Development of Bats-diet type Image Classification model using Deep learning

Author - Jaswanth Galle

Supervisor - Professor Laurel Yohe

Objective

- The production and volume of biological data is currently outpacing the ability to process and analyze this data. Imaging data, such as the processed output from magnetic resonance imaging (MRI), micro-computed tomography scanning (μCT), and microscopy, is critical for medical diagnoses,Advances in machinery of scanners and microscopes have enabled fast-paced generation of high-resolution images, but much of the post-processing and analyses still relies on manual analysis and interpretation.

- The major objective is to build a image classification model on Bat-guts MRI scanned image data to predict their diet type.

Dataset

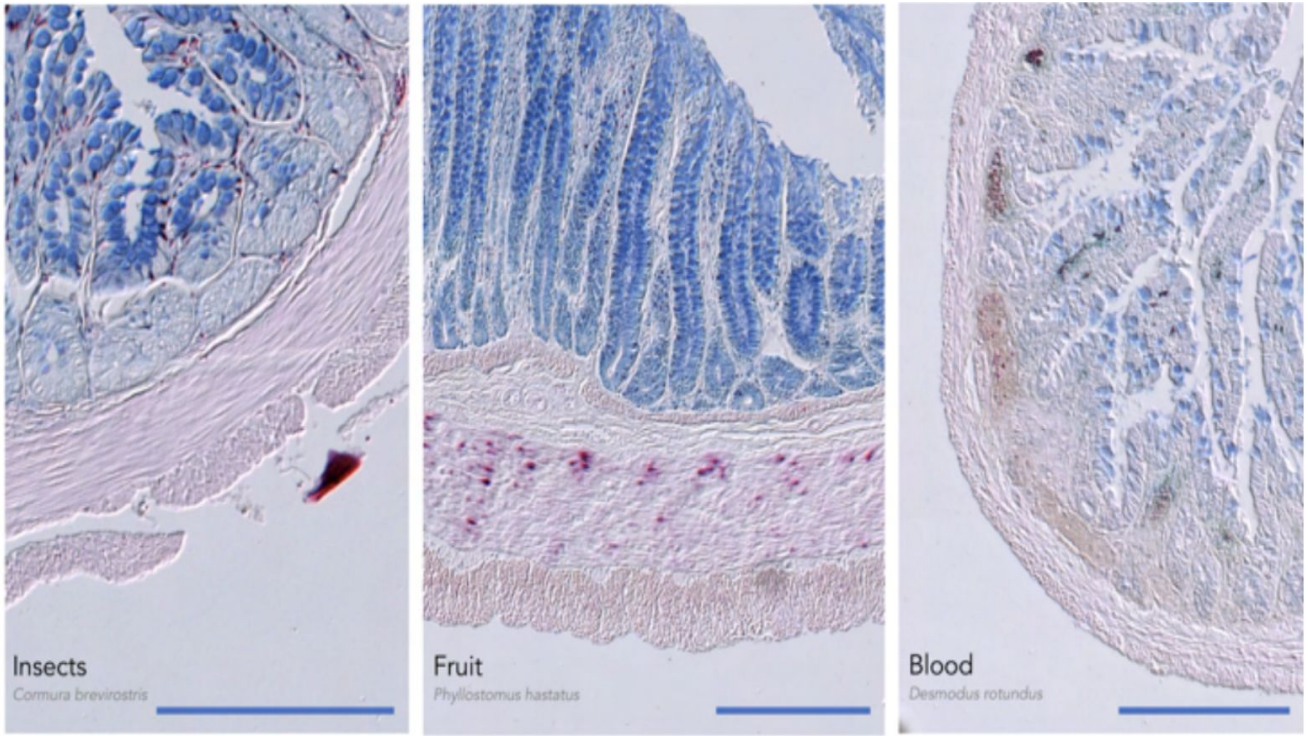
The dataset consists of High resolution histology scanned images of bats intestine which were divided into 3 clas

Insects , Blood , Plants

Images count for each category:

Blood : 20 images
Insects : 260 images
Plants: 179 images

- Dropped Blood category due to data insufficiency



Data Preprocessing

- **Image Resizing and Normalization:**

  1. Resized all the images to a fixed resolution 256 x 256 and 512 x 512
  2.Normalization is done to scale the pixel values of the images to a common range (e.g., [0, 1] or [-1, 1]). This helps in improving the convergence of the training process.

- **Data Augmentation :**

  Keras ImageDataGenerator is used to perform data augmentations to the original
  data and further, it makes the transformation of this data on a
  random basis and gives the output resultant containing only the data that is newly transformed. It does not add the data.

Data Preprocessing

- **Encoding Labels:**

  Converted the categorical class labels (Insects, Plants) into numerical format
  - one-hot encoding For example, Insects: [1, 0], Plants: [0, 1]

- **Train-Validation-Test Split:**

  The dataset is splitted into three subsets: training set, validation set, and test set. The split ratio is 80-10-10.The training set is used to train the model, the validation set is used to tune hyperparameters and monitor performance, and the test set is used to evaluate the final model's performance

PROPOSED METHODS

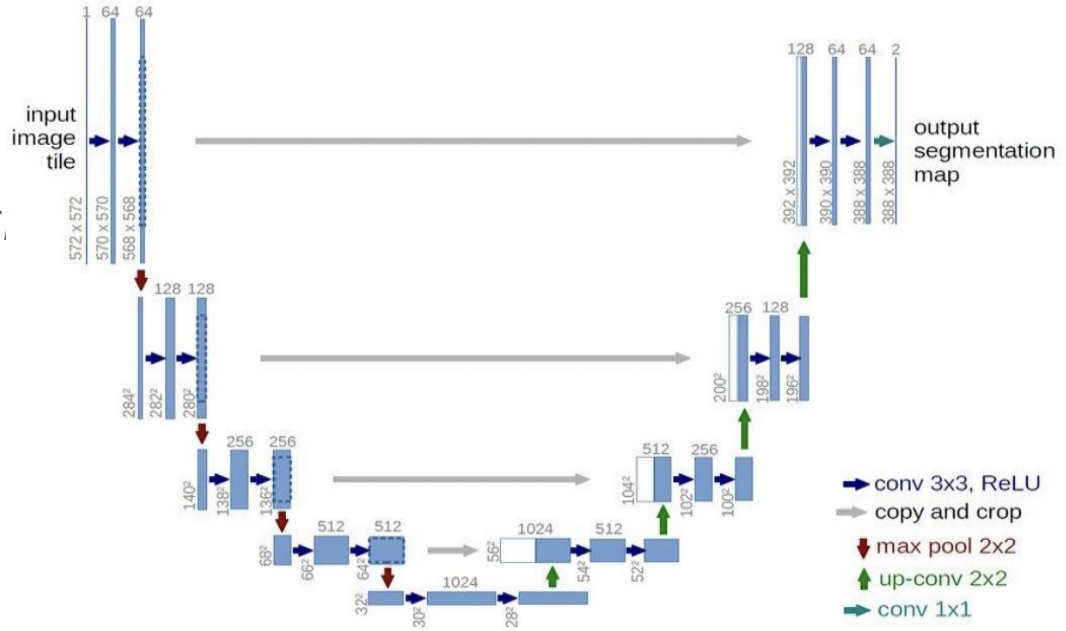
In order to select a model for performing image classification in order to classify images, I trained 10 models present in the Keras Applications library on our dataset for 10 epochs. This is done to understand the potential of the models on the dataset. Then we sort the models based on their train and validation metrics (accuracy and loss). The table below shows the performances of the model sorted in ascending order based on their train and test accuracies. As we can see, the Xception model gives the best train and validation accuracy but does not have the maximum number of parameters.

Index	model_name	num_model_params	validation_accuracy	train_accuracy
0	Xception	20761380	0.516428597	0.467295599
1	InceptionResNetV2	54436636	0.526428597	0.4232704163
2	ResNet101V2	42526570	0.496328597	0.4547169924
3	MobileNet	3448854	0.496428593	0.4358490825
4	ResNet152V2	68331548	0.496428317	0.4169811134
5	EfficientNetV2L	117765848	0.496428532	0.4192452931
6	VGG19	20024384	0.496428532	0.4216981113
7	ResNet50V2	23554800	0.496428597	0.3106918097
8	EfficientNetB7	64547687	0.496428597	0.3044025064
9	InceptionV3	21752784	0.496428532	0.2918238997
10	DenseNet169	12665880	0.496428523	0.2855345964

Table 1: Experiment Results on Keras Applications' models.

U-net Model Architecture

- Using a simple CNN model is not novel
- The model was trained on Vanila U-net architecture
- U-Net was developed in 2015 by Olaf Ronneberger and his team for their work on biomedical images.
- U-Net gets its name from its architecture. The “U” shaped model comprises convolutional layers and two networks. First is the encoder, which is followed by the decoder.



U-net Model Architecture

Encoder Network (Contracting Network):

- Learns feature map of input image to answer "what" is in the image.
- Consists of 4 encoder blocks with 2 convolutional layers (3x3 kernel) and ReLU activation.
- Utilizes max pooling (2x2 kernel) to reduce spatial dimensions

Bottleneck Layer:

- Positioned between the encoder and decoder networks.
- Comprises 2 convolutional layers with ReLU activation.
- Generates final feature map representation.

Skip Connections:

- Key differentiator of U-Net from traditional CNNs.
- Grey arrows in the model architecture diagram.
- Utilizes high-resolution features from encoder blocks to aid segmentation map generation.

U-net Model Architecture (continue)

Decoder Network (Expansive Network):

- Up-samples feature maps to the size of the input image.
- Consists of 4 decoder blocks.
- Each block starts with transpose convolution (up-conv) with 2x2 kernel.
- Concatenates output with corresponding skip connections from the encoder block.
- Employs two convolutional layers (3x3 kernel) with ReLU activation.

Final Output:

- A 1x1 convolution layer with sigmoid activation.
- Produces a segmentation mask containing pixel-wise classification.
  Captures both feature information and localization through the
- contracting and expansive paths.

Transfer Learning with U-net

Vanilla Unet Trained Architecture

```
[ ] from keras_unet.models import vanilla_unet

    base_model  = vanilla_unet(input_shape=(256, 256, 3))
```

```
    ----------------------------------------
    keras-unet init: TF version is >= 2.0.0 - using `tf.keras` instead of `Keras`
    ----------------------------------------
```

```
[ ] from keras.models import Sequential
```

```
    from tensorflow.keras.layers import BatchNormalization
    add_model = Sequential()
    add_model.add(Flatten(input_shape=base_model.output_shape[1:]))
    add_model.add(Dense(256,activation='relu'))
    add_model.add(Dropout(0.3))
    add_model.add(Dense(256,activation='relu'))
    add_model.add(Dropout(0.2))
    add_model.add(Dense(1,activation='sigmoid'))
    model = Model(inputs=base_model.input,outputs = add_model(base_model.output))
    model.summary()
```

Results

```
# Classification Report
print(classification_report(y_test,predictions_onehot,target_names=['Insects','Plants']))
```

```
              precision    recall  f1-score   support

     Insects       0.64      0.70      0.67        10
      Plants       0.67      0.60      0.63        10

   micro avg       0.65      0.65      0.65        20
   macro avg       0.65      0.65      0.65        20
weighted avg       0.65      0.65      0.65        20
 samples avg       0.65      0.65      0.65        20
```

SERVICE API

Bat Guts Image Classification 0.0.0-dev OAS3

/openapi.json

Author

Jaswanth Galle.

Instructor

Dr.Laurel Yohe.

Users

You will be able to :

- **Bat guts can be classified by their intestine images.**
- **Displaying the uploaded image.**

default

GET	/	Index
POST	/displayimage	Display
POST	/feedback	Feedback
POST	/predict	Predict Api

Key Findings

- We are having less number of images(overall) and there is class imbalance as well
- The size of the images are very Large , the major challenge would be to resize this images without data loss
- Most of the images are horizontal stacked image versions of bats intestine , which we have most similar kind of image data in the labels , to overcome this we need more data to have different patterns
- There is high similarity between the images of different labels , these images need to be segmented to find the exact difference in pattern of images

Future work

- More data is need to be added in training for each label
- Blood labelled data will be included in model training data
- Accuracy improvement with segmenting the images
- Image with Audio classification

Thank You