Sparse Representation Classifier for Artificial Intelligent Microscopy of Blood-Based Diseases Using a Tensor Processing Unit

Purpose

Research Question: How can a novel machine learning model be developed to minimize the number of data examples needed for training a network running on a tensor processing unit that is capable of characterizing white blood cell subtypes for blood-based diseases during real-time microscopy?

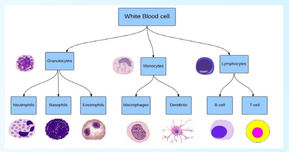
Purpose: The purpose of this experiment is to develop a sparse representation model running on a tensor processing unit that can identify and characterize patient blood samples for blood-based diseases during real-time microscopy using fewer data examples for training compared to a convolutional neural network. Automated methods to detect and classify blood cell subtypes have important medical applications for blood-disease diagnosis, and by comparing machine learning models to determine which network requires the least number of blood samples for training, it is possible to minimize the amount of blood data and thus the cost associated with blood disease diagnosis.

Hypothesis

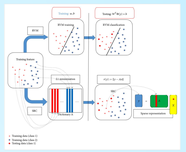
Hypothesis: If a sparse representation classifier is developed running on a tensor processing unit, then the model will be able to identify and characterize white blood cell subtypes for blood-based diseases during real-time microscopy using half the number of data examples for training compared to a convolutional neural network.

Background

Background: Microscopic blood cell analysis is a critical activity in the pathological analysis of blood-based diseases, meaning automated methods to detect and classify blood cell subtypes have important medical applications for disease diagnosis. Deep learning, a sub-field of machine learning, can be applied to the task of medical imaging analysis and recognizing intricate structures within datasets, but one of the largest drawbacks is that deep learning is dependent on large amounts of data. A sparse representation classifier is a type of algorithm that utilizes patterns to recognize characteristics of data in a low-dimensional space and typically requires less data to train the model, which is especially useful for analyzing real-time microscopy. By comparing a sparse representation classifier with a standard convolutional neural network, it is possible to minimize the amount of blood data needed to train a model capable of high accuracy. Furthermore, by running this algorithm on a tensor processing unit, these machine learning tasks can be further optimized using this specialized hardware.



Abbas, Khamael & Banks, Jasmine & Chandran, V. & Tomeo-Reyes, Inmaculada & Nguyen, Kien. (2018). Classification of White Blood Cell Types from Microscope Images: Techniques and Challenges.



Lu Bing, Wei Wang, "Sparse Representation Based Multi-Instance Learning for Breast Ultrasound Image Classification", Computational and Mathematical Methods in Medicine, vol. 2017, Article ID 7894705, 10 pages, 2017. https://doi.org/10.1155/2017/7894705

References

Plenge, E., Klein, S.S., Niessen, W.J., Meijering, E. (2015). Multiple Sparse Representations Classification. PLOS ONE, 10(7), e0131968. https://doi.org/10.1371/journal.pone.0131968

Zhu, Q., Feng, Q., Huang, J., Zhang, D. (2016). Sparse representation classification based on difference subspace. IEEE Congress on Evolutionary Computation (CEC), pp. 4244-4249. doi: 10.1109/CEC.2016.7744329

Sadafi A. et al. (2019). Multiclass Deep Active Learning for Detecting Red Blood Cell Subtypes in Brightfield Microscopy. In: Shen D. et al. (eds) Medical Image Computing and Computer Assisted Intervention – MICCAI 2019. MICCAI 2019. Lecture Notes in Computer Science, vol 11764. Springer, Cham. https://doi.org/10.1007/978-3-030-32239-7_76

Materials and Procedures

Materials: Computer, Internet Connection, Image Dataset of Blood Cells (Subtypes of Eosinophil, Lymphocyte, Monocyte, and Neutrophil), Google Colab, Tensor Processing Unit, Jupyter Notebook IDE, Computer Terminal, Tensorflow Python Library

Procedure:

Data Pre-Processing:

1. Download the 12,500 augmented images of blood cells with accompanying cell subtype labels from Kaggle with a JPEG format
2. Organize the total images into four different folders for each of the four different cell subtypes (Eosinophil, Lymphocyte, Monocyte, and Neutrophil), with approximately 3,125 images belonging to each cell type.
3. Standardize all of the images so they follow the specified size of 256 by 256 pixels.
4. From the folder of images for each of the four different cell types, randomly select 85% for training the sparse representation model and the convolutional neural network, while the other 15% will be used for validation of these two networks.

Building and Training the SRC Model and Convolutional Neural Network:

1. Build the sparse representation model as a three-layer dense network that implements a sequential architecture. Each dense layer of the sparse representation classifier should apply batch normalization and implement LeakyReLU.
2. Build the convolutional neural network model with four layers, three of which are dense layers. The neural network first contains a flattened layer and then three dense layers implementing LeakyReLU.
3. Implement a steepest gradient descent approach based on the Adam optimization algorithm with step length 1 x 10^-3 to optimize the parameters of both the sparse representation model and the convolutional neural network.
4. Train both the sparse representation model and the convolutional neural network using 500 randomly selected training images from each of the four cell subtypes, a total of approximately 2,000 training images.

Running the Validation Set Through the Models:

1. Run the remaining 15% of validation images through the trained sparse representation model and the trained convolutional neural network, recording the accuracy of each model.
2. Repeat steps 8 and 9, increasing the number of randomly selected images used for training by 1,000 each time until all of the training images have been used.
3. Record and compare the accuracy of each model when trained for 500, 1,500, and 2,500 training images per cell subtype to determine which model minimizes the amount of data needed to train a model capable of accurately classifying the four different blood cell subtypes.

Results and Statistical Analysis

Figure 1: Microscopic Image of a Neutrophil White Blood Cell

Figure 2: ROC Curves for 500 Training Images Per Cell Subtype

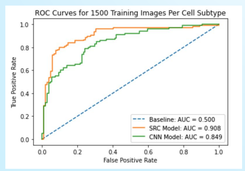


Figure 3: ROC Curves for 1500 Training Images Per Cell Subtype

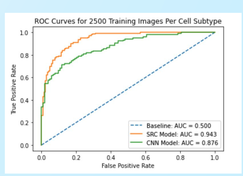


Figure 4: ROC Curves for 2500 Training Images Per Cell Subtype

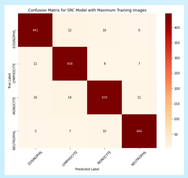


Figure 5: Confusion Matrix of SRC Model for Maximum Training Images

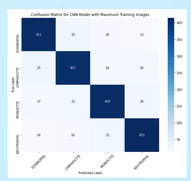
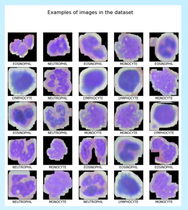


Figure 6: Confusion Matrix of CNN Model for Maximum Training Images

Pictures



White blood cell image dataset with labeled examples of each cell subtype

```
# Detect hardware, return appropriate distribution strategy
try:
    tpu = tf.distribute.cluster_resolver.TPUClusterResolver()
    # TPU detection

    print('Running on TPU ', tpu.master())
except ValueError:
    tpu = None

if tpu:
    tf.config.experimental_connect_to_cluster(tpu)
    tf.tpu.experimental.initialize_tpu_system(tpu)
    strategy = tf.distribute.experimental.TPUStrategy(tpu)
else:
    strategy = tf.distribute.get_strategy()
    # Default distribution strategy in Tensorflow
```

Configuring the tensor processing unit with the Google Colab notebook

```
def load_data():

    datasets = ['/kaggle/input/blood-cells/dataset2-master/dataset2-master/images/TRAIN', '/kaggle/input/blood-cells/dataset2
    images = []
    labels = []

    count = 0
    for dataset in datasets:

        # Iterate through folders in each dataset
        for folder in os.listdir(dataset):

            if folder in ['EOSINOPHIL']: label = 0
            elif folder in ['LYMPHOCYTE']: label = 1
            elif folder in ['MONOCYTE']: label = 2
            elif folder in ['NEUTROPHIL']: label = 3

            # Iterate through each image in folder
            for file in tqdm(os.listdir(os.path.join(dataset, folder))):

                # get pathname of each image
                img_path = os.path.join(os.path.join(dataset, folder), file)

                # Open,
                image = cv2.imread(img_path)
                image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

                # Add padding to the image to better detect cell at the edge
                image = cv2.copyMakeBorder(image,10,10,10,10,cv2.BORDER_CONSTANT,value=[255, 255, 255])
```

Loading the blood cell dataset into the Google Colab notebook and separating the images appropriately

```
def validation(img_test,img_train, train_labels,test_labels, CAE, num_class,args):

    img_test = np.array(img_test)
    img_test = img_test.astype(float)
    img_train = np.array(img_train)
    img_train = img_train.astype(float)

    train_labels = np.array(train_labels[:])
    train_labels = train_labels - train_labels.min() + 1
    train_labels = np.squeeze(train_labels)

    test_labels = np.array(test_labels[:])
    test_labels = test_labels - test_labels.min() + 1
    test_labels = np.squeeze(test_labels)

    CAE.initialization()
    max_step = args.max_step  # 500 = num_class*25# 100*num_class*20
    pretrain_max_step = args.pretrain_step
    display_step = args.display_step #max_step
    lr = 1.0e-3

    class_ = np.zeros(np.max(test_labels))
    prediction = np.zeros(len(test_labels))
    ACC =[]
    Count=[]
```

Running the sparse representation classifier model on the validation set of blood cell images

ALL GRAPHS, DIAGRAMS, AND PICTURES WERE CREATED BY A. PARTHASARATHY

Conclusion

Conclusion: In conclusion, the hypothesis, if a sparse representation classifier is developed running on a tensor processing unit, then the model will be able to identify and characterize white blood cell subtypes for blood-based diseases during real-time microscopy using half the number of data examples for training compared to a convolutional neural network, was supported by the data gathered from the experiment. The results demonstrate that when the sparse representation classifier was run for 500 training images per cell subtype, the accuracy of the model was 85.1%, which is greater than the accuracy of 84.9% for the convolutional neural network with 1500 training images per cell subtype, more than two times the number of training images.

When the sparse representation classifier was trained using 2500 training images per cell subtype, the accuracy of the model was 94.3%, which is 6.7% greater than the accuracy of 87.6% for the convolutional neural network with 2500 training images per cell subtype as shown in Figure 4. Additionally, when the sparse representation classifier was run for 500 training images per cell subtype, the accuracy of the model was 85.1%, which is 9.1% greater than the accuracy of 76.0% for the convolutional neural network with 500 training images per cell subtype as shown in Figure 2. By observing the confusion matrices of both models for the maximum training images of 2500, the number of false classifications by the sparse representation classifier is much smaller compared to the number of false classifications by the convolutional neural network as shown in Figures 5 and 6.

Limitations

Limitations: One of the limitations of this research is that the selected blood image dataset is not entirely representative of the total population of subtypes for individuals afflicted with blood-based diseases. There are many challenging diseases, such as leukemia, that have various subtypes depending upon cell morphology, so the accuracy of the model may be slightly compromised when exposed to a different sample of blood data. Another limitation is posed by lack of understanding of how exactly these algorithms are building and selecting the characteristics, or features, they use to classify data inputs.

Error Analysis

Error Analysis: An example of random errors could include inherent defects within the white blood cell subtype image dataset itself, loss of data during the download and query process, and corruption of portions of data during the transfer of files from directory to directory.

Future Research

Future Research: Future research can be done to expand the capabilities of this algorithm for detecting other white blood cell subtypes, such as basophils and macrophages, as well as implementing this sparse representation model for real-time microscopy. Using augmented reality and cameras oriented through microscopic lenses, this algorithm can be developed to classify cell subtypes in real time.

Applications

Applications: Using this research, doctors will be able to minimize the amount of blood data, and thus the cost, associated with using cell subtypes for blood disease diagnosis. Sparse representation classifiers can achieve high accuracy with less microscopic blood images when compared to other models, all while avoiding human error through computer aided classification. In addition, the use of a tensor processing unit reduces the overall time associated with analysis and processing of blood images, opening up the possibility of utilizing this hardware for reducing the costs of diagnosing other maladies.