Development of Bat-guts Image Classification Using Deep Learning

Submitted By : Jaswanth Galle

DSBA 6400 | Fall 2023

Mentor: Professor Laurel Yohe, University of North Carolina at Charlotte

Faculty Advisor: Professor Doug Hague

Table of Contents

Introduction

The production and volume of biological data is currently outpacing the ability to process and analyze this data. Extracting quantitatively meaningful statistical relationships relevant to the hypothesis in question remains an open challenge in computational biology and bioinformatics when data sets have tremendous levels of both noise and empirical variance. Imaging data, such as the processed output from magnetic resonance imaging (MRI), micro-computed tomography scanning (μCT), and microscopy, is critical for medical diagnoses, pathological characterization of disease, and foundational understanding of morphological research disciplines such as embryology, anatomical sciences, and even paleontology. 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. In contrast to genomics, high-throughput bioinformatics of phenotypic data falls short despite the onset of "next-generation" imaging. We are at a crossroads in the biological sciences in terms of imaging data: the methods and instrumentation have advanced to the point of generating high-resolution imaging data at large scales but fine-scale automated analyses equivalent to manual quantifications are lacking and cannot scale to the levels necessary to analyze these data sets. At the same time, the field of computational neural networks and machine learning have revolutionized image processing and pattern recognition outside of biology. These methods must become integrated with biological phenotypic data in a more seamless fashion and tools must be developed for biologists without strong computational experience to utilize. Federally funded digital image repositories exist to house large image data sets, but pipelines must be developed to practically extract and make use of these data.

DATASET AND CHALLENGES

A central component of our research is to understand the evolution and adaptation of bats that evolved to specialize on different food items. Of the over 1000 species of bats, most are insectivorous and rely on echolocation to find food. However, several clades have evolved divergent diets (Fig. 1), including plant-consumption, carnivory (small vertebrates), and even sanguivorous (blood-feeding). These specializations can be so extreme (e.g., consuming mostly nectar or exclusively figs or entirely avian blood), that our understanding of the metabolism and harnessing of nutrients of these food resources is poorly understood. Given that these animals are flying mammals, their tight energy budgets call into question how they can sustain themselves on such extreme specializations. An obvious region of the body to focus this question is the intestine, as this is the "gate" through which nutrients are extracted from food. At the same time, the intestine serves as a barrier to harmful pathogens and healthy gut lining minimizes foreign infection; diet may have a strong effect on this. Understanding the cellular basis of diet and gut lining is an essential tool for testing this hypothesis, but as of right now, there are very few quantitative tools to determine these differences. Beyond cell counting and qualitative description, there is a need to develop an interface that extracts information from intestinal images and determines differences in quantitative manner that can be extracted at large scales. Given the discrete nature of bat diet (e.g., insect-feeding, fruit-feeding, blood-feeding), the study system is a useful dataset to explore the performance of neural networks and tailor these algorithms for this problem. Through completion of this objective, I will have developed a tool to quantitatively uncover whether the intestinal morphology is predictive of the diets of bats, as well as uncover which regions of the intestine contains the most informative morphology for predicting diets.

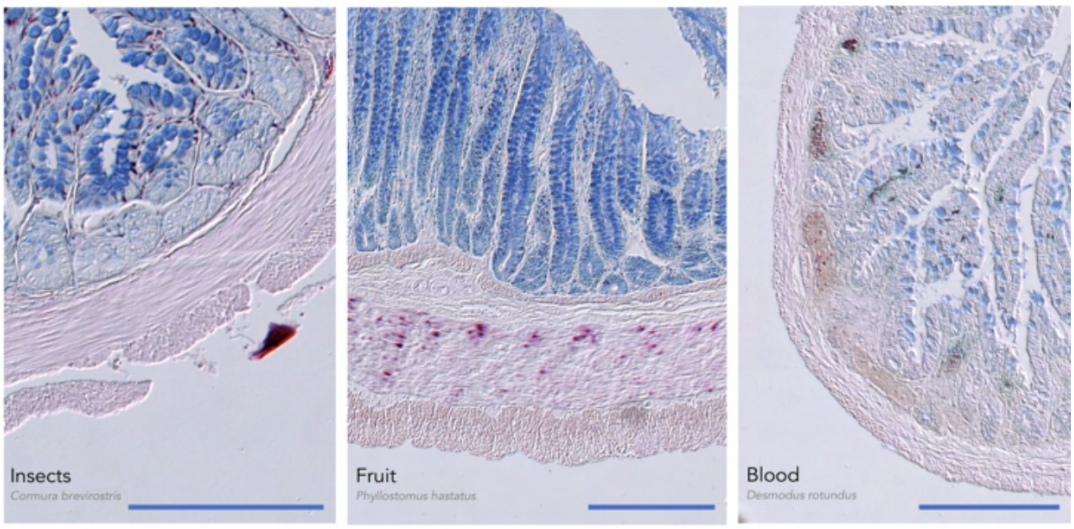


Fig 1 Distal small intestine histology, stained with Alcian blue in three bat species of different diets. Blue cells are goblet cells. Scale is 0.25 mm.

The dataset consists of High resolution MRI scanned images of bats intestine which were divided into 3 classes:Insects , Blood , Fruit Data count for each category is Blood : 20 images , Insects : 198 images ,Plants: 98 images. The major challenges of this data are

- 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

PROPOSED METHODS

Xception Model

In order to select a model for performing image classification in order to detect bats diet type, we train 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	0.666428597	0.667295599
1	InceptionResNetV2	54436636	0.626428597	0.6232704163
2	ResNet101V2	42526570	0.496428597	0.6547169924
3	MobileNet	3448854	0.496428597	0.6358490825
4	ResNet152V2	68331548	0.596428597	0.616981113
5	EfficientNetV2L	117765848	0.596428597	0.5792452931
6	VGG19	20024384	0.596428597	0.616981113
7	ResNet50V2	23554800	0.596428597	0.6106918097
8	EfficientNetB7	64547687	0.496428597	0.6044025064
9	InceptionV3	21752784	0.596428597	0.5918238997
10	DenseNet169	12665880	0.596428597	0.5855345964

Table 1: Experiment Results on Keras Applications' models.

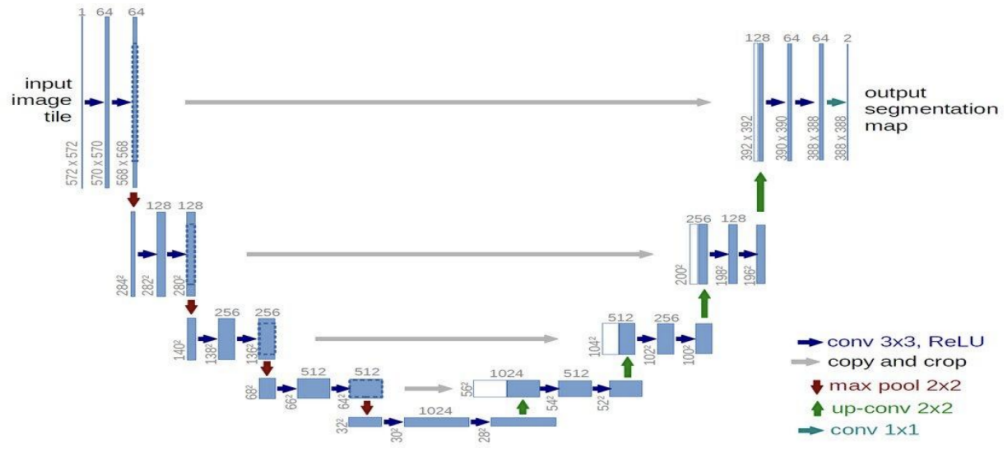
Therefore, we select the Xception model and fine tune it further, by training it on the augmented dataset, using the Adam Optimizer with a learning rate of 0.0001 and then further fine tuning it at an even smaller learning rate of 0.00001 and achieve respectable results shown in the results

section. We hypothesize that the Inception philosophy employed in the Xception model allows it to analyze the image in various scales resulting in the impressive performance.

Transfer learning with U-net model:

U-Net: Convolutional Networks for Biomedical Image Classification:

U-Net was developed in 2015 by Olaf Ronneberger and his team for their work on biomedical images. It won the ISBI challenge by outperforming the sliding window technique by using fewer images and data augmentation to increase the model performance.Sliding window architecture performs localization tasks well on any given training dataset. It is used to create a local patch for each pixel, creating separate class labels for each pixel. However, two main drawbacks of this architecture were that, firstly, a lot of overall redundancy is created due to overlapping patches. Secondly, the training procedure was slow, taking a lot of time and resources.These reasons made the architecture not feasible for various tasks. U-Net overcomes these two drawbacks.We initially talked about how segmentation consists of classification and localization. Let's understand how a U-Net performs these two tasks and why it is so apt for segmentation.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. With the U-Net, we can solve the above two questions of segmentation: "what" and "where."



The encoder network is also called the contracting network. This network learns a feature map of the input image and tries to solve our first question- “what” is in the image? It is similar to any classification task we perform with convolutional neural networks except for the fact that in a U-Net,we do not have any fully connected layers in the end, as the output we require now is not the class label but a mask of the same size as our input image.

This encoder network consists of 4 encoder blocks. Each block contains two convolutional layers with a kernel size of 3*3 and valid padding, followed by a Relu activation function. This is inputted to a max pooling layer with a kernel size of 2*2. With the max pooling layer, we have halved the spatial dimensions learned, thereby reducing the computation cost of training the model.In between the encoder and decoder network, we have the bottleneck layer. This is the bottommost layer, as we can see in the model above. It consists of 2 convolutional layers followed by Relu. The output of the bottleneck is the final feature map representation.

Now, what makes U-Net so good at image segmentation is skip connections and decoder networks. What we have done till now is similar to any CNN. The skip connections and decoder network separates the U-Net from other CNNs.

Transfer learning with U-net

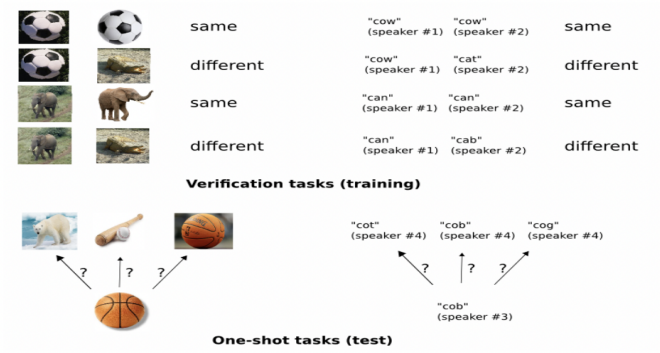
I normalized the input images. This step is followed by defining our evaluation metrics. I use binary cross-entropy, dice loss, and a custom loss function composed of these two. I also use a 80:10:10 ratio for our train:val:test split and display the size of each.

U-net Architecture with CNN layers

```
[ ]  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(0.5))
     add_model.add(Dense(2,activation='softmax'))
     model = Model(inputs=base_model.input,outputs = add_model(base_model.output))
     model.summary()
```

Similarity check Using Siamese Network with VGG layers:

- Even for the less complex networks, deep neural networks require a vast quantity of data to produce the best outcomes.
- Because our dataset is so limited, we are utilizing RGB channels of images to train our deep neural networks.
- Siamese networks are widely renowned for their ability to handle limited data and class imbalances while maintaining high accuracy. Consider that there are 100 class A and 50 class B images. The new dataset will have 5000 image pairs. The label is 1 if both pairings are from the same class, otherwise it is 0.

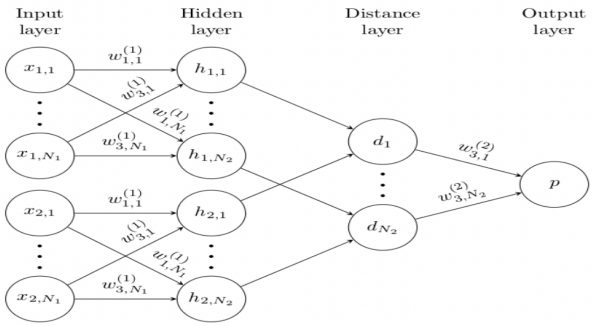


Siamese network image pair labeling

- The network will take a pair of images as the input. so, the classification problem is modified into the similarity problem.
- The model has two networks, both of which are identical. Network will extract the features for both of them and calculate vector distances, and provide the similarity of both inputs as a training network.
- In these we have used contrastive Loss which calculates the similarity between two output vectors.

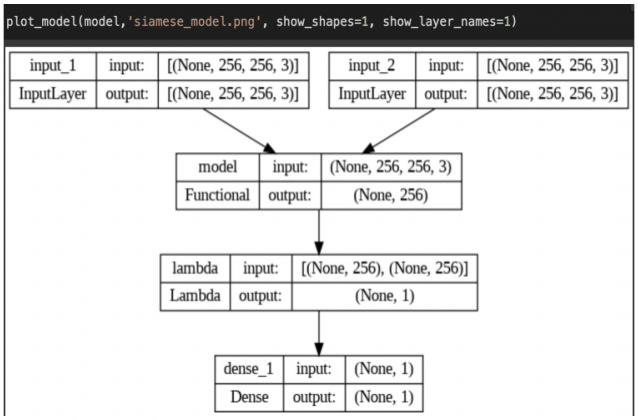
$$loss(d,Y) = \frac{1}{2} * Y * d^2 + (1-Y) * \frac{1}{2} * \max(0, m-d)^2$$

- Where d is the distance between output of two networks, Y is the label , and m is the marginal parameter.



Siamese Network Architecture[1]

- One input is fixed during inference, and the query image is provided to identify the similarity of the query image to the fixed input and create the classification label.
- We used the non-pre-trained VGG16 as the basic model.



Our Siamese network model as base model VGG16

RESULTS AND DISCUSSION

The experiments are carried out using the Google Colab with CPU RAM of 12 GB and the 15GB RAM GPU.

Experimental results on Image classification :

Model	Accuracy	Recall
Xception	**66%**	**65%**
Siamese	**62%**	**62%**
Unet	**67%**	**67%**

Unet Model Results:

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

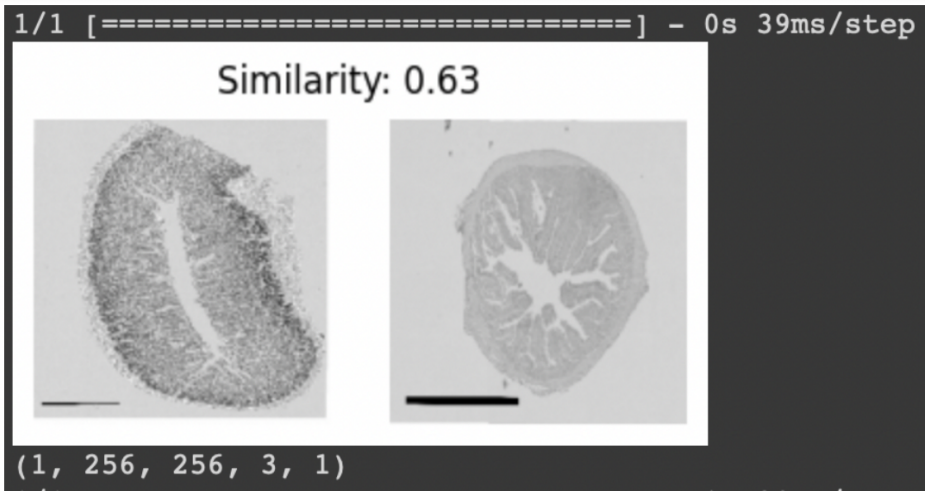
```
              precision    recall  f1-score   support

     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
```

Siamese network with VGG layers Similarity scores b/w known and unknown image:

Left Side is Unknown image,

- Right Side is Known Plant labeled image of bat-gut
- Siamese network with base model VGG16

  gives a similarity score of 63% between known and unknown image



CONCLUSION AND FUTURE WORK

As we can see, the Transfer learned U-net model performs the best when recall is considered. This surprising comparison between the U-net and the Xception model might suggest that an architecture that has a complexity that is between these 2 models might be best suited to this problem. Furthermore, we can say that the class imbalance problem has been dealt with effectively as the model does not just predict all Insects or Plants. Therefore, we hypothesize that it has learnt some differentiating features between the images of Plants and Insects. Although the Siamese network does not produce the best results, given the fact that it was trained only on a subset of the entire training data (due to computational constraints) shows that, if enough experimentation is done with more models and much larger training sizes, the Siamese network with VGG layers could be a promising solution to image classification, especially in cases where there is limited data.

REFERENCES

[1] Koch, Gregory, Richard Zemel, and Ruslan Salakhutdinov. "Siamese neural networks for one-shot image recognition." In ICML deep learning workshop, vol. 2, p. 0. 2015.

[2] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[3] Olaf Ronneberger , Phillip Fischer , Thomas Brox . “U-Net: Convolutional Networks for Biomedical Image Segmentation” conditionally accepted at MICCAI 2015, arXiv:1505.04597

[4] Mayur Mallya, Ghassan Hamarneh . “Deep Multimodal Guidance for Medical Image Classification” .arxiv:2203.05683