Brain Tumour Classification using ResNet50 with LIME and ScoreCAM

Installing dependencies and imports:

```
!pip install kagglehub lime scikit-image opencv-python matplotlib tensorflow --quiet
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 275.7/275.7 kB 19.9 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
  Building wheel for lime (setup.py) ... done
```

```
import os, cv2, numpy as np, matplotlib.pyplot as plt, tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from lime import lime_image
from skimage.segmentation import mark_boundaries

print("✅ TensorFlow:", tf.__version__)
```

```
✅ TensorFlow: 2.19.0
```

Dataset Loading:

```
print("Setting up dataset path...")
dataset_path = None
try:
    import kagglehub
    dataset_path = kagglehub.dataset_download('masoudnickparvar/brain-tumor-mri-dataset')
    print('✅ Dataset downloaded via KaggleHub.')
except Exception as e:
    print('⚠️ KaggleHub not available or failed:', e)

repo_data_path = os.path.abspath(os.path.join(os.getcwd(), '..', 'data', 'brain-tumor-mri-dataset'))
if os.path.isdir(repo_data_path):
    dataset_path = repo_data_path

if not dataset_path:
    raise RuntimeError("Dataset path not found. Install kagglehub or place dataset manually.")

print('📁 Using dataset_path =', dataset_path)

train_dir = os.path.join(dataset_path, 'Training')
test_dir  = os.path.join(dataset_path, 'Testing')
```

```
Setting up dataset path...
Using Colab cache for faster access to the 'brain-tumor-mri-dataset' dataset.
✅ Dataset downloaded via KaggleHub.
📁 Using dataset_path = /kaggle/input/brain-tumor-mri-dataset
```

Data Pipeline:

Parameter	Value
Image Size	224×224
Batch Size	16
Classes	4

Label Mode	Categorical

```
IMG_SIZE = (224, 224)
BATCH_SIZE = 16

train_ds = tf.keras.utils.image_dataset_from_directory(
    train_dir, image_size=IMG_SIZE, batch_size=BATCH_SIZE, label_mode='categorical', shuffle=True)
val_ds = tf.keras.utils.image_dataset_from_directory(
    test_dir, image_size=IMG_SIZE, batch_size=BATCH_SIZE, label_mode='categorical', shuffle=False)

class_names = train_ds.class_names
print("Classes:", class_names)

AUTOTUNE = tf.data.AUTOTUNE
train_ds = train_ds.prefetch(AUTOTUNE)
val_ds = val_ds.prefetch(AUTOTUNE)
```

```
Found 5712 files belonging to 4 classes.
Found 1311 files belonging to 4 classes.
Classes: ['glioma', 'meningioma', 'notumor', 'pituitary']
```

Model Architecture

```
from tensorflow.keras.applications import ResNet50

base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(*IMG_SIZE, 3))
base_model.trainable = False  # freeze backbone initially

inputs = keras.Input(shape=(*IMG_SIZE, 3))
x = tf.keras.applications.resnet50.preprocess_input(inputs)
x = base_model(x, training=False)
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dropout(0.3)(x)
outputs = layers.Dense(len(class_names), activation='softmax')(x)
model = keras.Model(inputs, outputs)

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

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 ━━━━━━━━━━━━━━━━━━━━ 5s 0us/step
Model: "functional"
```

Layer (type)	Output Shape	Param #	Connected to
input_layer_1 (InputLayer)	(None, 224, 224, 3)	0	-
get_item (GetItem)	(None, 224, 224)	0	input_layer_1[0]…
get_item_1 (GetItem)	(None, 224, 224)	0	input_layer_1[0]…
get_item_2	(None, 224, 224)	0	input_layer_1[0]…

How can I install Python libraries? Load data from Google Drive Show an example of training a

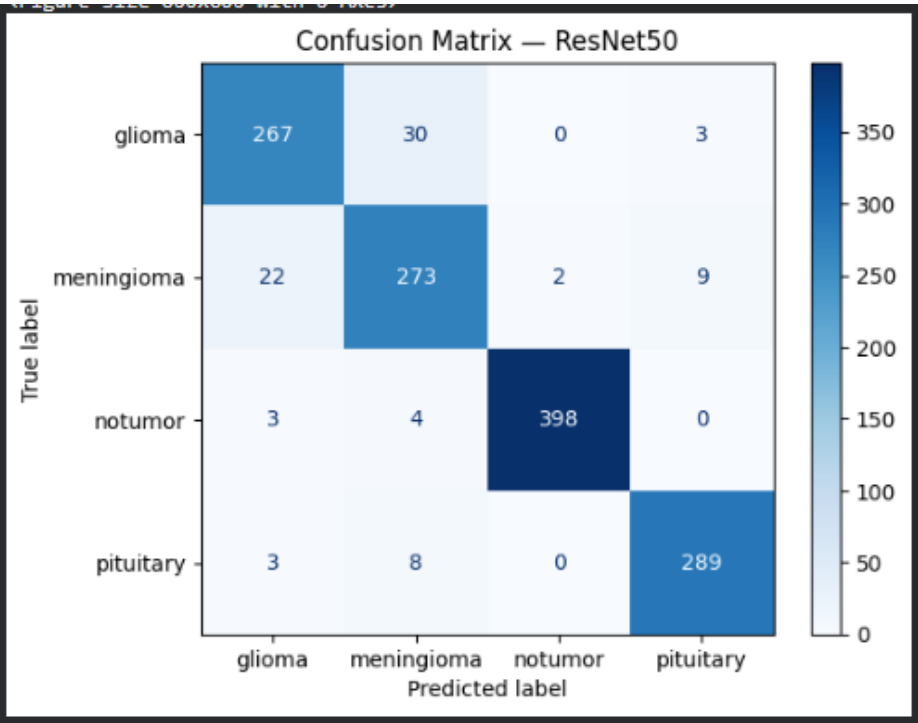
Model Training

```python
EPOCHS = 20  # increased from 8 → 20

callbacks = [
    keras.callbacks.ModelCheckpoint(
        'best_resnet50.h5',
        monitor='val_accuracy',
        save_best_only=True,
        verbose=1
    ),
    keras.callbacks.EarlyStopping(
        monitor='val_loss',
        patience=5,
        restore_best_weights=True,
        verbose=1
    )
]

history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=EPOCHS,
    callbacks=callbacks
)

test_loss, test_acc = model.evaluate(val_ds)
print(f"✅ Test Accuracy: {test_acc:.4f}")
```

How can I install Python libraries? Load data from

```
Epoch 1/20
356/357 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - accuracy: 0.9251 - loss: 0.2028
Epoch 1: val_accuracy improved from -inf to 0.90008, saving model to best_resnet50.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is cons
357/357 ━━━━━━━━━━━━━━━━━━━━ 20s 57ms/step - accuracy: 0.9250 - loss: 0.2029 - val_accuracy: 0.9001 - val_loss: 0.2632
Epoch 2/20
356/357 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - accuracy: 0.9215 - loss: 0.2066
Epoch 2: val_accuracy improved from 0.90008 to 0.90770, saving model to best_resnet50.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is cons
357/357 ━━━━━━━━━━━━━━━━━━━━ 21s 58ms/step - accuracy: 0.9215 - loss: 0.2066 - val_accuracy: 0.9077 - val_loss: 0.2465
Epoch 3/20
356/357 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - accuracy: 0.9292 - loss: 0.1807
Epoch 3: val_accuracy improved from 0.90770 to 0.91762, saving model to best_resnet50.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is cons
357/357 ━━━━━━━━━━━━━━━━━━━━ 21s 58ms/step - accuracy: 0.9292 - loss: 0.1807 - val_accuracy: 0.9176 - val_loss: 0.2074
Epoch 4/20
357/357 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step - accuracy: 0.9273 - loss: 0.1930
Epoch 4: val_accuracy did not improve from 0.91762
357/357 ━━━━━━━━━━━━━━━━━━━━ 20s 57ms/step - accuracy: 0.9273 - loss: 0.1930 - val_accuracy: 0.9146 - val_loss: 0.2168
Epoch 5/20
357/357 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - accuracy: 0.9324 - loss: 0.1730
Epoch 5: val_accuracy did not improve from 0.91762
357/357 ━━━━━━━━━━━━━━━━━━━━ 21s 59ms/step - accuracy: 0.9324 - loss: 0.1730 - val_accuracy: 0.9146 - val_loss: 0.2172
Epoch 6/20
```

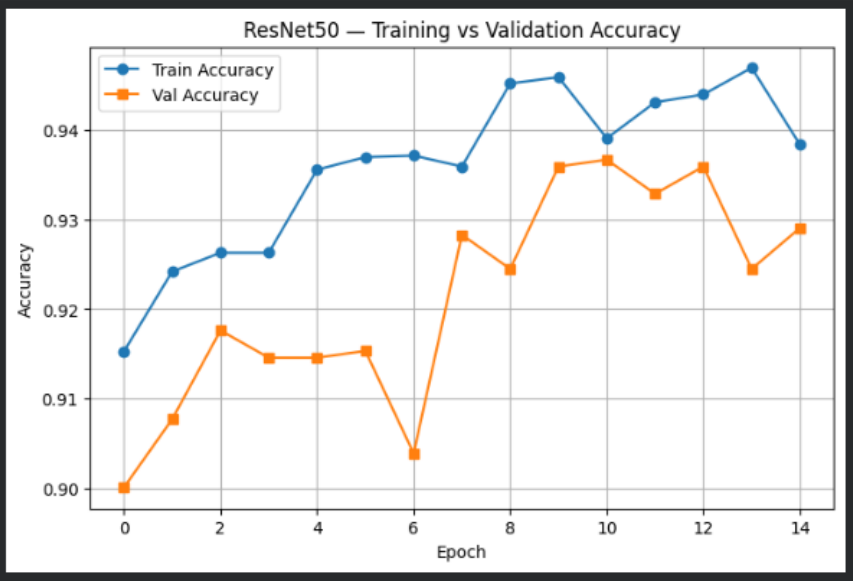
Evaluation Metrics



Classification Report:

	precision	recall	f1-score	support
glioma	0.91	0.89	0.90	300
meningioma	0.87	0.89	0.88	306
notumor	0.99	0.98	0.99	405
pituitary	0.96	0.96	0.96	300
accuracy			0.94	1311
macro avg	0.93	0.93	0.93	1311
weighted avg	0.94	0.94	0.94	1311

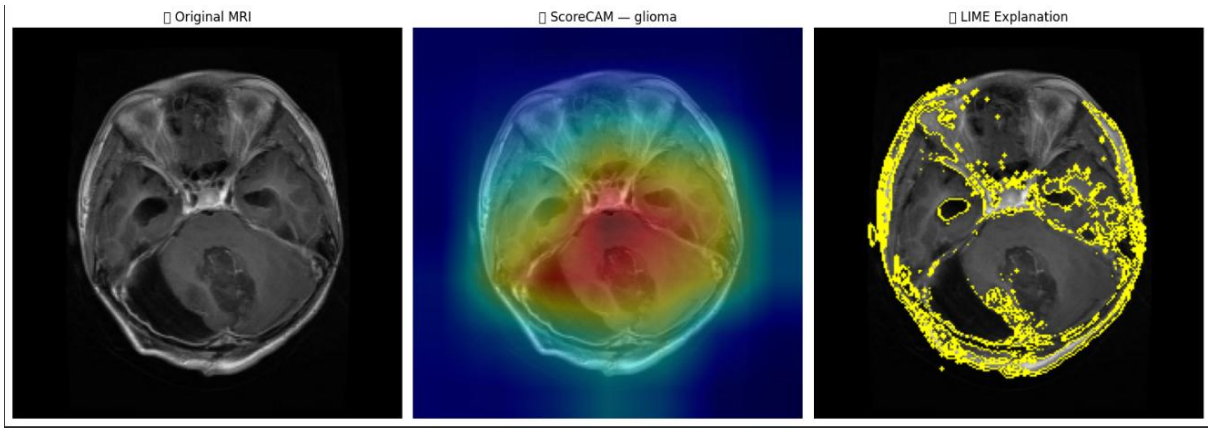
```python
plt.figure(figsize=(8,5))
plt.plot(history.history['accuracy'], label='Train Accuracy', marker='o')
plt.plot(history.history['val_accuracy'], label='Val Accuracy', marker='s')
plt.title('ResNet50 — Training vs Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)
plt.show()
```



LIME and ScoreCAM visualizations

```python
for imgs, labels in val_ds.take(1):
    sample_img = imgs[0].numpy().astype('uint8')
    break

explainer = lime_image.LimeImageExplainer()

def predict_fn(images):
    imgs = np.array(images, dtype=np.float32)
    imgs = tf.keras.applications.resnet50.preprocess_input(imgs)
    return model.predict(imgs)

explanation = explainer.explain_instance(
    sample_img.astype('double'),
    predict_fn,
    top_labels=1,
    hide_color=0,
    num_samples=1000
)

temp, mask = explanation.get_image_and_mask(
    explanation.top_labels[0],
    positive_only=True,
    num_features=8,
    hide_rest=False
)

plt.figure(figsize=(6,6))
plt.imshow(mark_boundaries(temp/255.0, mask))
plt.title("🧠 LIME — Positive Regions")
plt.axis("off")
plt.show()
```

```python
# -------------------------
# 🔥 SCORECAM HEATMAP
# -------------------------
inp = tf.expand_dims(sample_img, 0)
inp = tf.cast(inp, tf.float32)
inp_pp = tf.keras.applications.resnet50.preprocess_input(inp)

base_model = model.get_layer("resnet50")
last_conv_layer = "conv5_block3_3_conv"

activation_model = tf.keras.models.Model(
    inputs=base_model.input,
    outputs=base_model.get_layer(last_conv_layer).output
)

activations = activation_model(inp_pp)[0].numpy()
preds = model.predict(inp_pp, verbose=0)
pred_class = np.argmax(preds[0])
print(f"Predicted class: {class_names[pred_class]} ({preds[0][pred_class]:.3f})")

act_maps = np.maximum(activations, 0)
act_maps = act_maps / (act_maps.max(axis=(0,1)) + 1e-8)

score_map = np.zeros((224,224), dtype=np.float32)
for i in range(act_maps.shape[-1]):
    upsampled = cv2.resize(act_maps[:,:,i], (224,224))
    upsampled_inp = inp * upsampled[np.newaxis, ..., np.newaxis]
    pred = model.predict(tf.keras.applications.resnet50.preprocess_input(upsampled_inp), verb
    score_map += upsampled * pred[0, pred_class]

score_map = np.maximum(score_map, 0)
score_map /= (score_map.max() + 1e-8)
scorecam_overlay = cv2.addWeighted(
    sample_img, 0.6,
    cv2.applyColorMap(np.uint8(255 * score_map), cv2.COLORMAP_JET),
    0.4, 0
)
```



Method	What it measures	Type of reasoning	Output characteristics
ScoreCAM	*Feature importance* inside the CNN (activation-based)	"Where in the image the network is looking"	Smooth heatmap; localized regions

LIME	*Input perturbation sensitivity*	"Which superpixels change the prediction most"	Segmented mask; boundary-based

The **ResNet50** model was selected due to its strong balance between depth, accuracy, and interpretability. Its residual connections mitigate vanishing gradients, enabling deeper feature extraction while maintaining training stability. Moreover, ResNet50's pre-trained ImageNet weights provide excellent feature generalization, crucial for MRI-based medical classification where data is limited. Its modular design also integrates seamlessly with explainability methods like **LIME** and **ScoreCAM**, allowing transparent visual interpretation of predictions.