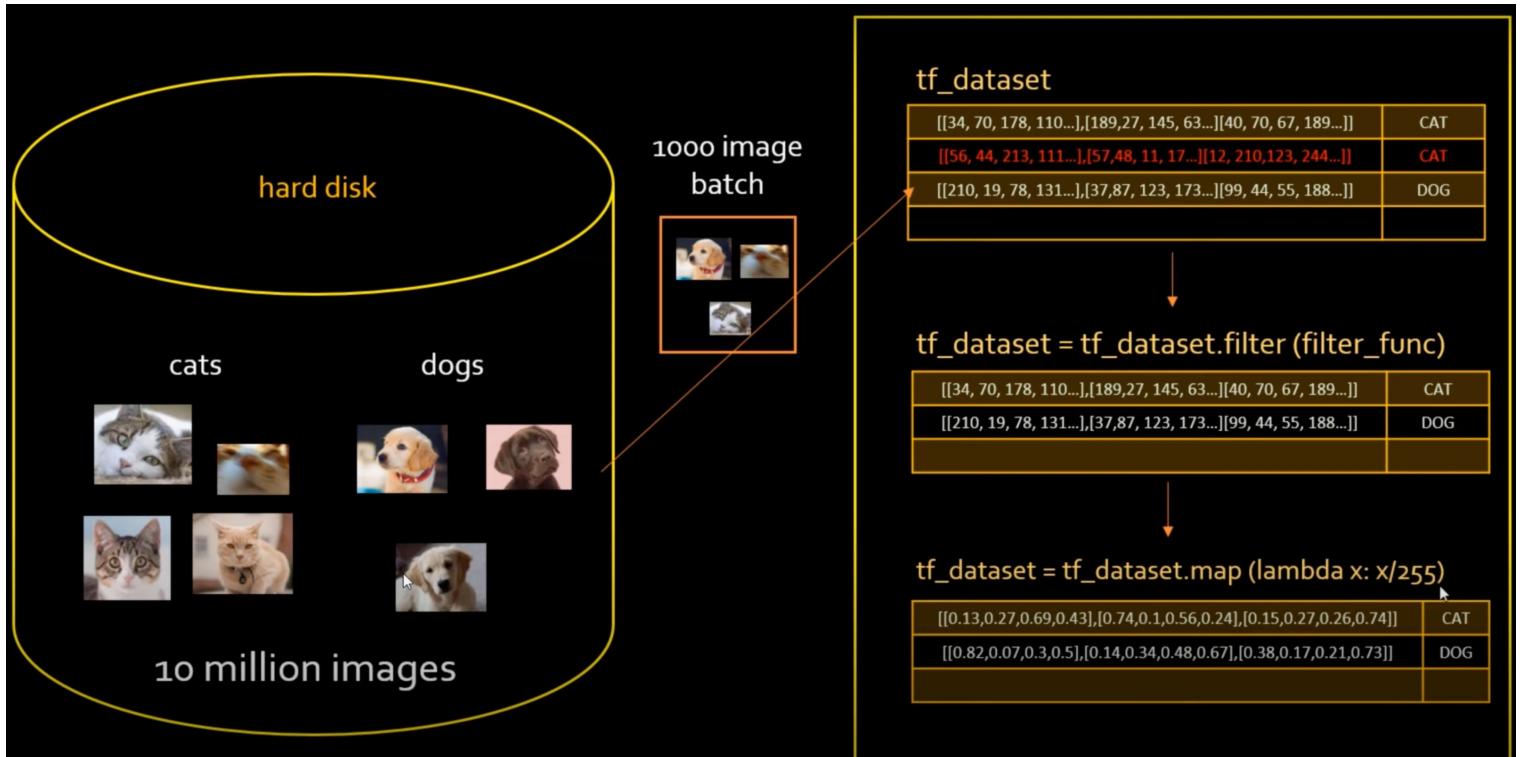
We use different methods to collect data

Data Collection from Internet

- We download the Data from the Internet
- Total Files = 193
- Classified into 3 classes
 1. Ford_Explorer
 2. Lexus_RX350
 3. Toyota_Matrix



Loading data
with
tf_dataset
batch_size=
10



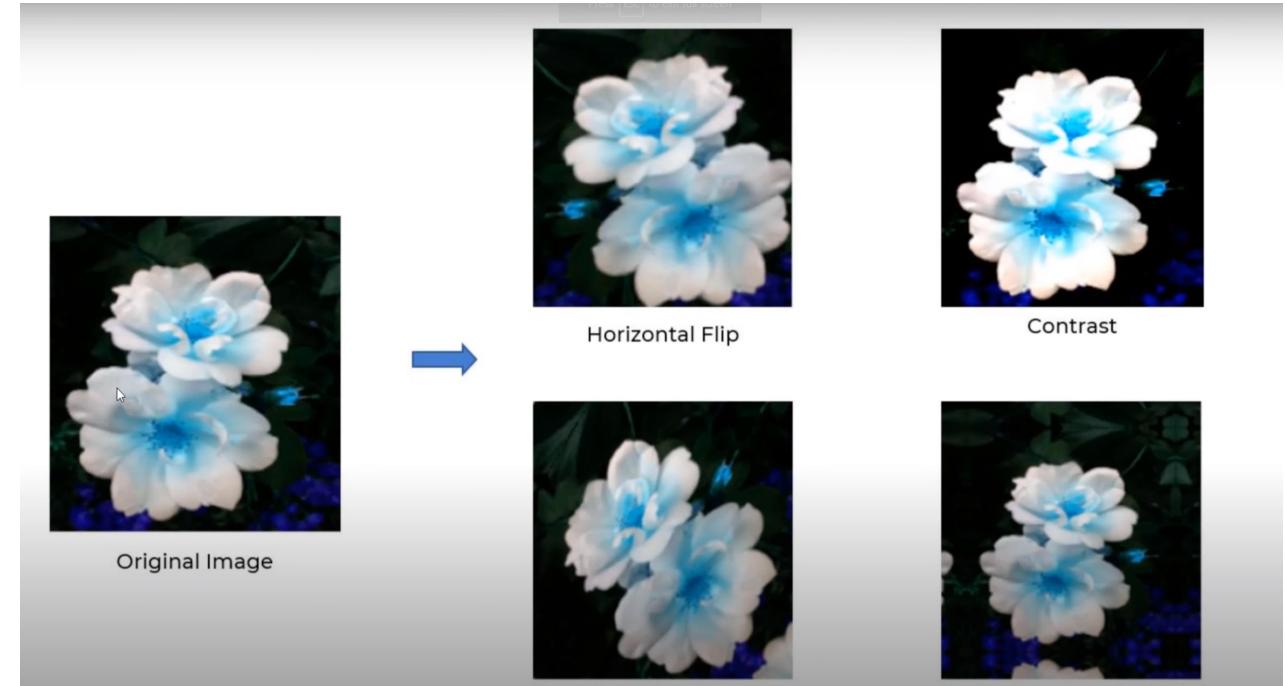
2. Data Preprocessing

We multiply one images into 4-5 images but with different shape and look

Data augmentation
Layer is used to
multiply an image

- Flip
- Random Zoom
- Random Rotate
- Resize
- Rescale

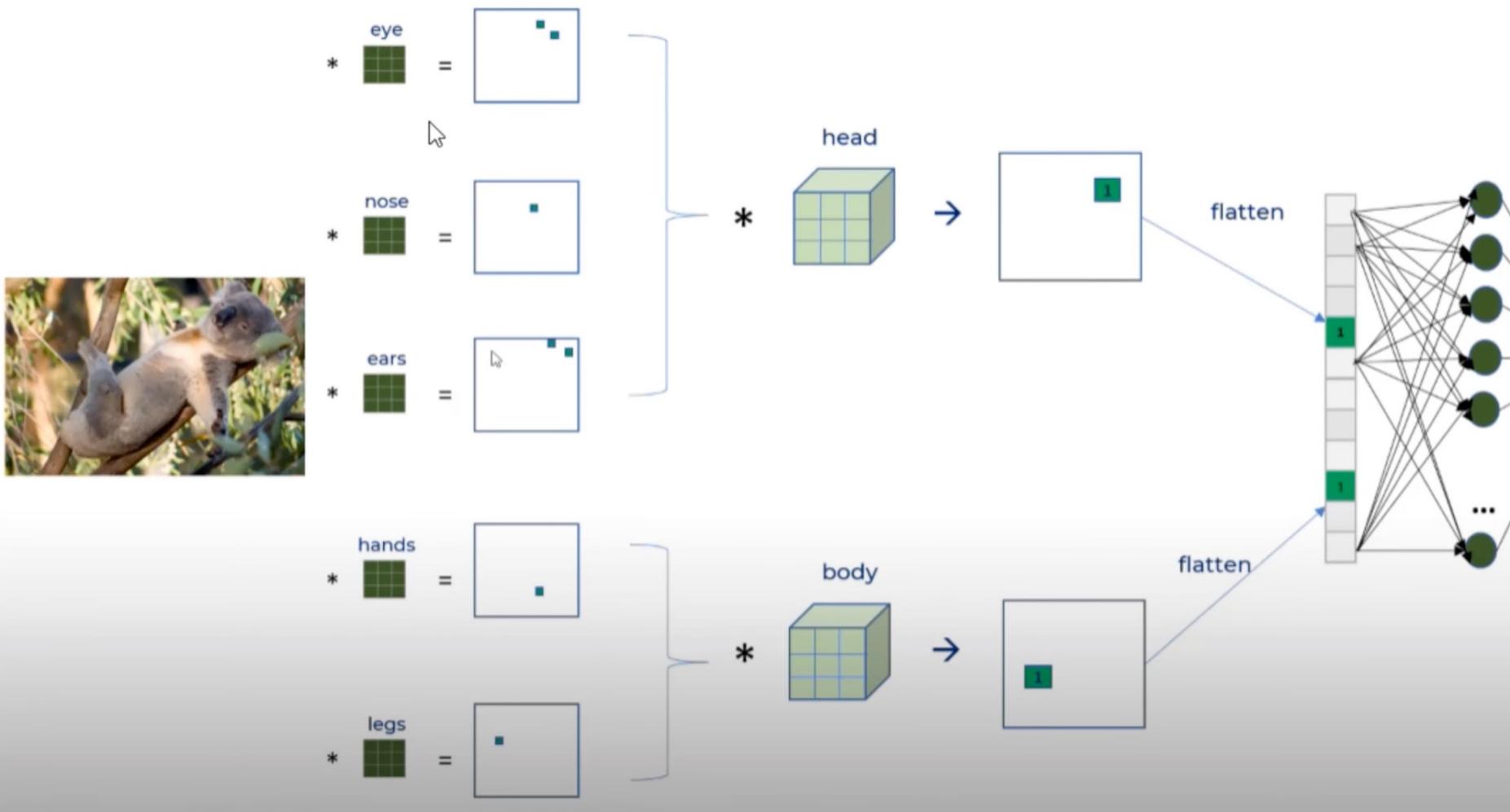
- 80% of data using for training
- 10% for validation
- 10% for testing



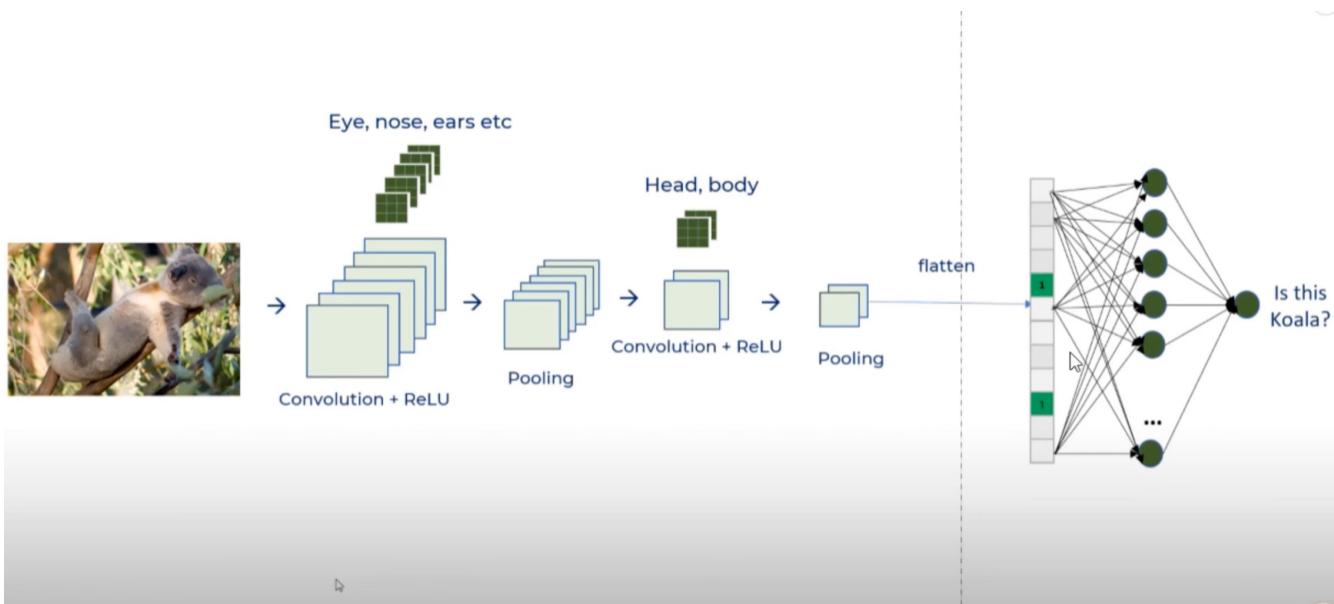
3. Training the Model

We are using Convolutional Neural Network (CNN)
Algorithm to train a model using training sample

Convolutional Neural Network Algorithm



Convolutional Neural Network Layers



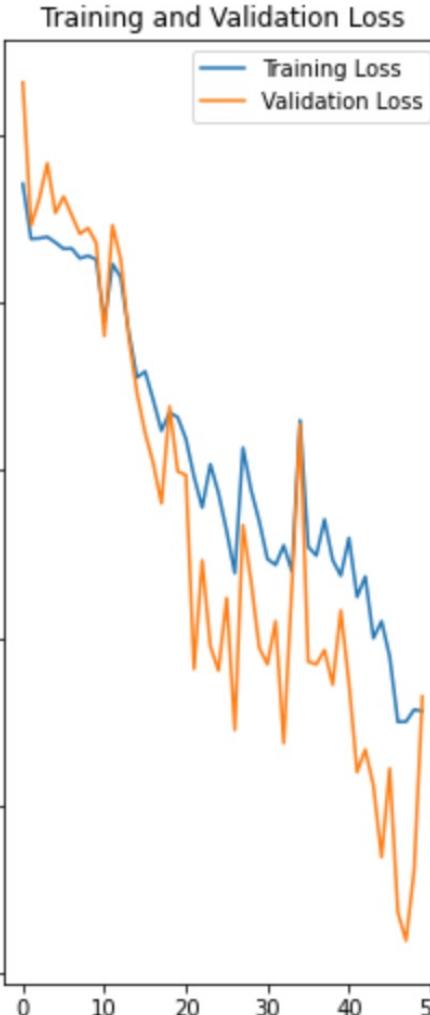
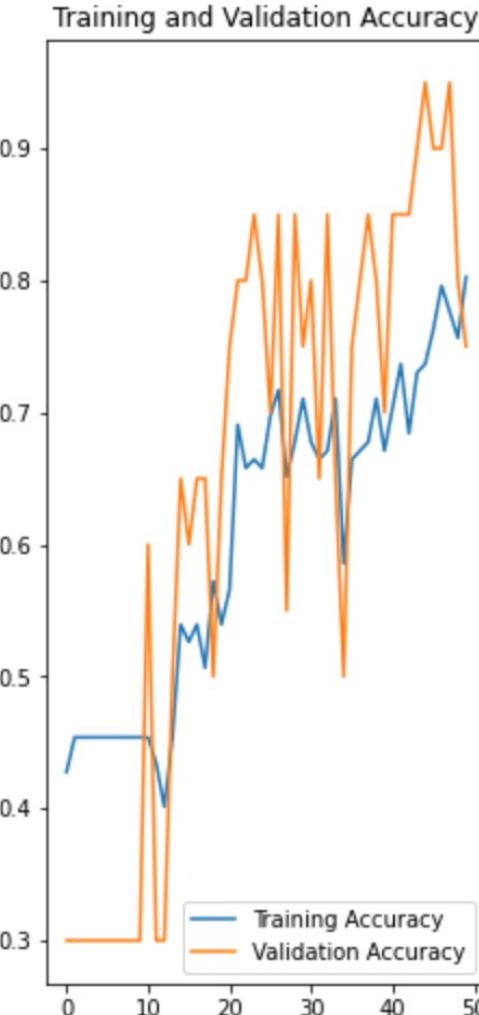
```
model = models.Sequential([
    resize_and_rescale,
    data_augmentation,
    layers.Conv2D(32, (3,3), activation='relu', input_shape=input_shape),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(n_classes, activation='softmax'),
])
I
model.build(input_shape=input_shape)
```

Training The Model 50 rounds and validation

```
-- . . .  
Epoch 43/50  
16/16 [=====] - 87s 5s/step - loss: 0.6734 - accuracy: 0.6842 - val_loss: 0.4667 - val_accuracy: 0.8500  
Epoch 44/50  
16/16 [=====] - 84s 5s/step - loss: 0.5997 - accuracy: 0.7303 - val_loss: 0.4238 - val_accuracy: 0.9000  
Epoch 45/50  
16/16 [=====] - 87s 5s/step - loss: 0.6199 - accuracy: 0.7368 - val_loss: 0.3381 - val_accuracy: 0.9500  
Epoch 46/50  
16/16 [=====] - 85s 5s/step - loss: 0.5776 - accuracy: 0.7632 - val_loss: 0.4440 - val_accuracy: 0.9000  
Epoch 47/50  
16/16 [=====] - 83s 5s/step - loss: 0.4997 - accuracy: 0.7961 - val_loss: 0.2722 - val_accuracy: 0.9000  
Epoch 48/50  
16/16 [=====] - 85s 5s/step - loss: 0.5000 - accuracy: 0.7763 - val_loss: 0.2386 - val_accuracy: 0.9500  
Epoch 49/50  
16/16 [=====] - 86s 5s/step - loss: 0.5144 - accuracy: 0.7566 - val_loss: 0.3226 - val_accuracy: 0.8000  
Epoch 50/50  
16/16 [=====] - 85s 5s/step - loss: 0.5126 - accuracy: 0.8026 - val_loss: 0.5300 - val_accuracy: 0.7500
```

```
history = model.fit(  
    train_ds,  
    batch_size=BATCH_SIZE,  
    validation_data=val_ds,  
    verbose=1,  
    epochs=50,  
)
```

Accuracy
Increase
steadily
toward 90%



Predicate with Test Sample

```
In [34]: plt.figure(figsize=(15, 15))
for images, labels in test_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))

        predicted_class, confidence = predict(model, images[i].numpy())
        actual_class = class_names[labels[i]]

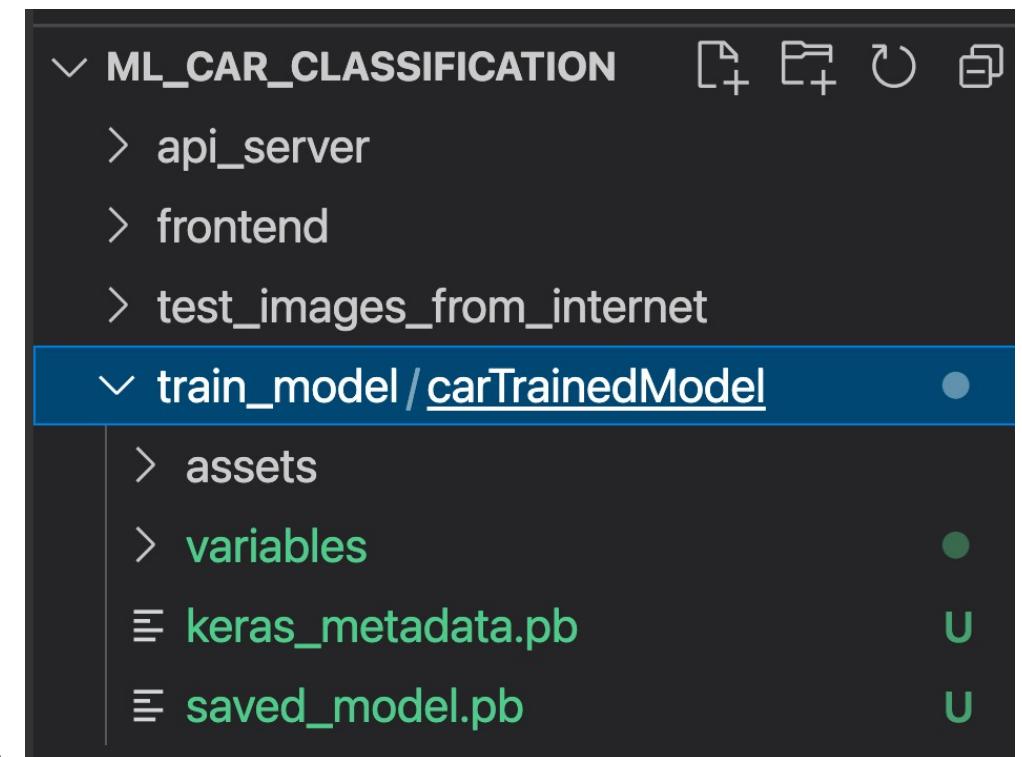
        plt.title(f"Actual: {actual_class},\n Predicted: {predicted_class}.\n Confidence: {confidence}%")
        plt.axis("off")
```



Saving Trained Model

Saving the Model

```
In [36]: model.save(f"../train_model/carTrainedModel")  
INFO:tensorflow:Assets written to: ../train_model/carTrainedModel
```



API Server

We are using Python FastAPI Server to Serve
the post-api with tensorflow (tf)

Serving Trained Model with API Server

```
MODEL = tf.keras.models.load_model("../train_model/carTrainedModel")

CLASS_NAMES = ["Ford_Explorer", "Lexus_RX350", "Toyota_Matrix"]

@app.get("/")
async def ping():
    return "Welcome to Car Classification API"

def read_file_as_image(data) -> np.ndarray:
    image = np.array(Image.open(BytesIO(data)))
    return image

@app.post("/predict")
async def predict(
    file: UploadFile = File(...)
):
    image = read_file_as_image(await file.read())
    img_batch = np.expand_dims(image, 0)

    predictions = MODEL.predict(img_batch)

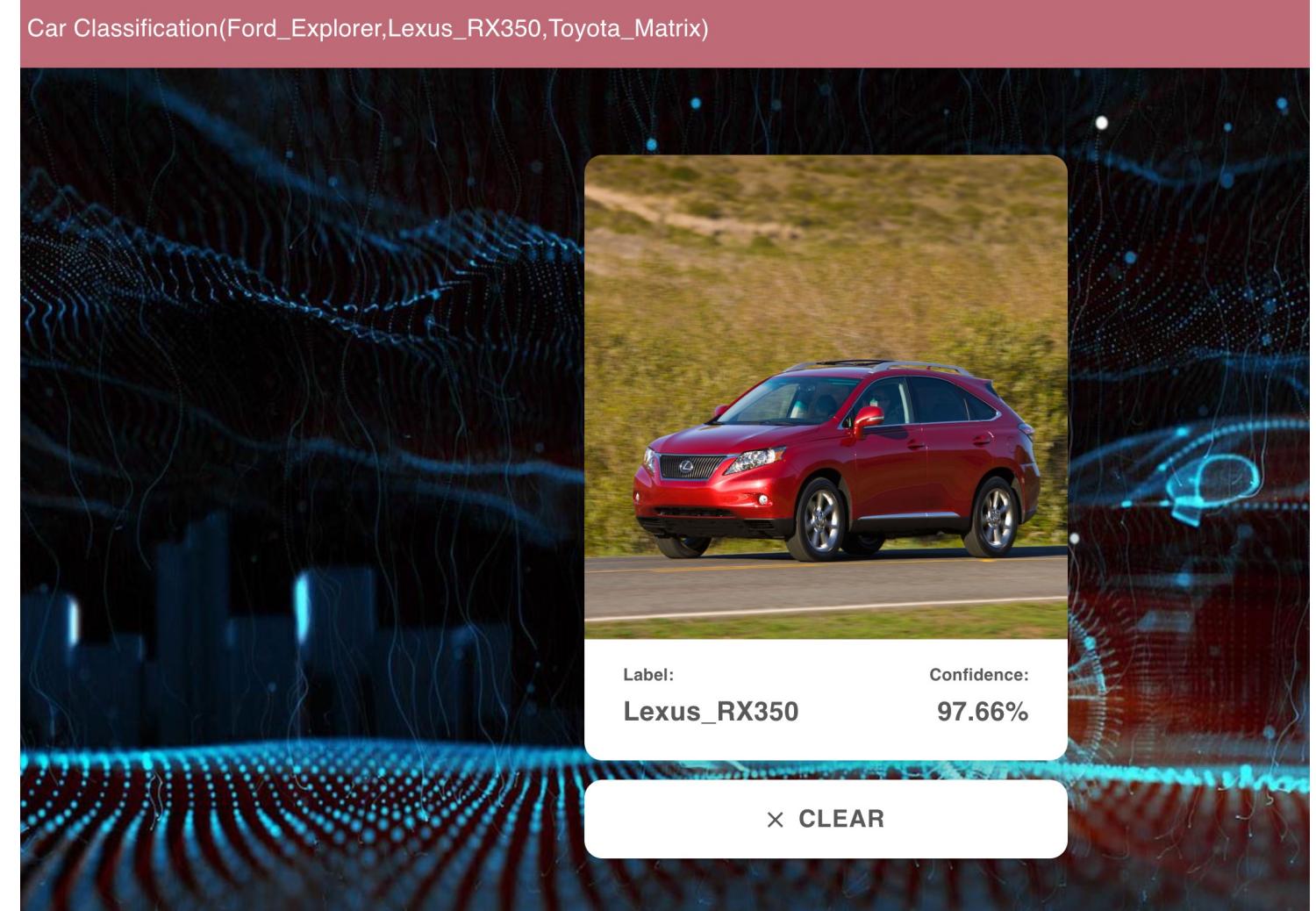
    predicted_class = CLASS_NAMES[np.argmax(predictions[0])]
    confidence = np.max(predictions[0])
    return {
        'class': predicted_class,
        'confidence': float(confidence)
    }

if __name__ == "__main__":
    uvicorn.run(app, host='localhost', port=8080)
```

4. Consuming API

We are React Javascript Simple App, as simple application.

Serving Application with React JS



Thank You