Individual Final Report

DATS6203 Machine Learning 2, Prof: Amir Jafari

Phoebe Wu

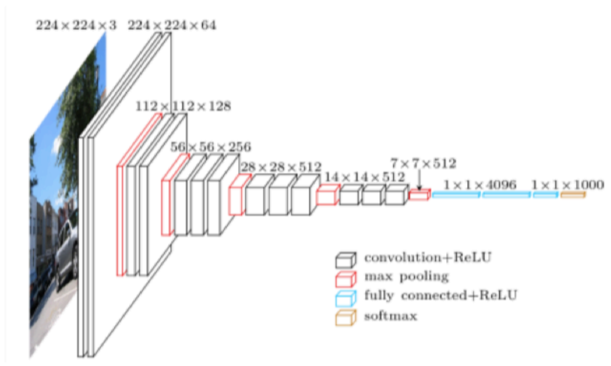
● Introduction

This project goal is to use deep learning network to classify Bees Subspecies and Hive Health. The data is from Kaggle:
https://www.kaggle.com/jenny18/honey-bee-annotated-images/kernels
In this project, we first explore the dataset and operate data preprocessing. Then we use two different neural network frameworks, Keras and PyTorch, to classify Bee Subspecies and Hive Health. We use Gradient Descent Algorithm to optimize the network. In the end we evaluate the result by f1 score, AUC ROC and the test accuracy. We sperate our project into four part including two frameworks and two classification. Each member of our group does one kind of classification by using one of the frameworks.

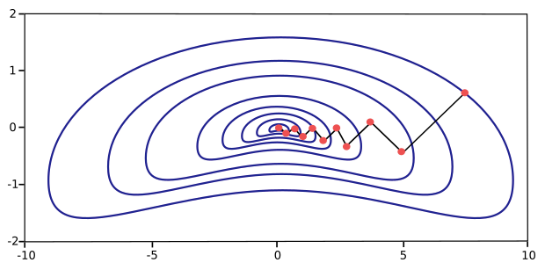
● Description of my work

My work is using Deep learning technique to classify Hive Health by the bee images. The Deep learning network is combining a set of learning models with complex architectures using different transfer functions and training algorithm in deep cascade of layer to get the results. Which includes Multilayer perceptron, Convolution neural network and Recurrent neural network. Since we are classifying the bee images, we use the Convolution Neural Network to build the model and use the Gradient Descent Algorithm to train the model.

- Convolution neural network



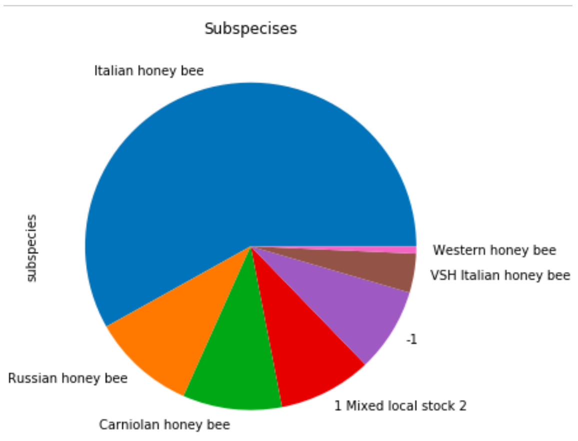
- Gradient descent

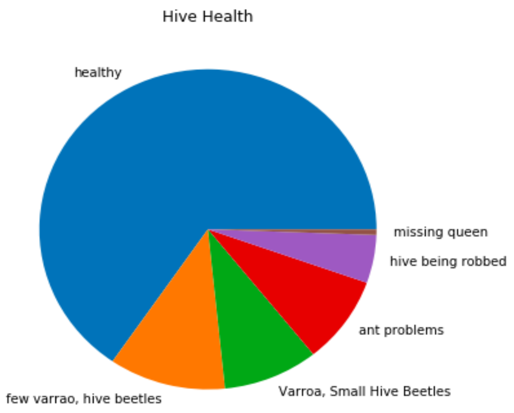


The equation: $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k$

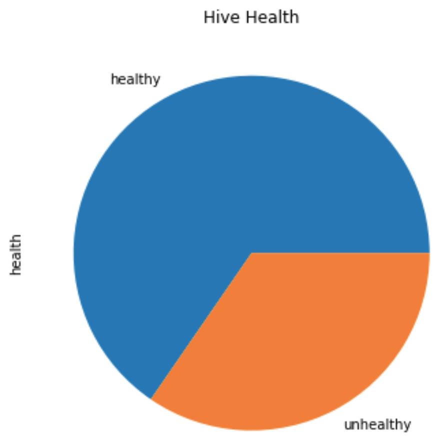
- My portion of the work

- EDA and Pre-processing





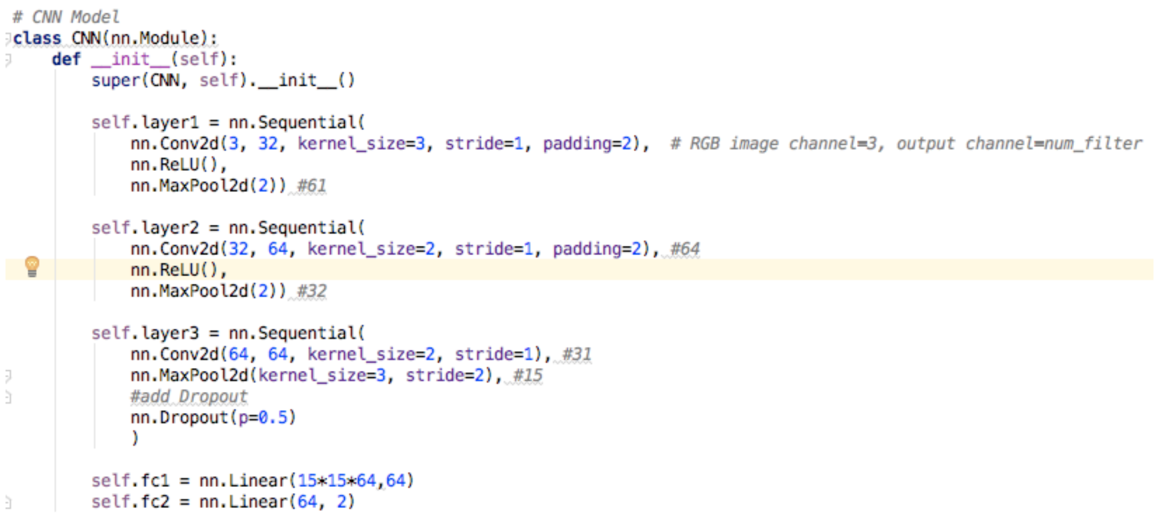
We found out that our data is imbalance, so we decided to convert the rest to unhealthy. That gives us a better place to apply classification.



- Data Loading and data splitting
- CNN Model building

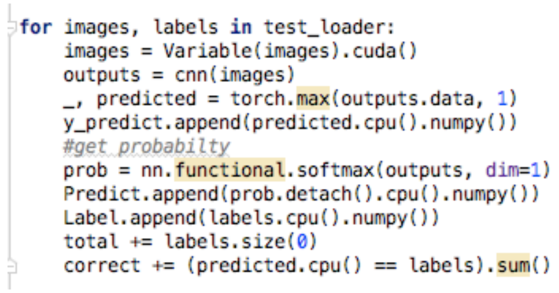
Trying numbers of models, including different transfer functions, adding layers, changing epochs and batch size, and adding dropout nodes. The most important part of the CNN model in Pytorch is calculating the output size each time after we ran the convolution neural network. Here is the formula to calculate output size:

output_size = (in_size - kernel_size + 2*(padding) / stride) + 1



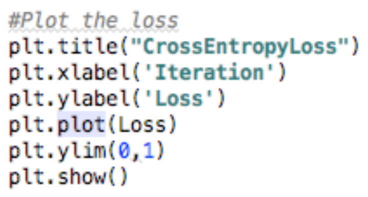
```
# CNN Model
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()

        self.layer1 = nn.Sequential(
            nn.Conv2d(3, 32, kernel_size=3, stride=1, padding=2),  # RGB image channel=3, output channel=num_filter
            nn.ReLU(),
            nn.MaxPool2d(2)) #61

        self.layer2 = nn.Sequential(
            nn.Conv2d(32, 64, kernel_size=2, stride=1, padding=2), #64
            nn.ReLU(),
            nn.MaxPool2d(2)) #32

        self.layer3 = nn.Sequential(
            nn.Conv2d(64, 64, kernel_size=2, stride=1), #31
            nn.MaxPool2d(kernel_size=3, stride=2), #15
            #add Dropout
            nn.Dropout(p=0.5)
            )

        self.fc1 = nn.Linear(15*15*64,64)
        self.fc2 = nn.Linear(64, 2)
```

- Testing the model

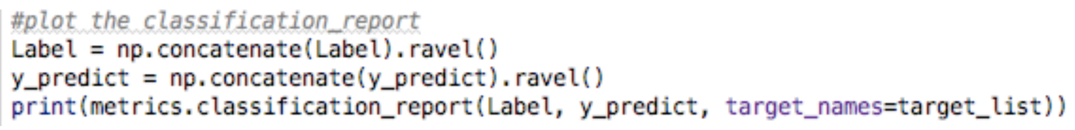


```
for images, labels in test_loader:
    images = Variable(images).cuda()
    outputs = cnn(images)
    _, predicted = torch.max(outputs.data, 1)
    y_predict.append(predicted.cpu().numpy())
    #get probabilty
    prob = nn.functional.softmax(outputs, dim=1)
    Predict.append(prob.detach().cpu().numpy())
    Label.append(labels.cpu().numpy())
    total += labels.size(0)
    correct += (predicted.cpu() == labels).sum()
```

- Plotting the result

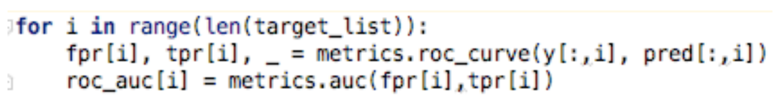


```
#Plot the loss
plt.title("CrossEntropyLoss")
plt.xlabel('Iteration')
plt.ylabel('Loss')
plt.plot(Loss)
plt.ylim(0,1)
plt.show()
```

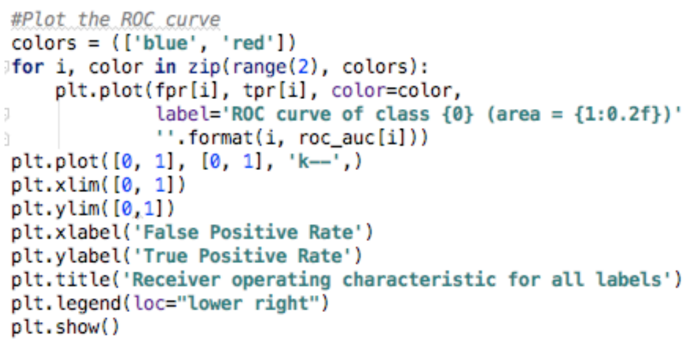


```
#plot the classification_report
Label = np.concatenate(Label).ravel()
y_predict = np.concatenate(y_predict).ravel()
print(metrics.classification_report(Label, y_predict, target_names=target_list))
```

#Calculate the AUC



```
for i in range(len(target_list)):
    fpr[i], tpr[i], _ = metrics.roc_curve(y[:,i], pred[:,i])
    roc_auc[i] = metrics.auc(fpr[i],tpr[i])
```



```
#Plot the ROC curve
colors = (['blue', 'red'])
for i, color in zip(range(2), colors):
    plt.plot(fpr[i], tpr[i], color=color,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(i, roc_auc[i]))
plt.plot([0, 1], [0, 1], 'k--',)
plt.xlim([0, 1])
plt.ylim([0,1])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic for all labels')
plt.legend(loc="lower right")
plt.show()
```

- Result

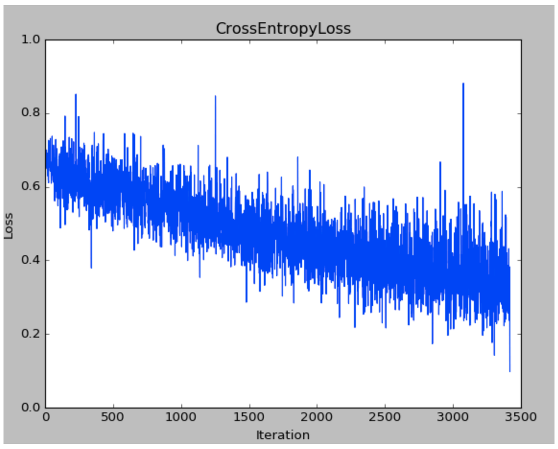
i. Epochs and batch size

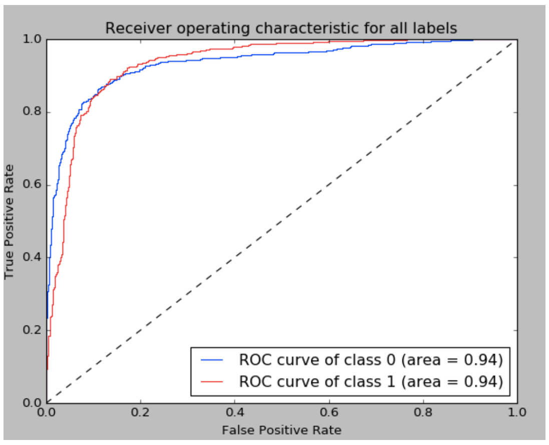
Epochs = 30	Batch size = 32	Batch size = 64
Computational Time	198.37	191.58
Test Accuracy	87%	83%
F1 Score	0.87	0.83
AUC	0.94	0.88

ii. Dropout nodes

Epochs = 30, Batch size = 32	Without Dropout	With Dropout
Computational Time	165.05	198.37
Test Accuracy	86%	87%
F1 Score	0.86	0.87
AUC	0.91	0.94

Train Loss ROC Curve





Classification Report

```
('Computational Time:', 198.369313955307)
Test Accuracy of the model on the 1552 test images: 87 %
             precision    recall  f1-score   support

    healthy       0.89      0.94      0.91      1035
  unhealthy       0.85      0.76      0.80       517

avg / total       0.88      0.88      0.87      1552
```

From the performance, the accuracy and AUC score for the model without dropout nodes are 86% and 91% respectively. On the other hand, the accuracy and AUC score with dropout nodes are 87% and 94%. However, the model with dropout node cost much computational time. Therefore, the dropout node in the mode we used to classify the hive health from the bee images do not affect a lot.

For the batch size, in our model, the small batch size (batch size = 32) shows the better overall performance than the bigger batch size (batch size = 64). Nevertheless, there is not much different in the computational time for the two batch sizes. Thus, the batch size does affect the performance of our model.

● Summary and conclusion

From the model and data, I get the model of 87% accuracy in classifying the health status of the hive using CNN by PyTorch.

In this project, I learn more while using deep learning techniques and convolution neural network to classify the image data. While doing the project, the first challenge I face is loading the data. It takes my groupmate YiJia and I almost one day to figure out. Second, I spent a long time to find out how to calculate the size in the convolution neural network.

After the project, I think the improvement for our project is to increase the data since our data is only 5173 images. Moreover, for the model, I can try more different transfer functions and using different optimizer including changing the parameters to see if I can get the better result.

● Calculation

Percentage of the code that I found or copied from the internet :

61 / 241 * 100 = 25.3%

- Reference

1. Jha, Shekhar. “VGG16-Honey Bee Health Classification.” *RSNA Pneumonia Detection Challenge | Kaggle*, 2018, www.kaggle.com/xanthate/vgg16-honey-bee-health-classification.
2. Gage, Justin. “Convolutional Neural Nets in Pytorch.” *Algorithmia Blog*, Algorithmia, 10 Apr. 2018, blog.algorithmia.com/convolutional-neural-nets-in-pytorch/.
3. “Torch.nn.” *PyTorch*, pytorch.org/docs/stable/nn.html.
4. “Data Loading and Processing Tutorial.” *PyTorch*, pytorch.org/tutorials/beginner/data_loading_tutorial.html.
5. Pukhov, Dmitry. “Honey Bee Health Detection with CNN.” *RSNA Pneumonia Detection Challenge | Kaggle*, www.kaggle.com/dmitrypukhov/honey-bee-health-detection-with-cnn.
6. Leigh. “PyTorch Tutorial: Dataset. Data Prepareation Stage.” *RSNA Pneumonia Detection Challenge | Kaggle*, www.kaggle.com/leighplt/pytorch-tutorial-dataset-data-prepareation-stage.
7. Preda, Gabriel. “Honey Bee Subspecies Classification.” *RSNA Pneumonia Detection Challenge | Kaggle*, 2018, www.kaggle.com/gpreda/honey-bee-subspecies-classification.
8. “Sklearn.metrics.f1_score.” *1.4. Support Vector Machines - Scikit-Learn 0.19.2 Documentation*, scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html.
9. “St-m-Hdstat-Rnn-Deep-Learning.”Wikistat, www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-hdstat-rnn-deep-learning.pdf.
10. Artelnics, Alberto Quesada. “5 Algorithms to Train a Neural Network.” Health Care Applications Neural Designer, Neural Designer, www.neuraldesigner.com/blog/5_algorithms_to_train_a_neural_network .