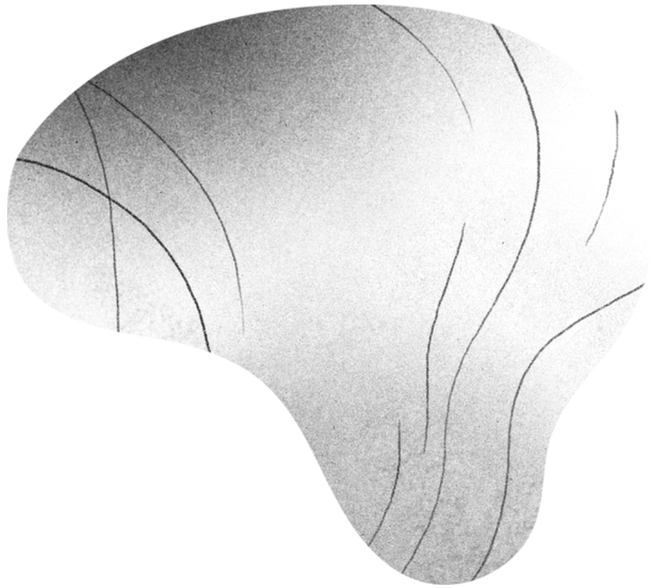
BRAIN TUMOR MRI IMAGES CLASSIFICATION USING DEEP LEARNING MODELS

AGENDA



01 Problem Statement

02 Proposed solution

03 Dataset

04 Methodology

05 Results

06 Conclusion and future work

Problem statement

Challenges in Brain Tumor Identification

- Brain tumor complexity:
  - Size
  - Location
- Diagnostic challenges:
  - Time Consuming.
  - Human error.
  - Subjective and vary based on experience.

Proposed Solution

1. **Data Collection and Preparation**
    - Collecting dataset of MRI images labeled with different types of brain tumors.
    - Apply some preprocessing techniques on the images to best fit the models
2. **Model Selection and Design**
    - Utilize deep learning models to classify the images
3. **Training the Model**
    - Training the models on our data
    - Perform hyperparameter tuning
4. **Model Evaluation and comparison**

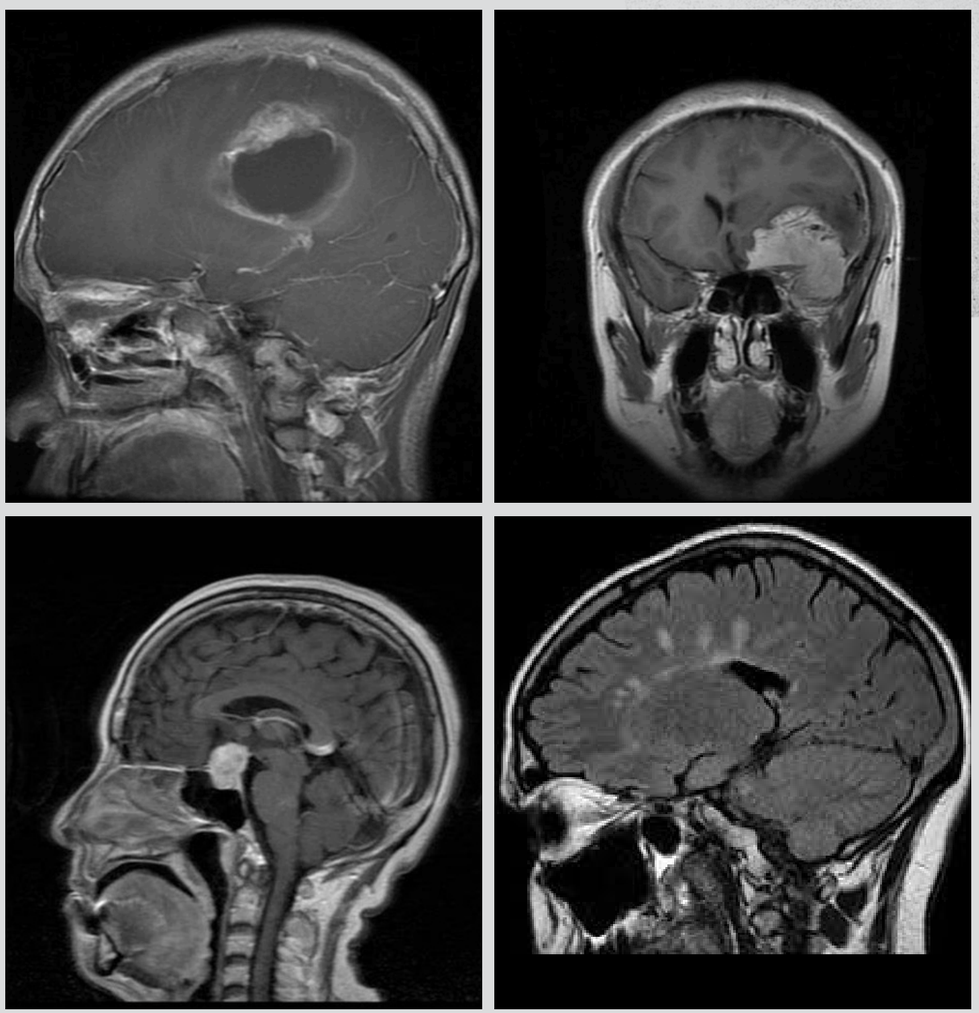
Dataset

Dataset Description

Dataset Shape

Dataset description

- Brain Tumor (MRI) Images dataset collected from Kaggle.
- Divided into 4 classes:
  - Giloma Tumor Images.
  - Meningioma Tumor Images
  - Pituitary Tumor Images
  - No Tumor Images
- Total number of Images is 3264 Image



Dataset shape

	Training	Testing	Total
Glioma Tumor	826	100	926
Meningioma Tumor	822	115	937
Pituitary Tumor	827	74	901
No Tumor	395	105	500
Total	2870	394	3264

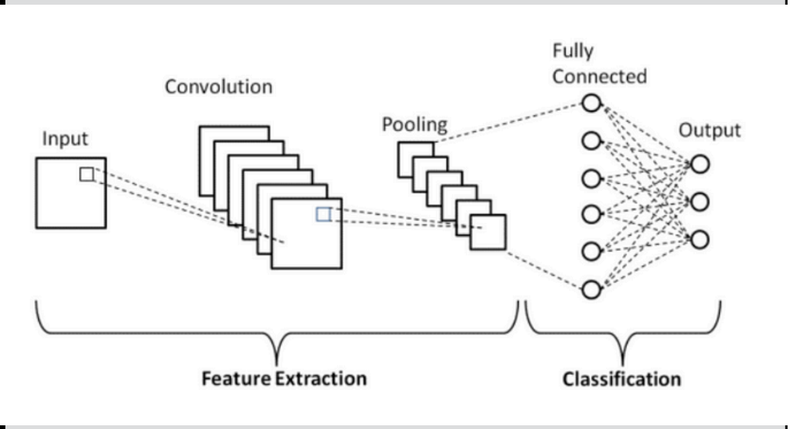
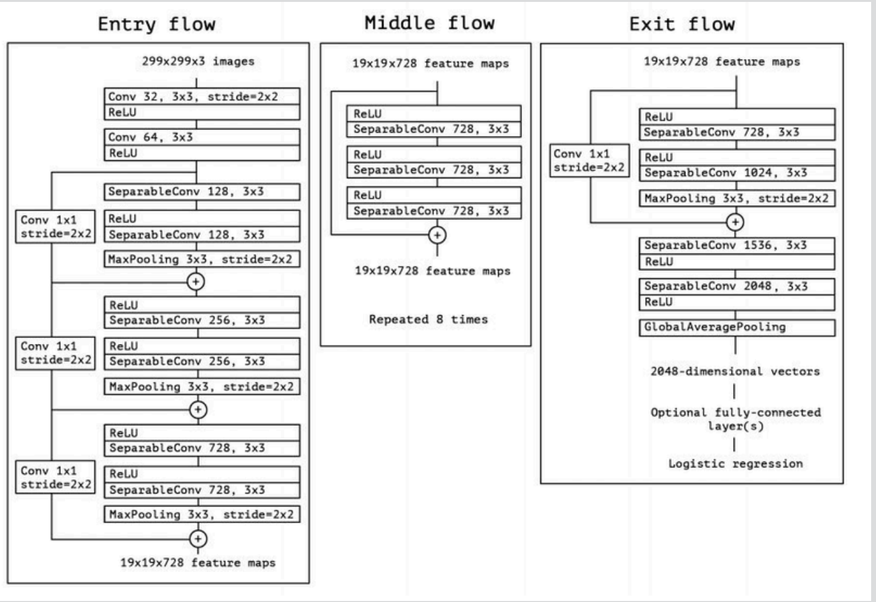
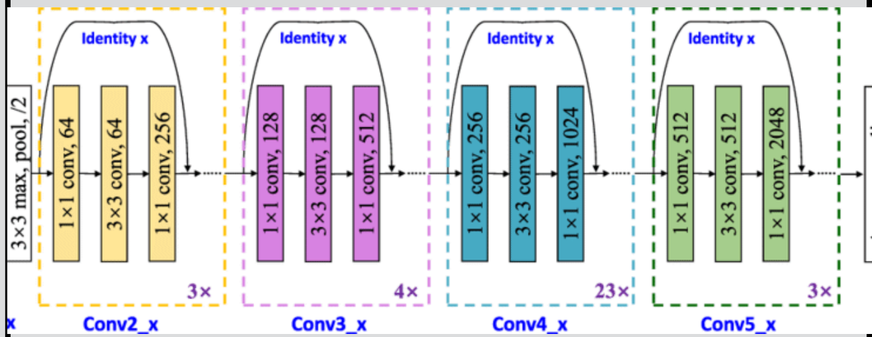
Methodology

- Models Planning
- Data Preparation
- Model Implementation and Training

Model Planning

	CNN	Xception	ResNet101
Overview	General deep learning architecture for image classification	Deep CNN architecture. Advanced Inception architecture with depthwise separable convolutions	Deep residual network with 101 layers developed by Microsoft.
Architecture	Convolutional layers, pooling layers, fully connected layers	• Built on depth wise separable convolutions with residual connections. • Contains 36 convolutional layers forming the feature extraction base.	• Residual blocks Consists of 101 layers, leveraging residual connections to ease the training of deep networks. • Residual connections allow the network to learn identity mappings
Advantages	Captures spatial hierarchies, reduced parameters, high accuracy	• high accuracy. • Suitable for medical imaging applications.	• high accuracy. • Suitable for research and medical imaging applications.

Model Planning

CNN	Xception	ResNet101
		

Data Preparation

```
X_train = []
Y_train = []
image_size = 150
labels = ['glioma_tumor','meningioma_tumor','no_tumor','pituitary_tumor']
for i in labels:
    folderPath = os.path.join('../input/brain-tumor-classification-mri/Training',i)
    for j in os.listdir(folderPath):
        img = cv2.imread(os.path.join(folderPath,j))
        img = cv2.resize(img,(image_size,image_size))
        X_train.append(img)
        Y_train.append(i)

for i in labels:
    folderPath = os.path.join('../input/brain-tumor-classification-mri/Testing',i)
    for j in os.listdir(folderPath):
        img = cv2.imread(os.path.join(folderPath,j))
        img = cv2.resize(img,(image_size,image_size))
        X_train.append(img)
        Y_train.append(i)

X_train = np.array(X_train)
Y_train = np.array(Y_train)
```

```
X_train,Y_train = shuffle(X_train,Y_train,random_state=101)
X_train.shape
```

```
(3264, 150, 150, 3)
```

01 Image Data loading

02 Image Resize

03 Label extraction

04 Converting Images to numpy arrays

05 Shuffling the Training images

Implementation and Training

- **Data Split**: Use train_test_split() to divide images and labels into 90% training and 10% testing sets.
- **Models Construction**: Add layers and build each model and finetuning pre-trained models.
- **Models Summary review**
- **Models Optimization**
- **Model Training**: Fit the data to the model, train with 90% training and 10% validation set.
- **Results Calculation and evaluation**

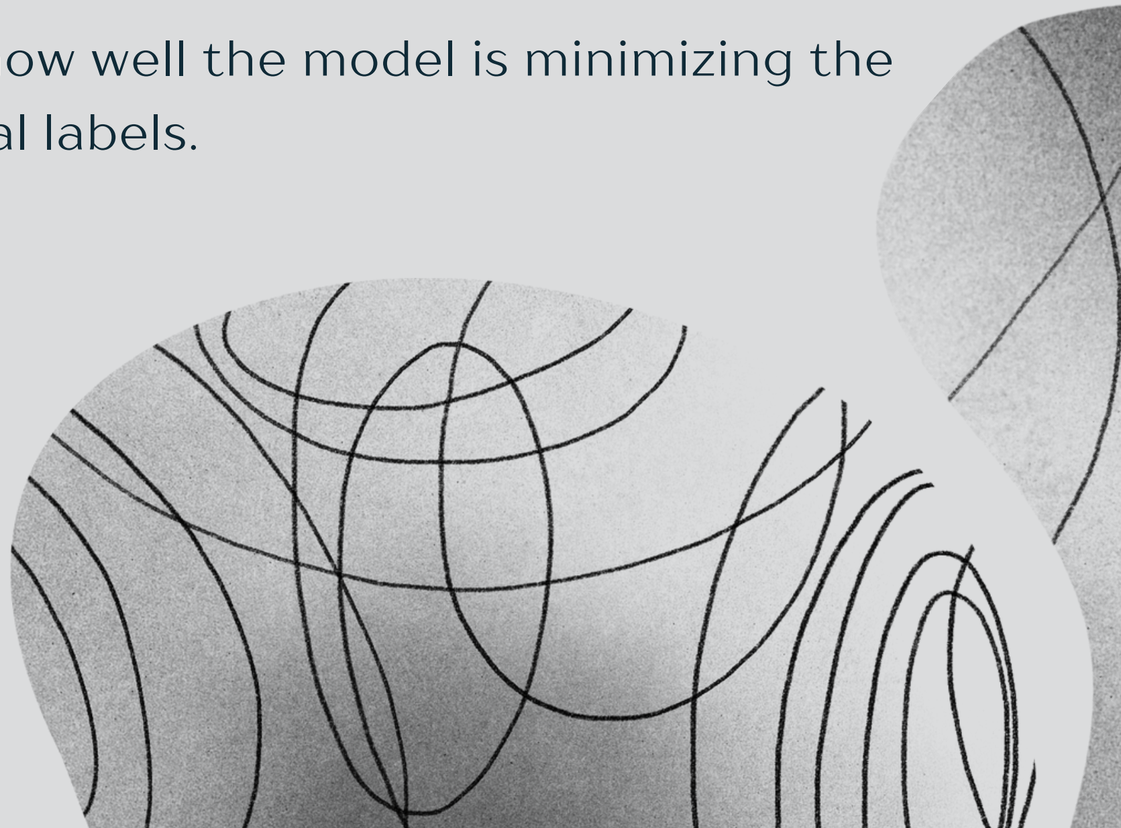
Results & Discussion

Evaluation Plan

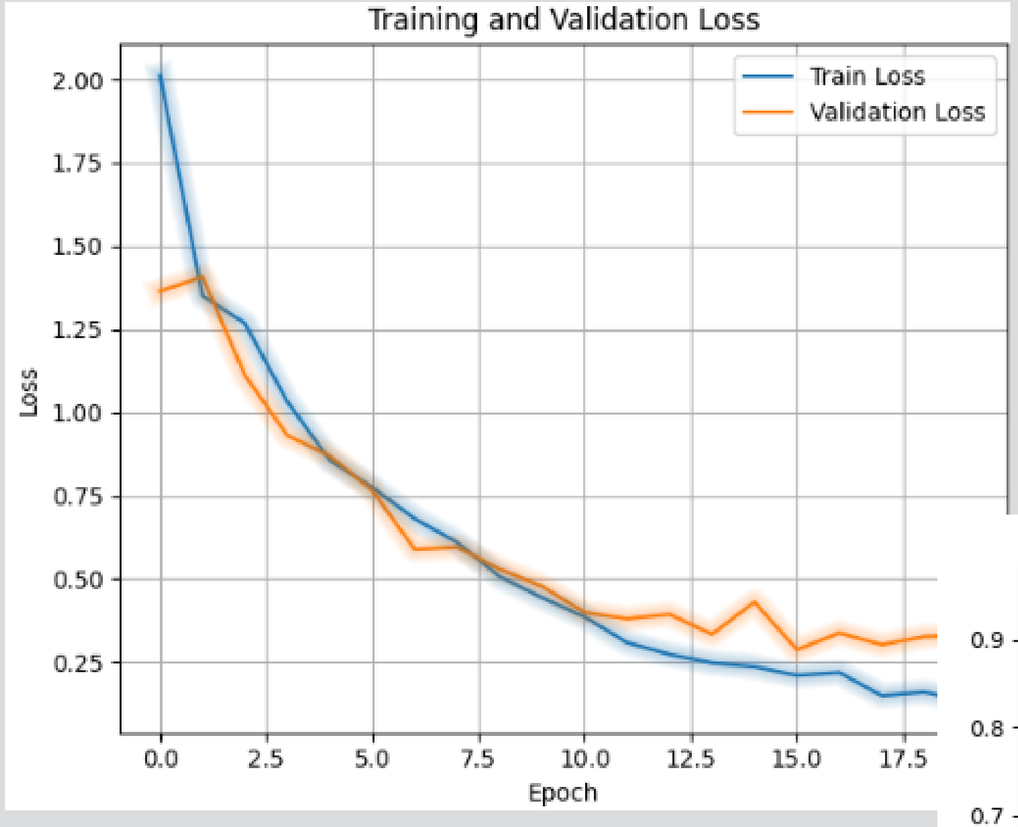
Models Result

Evaluation plan

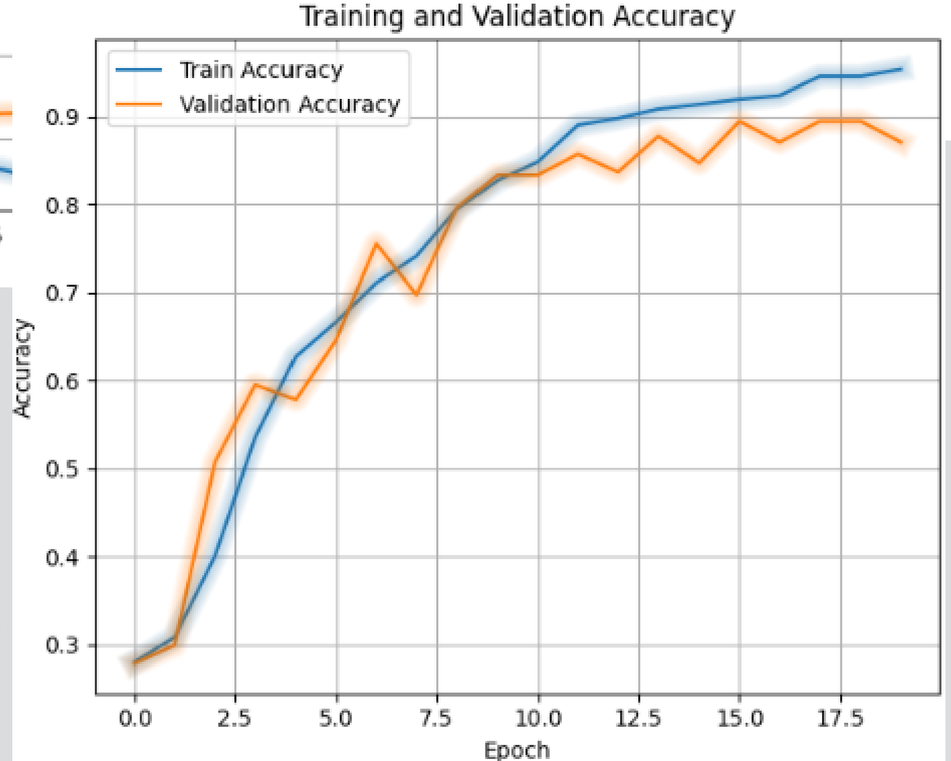
- ## Model evaluation
  - **Accuracy:** It measures the proportion of correctly classified instances out of the total number of instances
  - **Loss Function Value**: provides insight into how well the model is minimizing the difference between the predicted and actual labels.
  - **confusion matrices.**
  - **Important scores:** precision, recall, F1
  - **Models validation data**
- ## Model Testing
  - we tested the model on unseen data



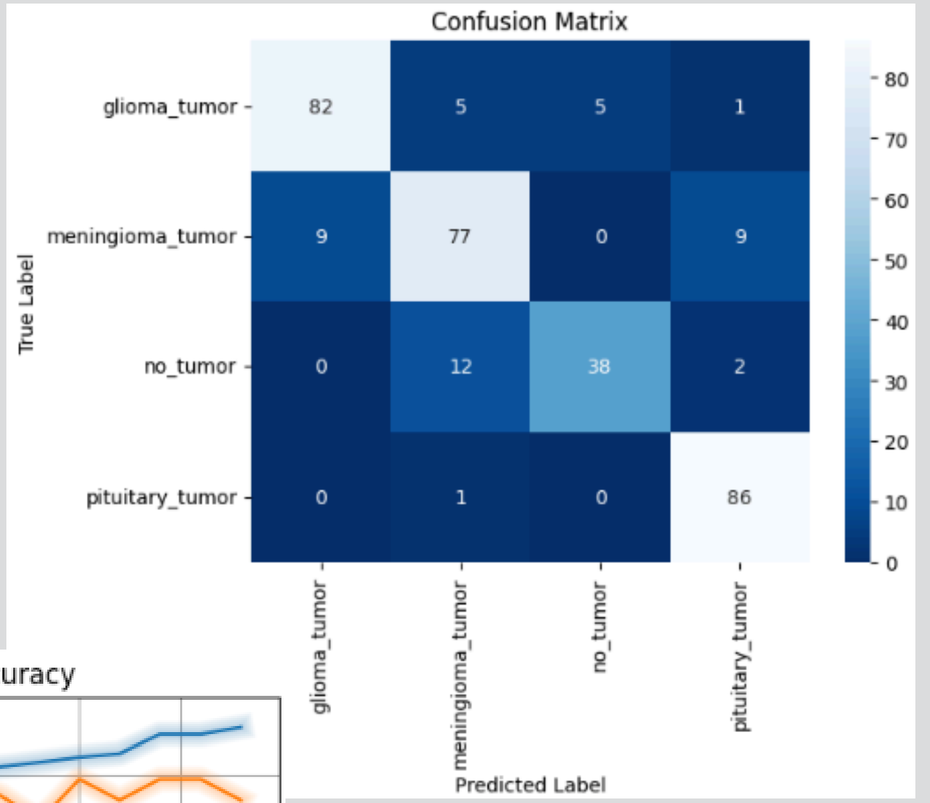
Model Results (CNN)



LOSS FUNCTION GRAPH 1% TRAINING - 3%VAL



ACCURCY GRAPH 95% TRAINING - 87% VAL



Prediction result (CNN)

TESTING ON UNSEEN DATA

```
#labels = ['glioma_tumor','meningioma_tumor','no_tumor','pituitary_tumor']
img = cv2.imread('../input/brain-tumor-classification-mri/Testing/glioma_tumor/image(20).jpg')
img = cv2.resize(img,(150,150))
img_array = np.array(img)
img_array.shape
```

```
(150, 150, 3)
```

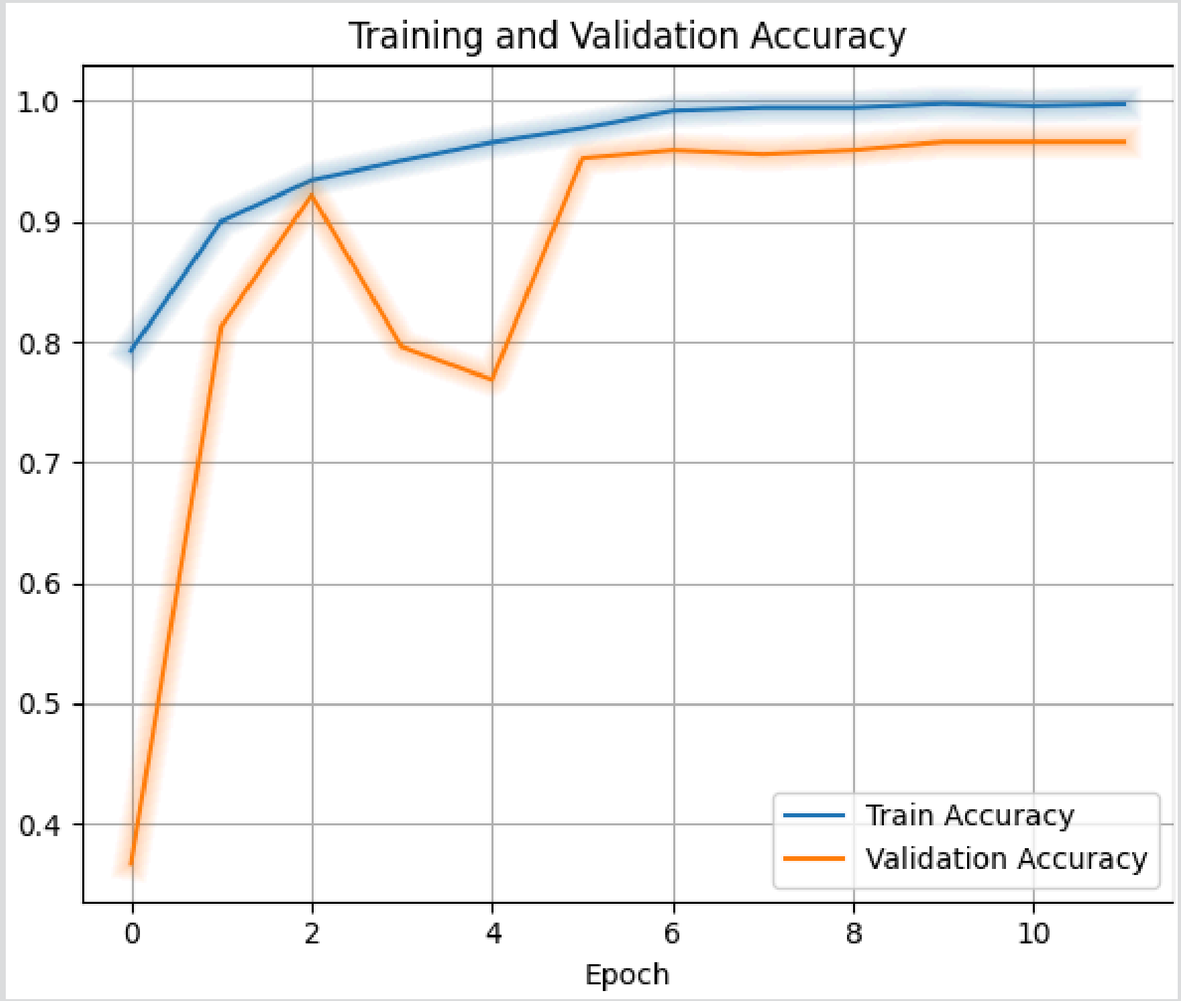
```
img_array = img_array.reshape(1,150,150,3)
img_array.shape
```

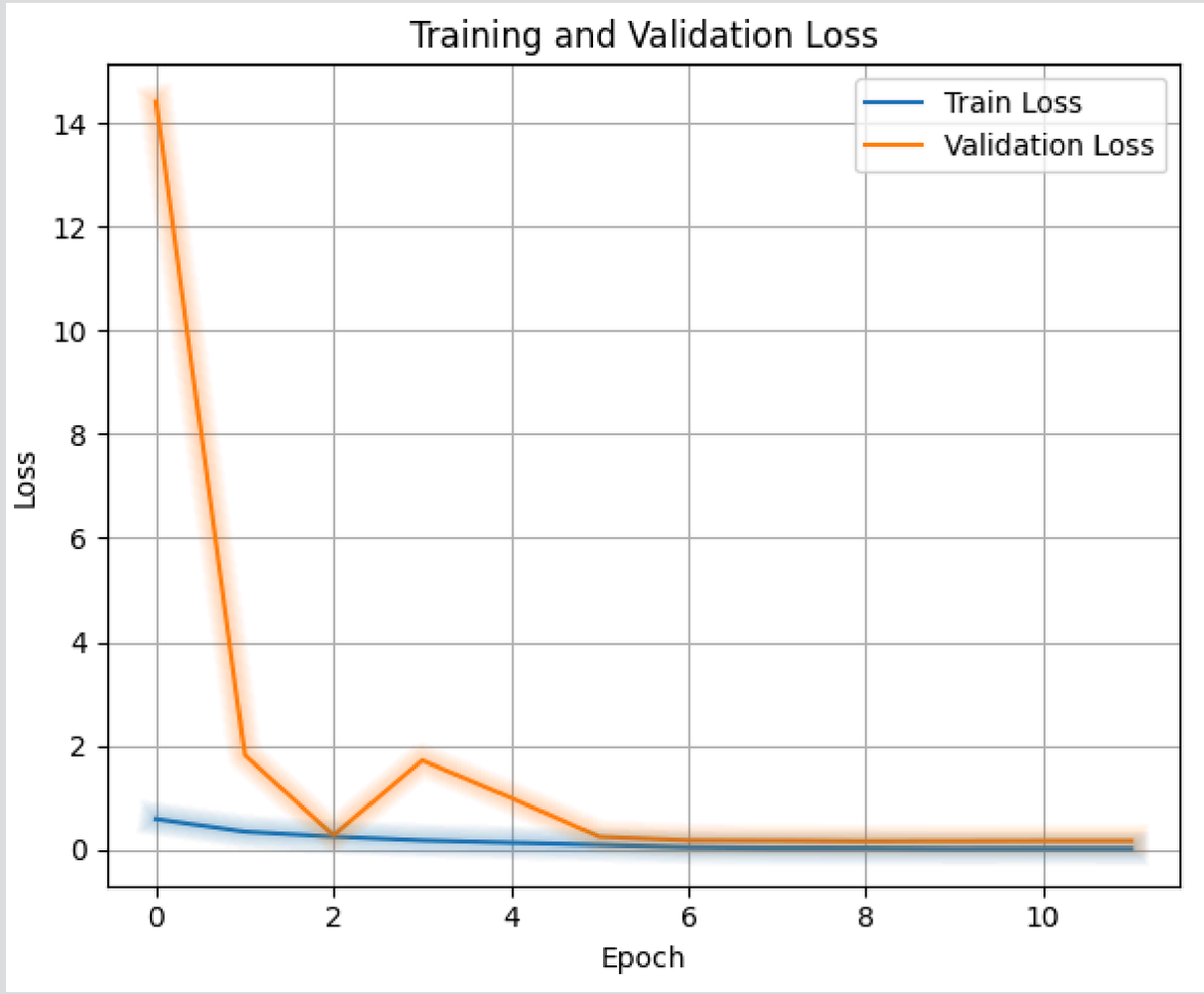
```
(1, 150, 150, 3)
```

```
a=model.predict(img_array)
indices = a.argmax()
indices
```

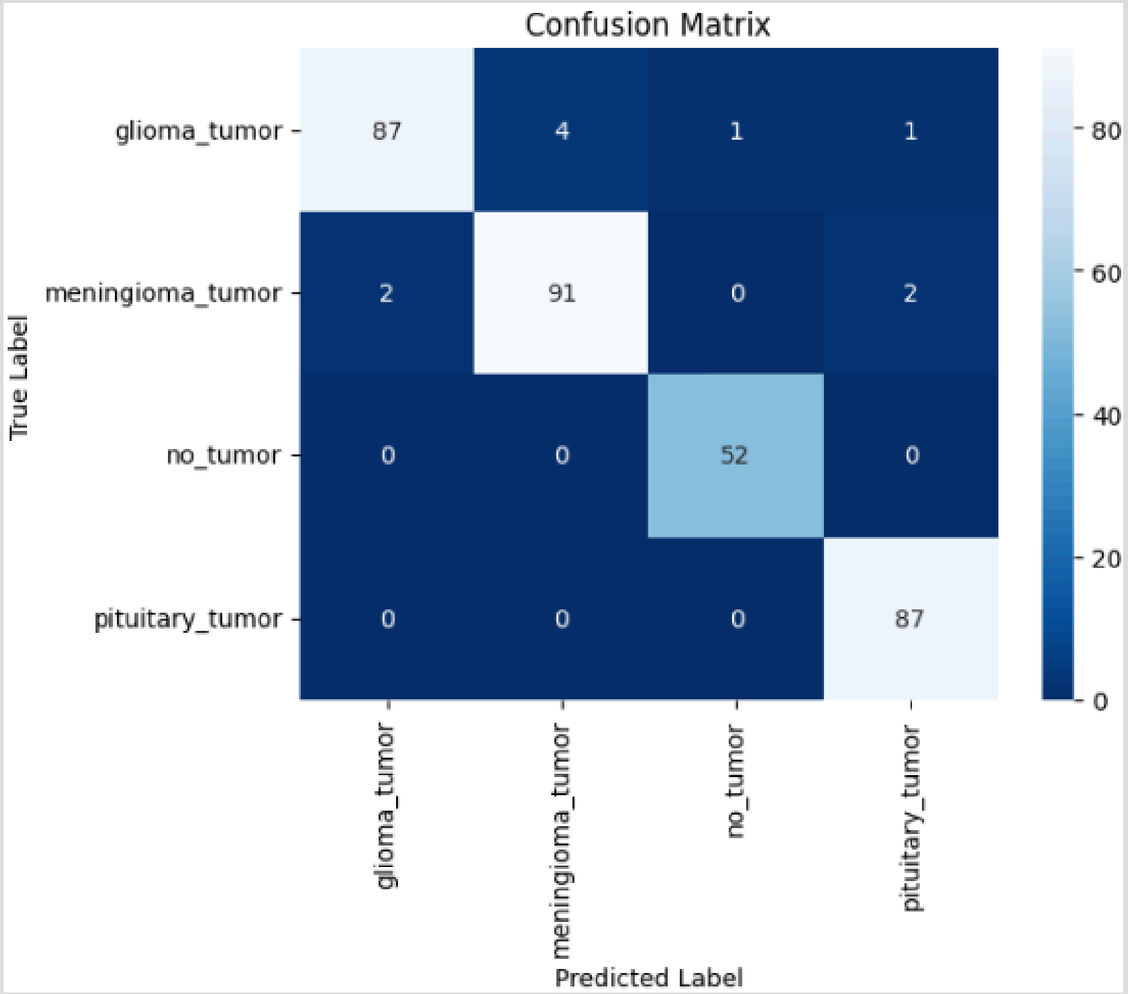
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
0
```

Model Results (Xception)





Model Results (Xception)



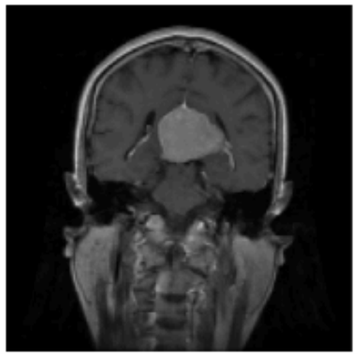
	precision	recall	f1-score	support
0	0.98	0.94	0.96	93
1	0.96	0.96	0.96	95
2	0.98	1.00	0.99	52
3	0.97	1.00	0.98	87
accuracy			0.97	327
macro avg	0.97	0.97	0.97	327
weighted avg	0.97	0.97	0.97	327

Prediction result (Xception)

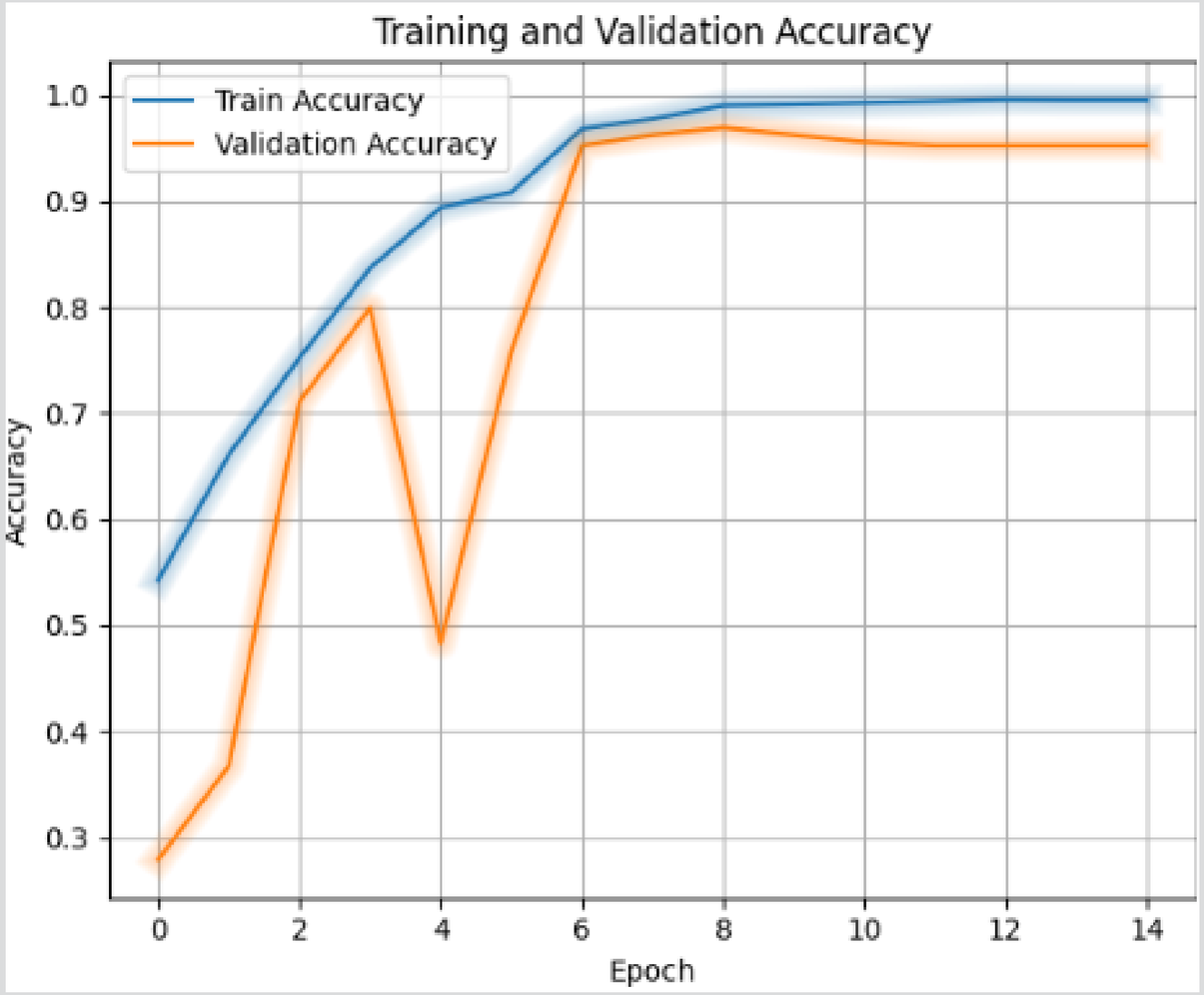
TESTING ON UNSEEN DATA

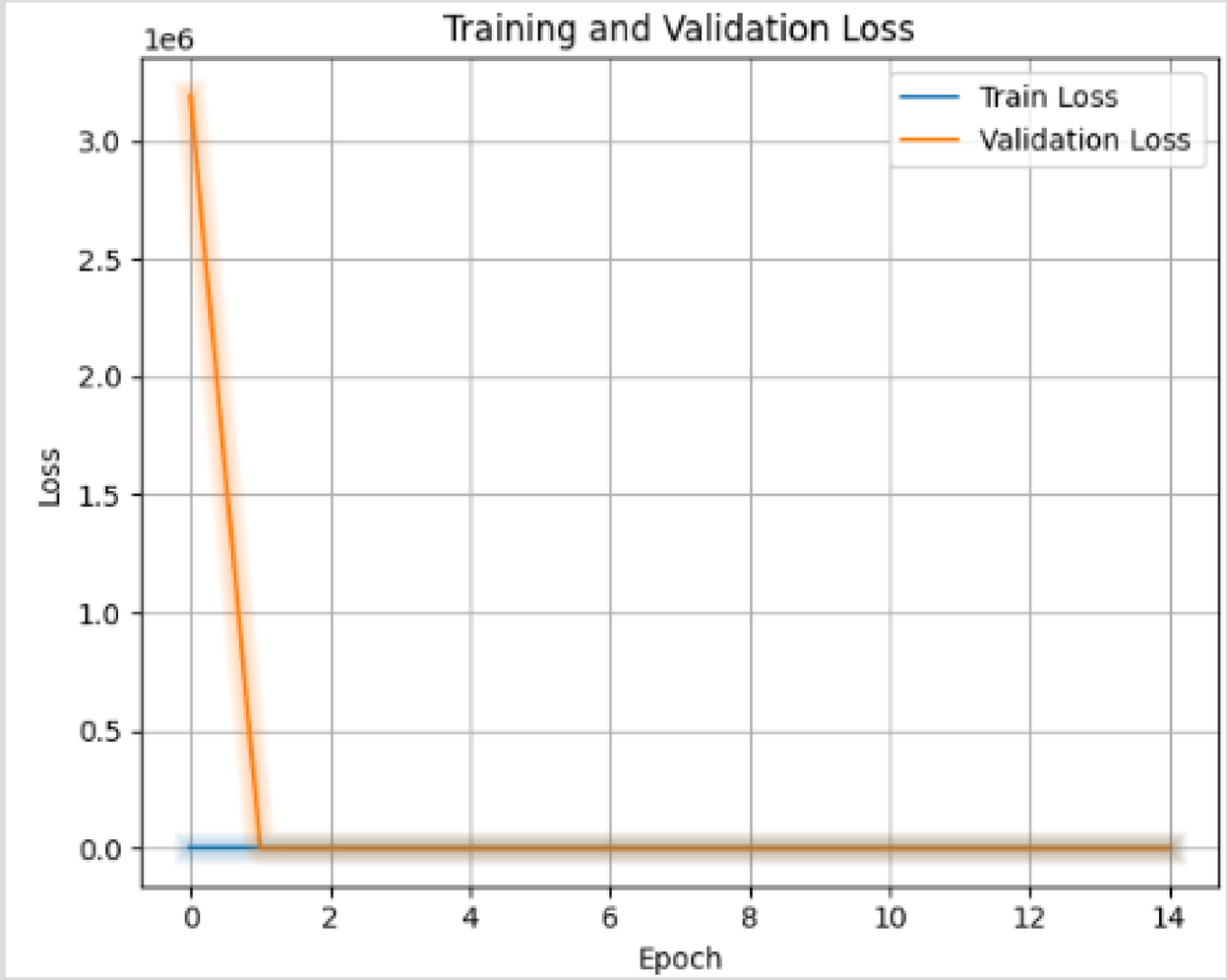
```
random_index = np.random.randint(0, len(X_test))
random_img = X_test[random_index]
predictions = model.predict(random_img.reshape(1, 150, 150, 3))  # Reshape and preprocess the image

# Interpret the model's predictions
predicted_class = np.argmax(predictions)  # Get the index of the class with the highest probability
predicted_label = labels[predicted_class]  # Convert class to label
confidence = predictions[0][predicted_class]

actual_index = y_test[random_index]  # Get the one-hot encoded actual class
actual_class = np.argmax(actual_index)
actual_label = labels[actual_class]

# Display the image and prediction information
print(f"\033[94mPredicted label: {predicted_label}\033[0m \n\033[92mActual label: {actual_label}\033[0m \n\033[93mConfidence: {confidence*100:.2f}%\033[0m\n")
plt.figure(figsize = (3,3))
plt.imshow(random_img)
plt.axis('off')
plt.show()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step
Predicted label: meningioma_tumor
Actual label: meningioma_tumor
Confidence: 99.95%
```

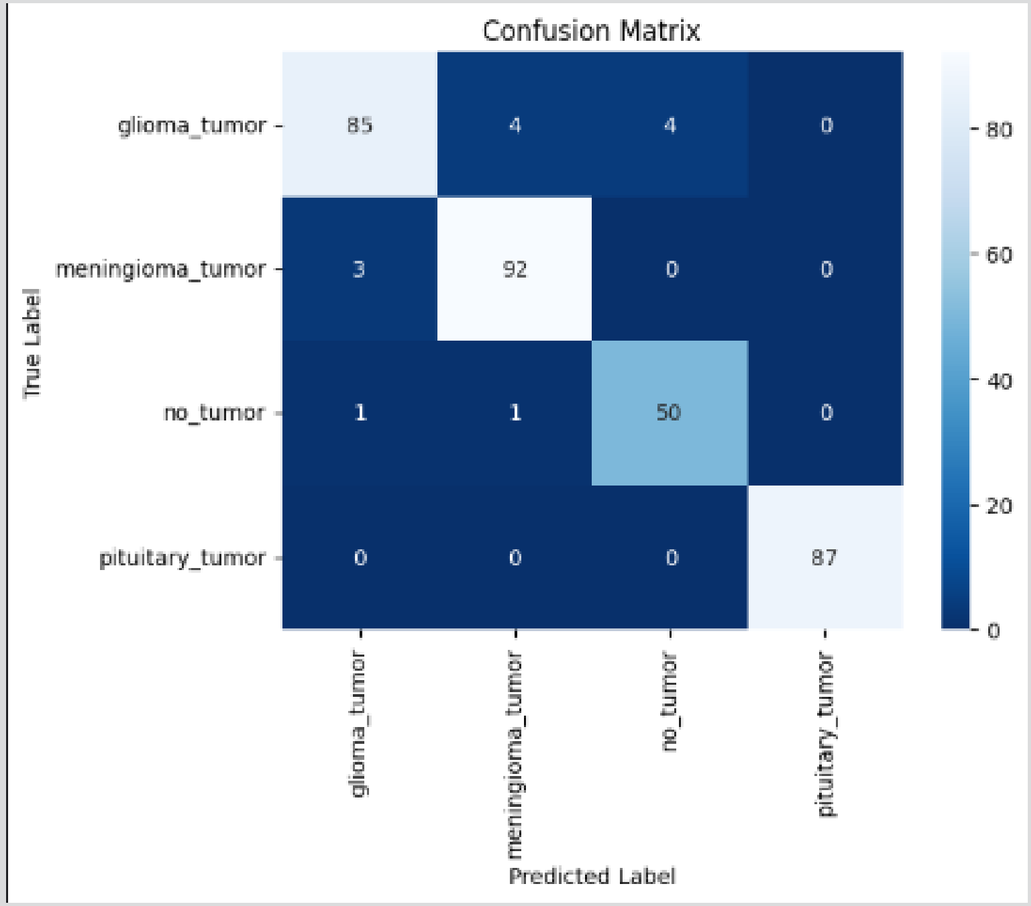


Model Results (ResNet101)





Model Results (ResNet101)



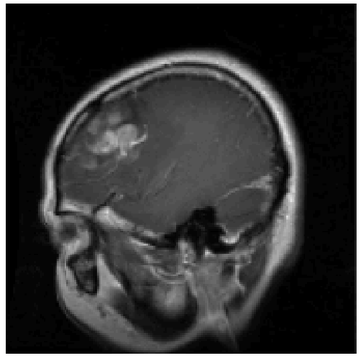
	precision	recall	f1-score	support
0	0.96	0.91	0.93	93
1	0.95	0.97	0.96	95
2	0.93	0.96	0.94	52
3	1.00	1.00	1.00	87
accuracy			0.96	327
macro avg	0.96	0.96	0.96	327
weighted avg	0.96	0.96	0.96	327

Prediction result (ResNet101)

TESTING ON UNSEEN DATA

```
random_index = np.random.randint(0, len(X_test))
random_img = X_test[random_index]
predictions = model.predict(random_img.reshape(1, 150, 150, 3))  # Reshape and preprocess the image

# Interpret the model's predictions
predicted_class = np.argmax(predictions)  # Get the index of the class with the highest probability
predicted_label = labels[predicted_class]  # Convert class to label
confidence = predictions[0][predicted_class]

actual_index = y_test[random_index]  # Get the one-hot encoded actual class
actual_class = np.argmax(actual_index)
actual_label = labels[actual_class]

# Display the image and prediction information
print(f"\033[94mPredicted label: {predicted_label}\033[0m \n\033[92mActual label: {actual_label}\033[0m \n\033[93mConfidence: {confidence*100:.2f}%\033[0m\n")
plt.figure(figsize = (3,3))
plt.imshow(random_img)
plt.axis('off')
plt.show()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
Predicted label: glioma_tumor
Actual label: glioma_tumor
Confidence: 99.69%
```



Results

	Training Accuracy	Training Loss (Error)	Validation Accuracy	validation Loss	Testing Accuracy
CNN	95%	1%	87%	3%	87%
xception	99%	0.009%	96%	1%	97%
ResNet101	99%	0.2%	95%	0.9%	96%

compassion between our best model and previous related works

Study	Accurcy	Classifcation type	Model used
Study 1	91.21%	Multi	SVM-KNN
Study 2	91.43%	Multi	CNN
Study 3	96.14%	Multi	CNN
Our model	97%	Multi	Xception

CONCLUSION

FUTURE WORK

CONCLUSION

- Accuracy: Xception achieved the highest accuracy (97%) followed by ResNet (96%) and CNN (87%).
    - F1-Score: All three models achieved similar F1-scores (around 0.87 for CNN, 0.96 for ResNet and Xception). F1-score considers both precision and recall, providing a balanced view of performance.
        - Class-wise Performance: Xception showed the most consistent performance across all classes, with high precision, recall, and F1-score for each class. ResNet performed slightly lower than Xception but still achieved excellent results. CNN had the lowest overall performance but still managed good results for most classes.

REFERENCES

- **Study 1**: J. Cheng, W. Huang, S. Cao, R. Yang, W. Yang, Z. Yun, Z. Wang, and Q. Feng, ‘‘Enhanced performance of brain tumor classification via tumor region augmentation and partition,’’ PIoS ONE, vol. 10, no. 10, Oct. 2015, Art. no. e0140381a
- **Study 2**: J. S. Paul, A. J. Plassard, B. A. Landman, and D. Fabbri, ‘‘Deep learning for brain tumor classification,’’ Proc. SPIE, Med. Imag., Biomed. Appl. Mol., Struct., Funct. Imag., vol. 10137, Mar. 2017, Art. no. 1013710. doi: 10.1117/12.2254195
- **Study 3**: H. H. Sultan, N. M. Salem and W. Al-Atabany, "Multi-Classification of Brain Tumor Images Using Deep Neural Network," in IEEE Access, vol. 7, pp. 69215-69225, 2019, doi: 10.1109/ACCESS.2019.2919122.

THANK YOU!