Data Augmentation: Malignant Lymphoma Classification Using Convolutional Neural Networks

Jacopo Righetto, Giacomo Sanguin

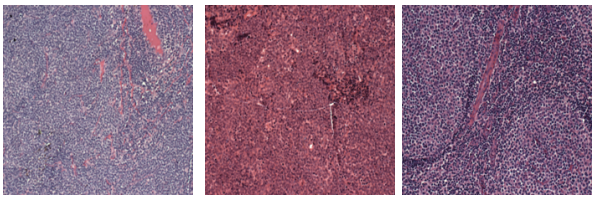
Università degli Studi di Padova

Abstract

This document demonstrates the application of convolutional neural networks (CNNs) for classifying images of malignant lymphoma. We use a dataset containing 375 images categorized into three classes: chronic lymphocytic leukemia (CLL), follicular lymphoma (FL), and mantle cell lymphoma (MCL). Our approach involves training a pre-trained network, AlexNet, with data augmentation techniques to enhance performance. The results indicate the potential of deep learning in assisting medical diagnostics by accurately classifying different types of lymphoma by augmenting the dataset.

1. Introduction

In this paper, we aim to recreate the effect of classifying malignant lymphoma images using a convolutional neural network (CNN). The dataset consists of 375 images, each belonging to one of three classes: chronic lymphocytic leukemia (CLL), follicular lymphoma (FL), and mantle cell lymphoma (MCL). Examples of these images are depicted in the figures below.



Examples of lymphoma images

- CLL: Images exhibit small, round lymphocytes.
- FL: Characterized by irregularly shaped cells with cleaved nuclei.
- MCL: Contains larger cells with more abundant cytoplasm.

We decompose the problem into several steps, including data preparation, augmentation, and CNN training, to achieve accurate classification.

2. Method Overview

Our method uses a pre-trained AlexNet model, with custom modifications and data augmentation techniques, to classify the lymphoma images. The following steps outline the process:

1. **Data Preparation**: Load and preprocess the dataset, resizing images to 227x227 pixels to fit the AlexNet input requirements.
2. **Data Augmentation**: Apply various augmentation techniques such as horizontal flipping, random rotation, cropping, shifting, color jittering, and adding noise to enhance the training set.

Data augmentation algorithm:

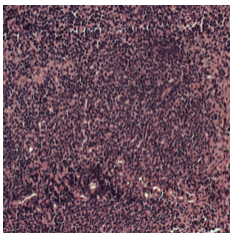
```
augment_data(images, labels)
    augmented_images = []
    augmented_labels = []
    for each image and label in images and labels:
        Apply horizontal flip to image
        Apply random crop to image
        Resize cropped image to original size
        Apply random shift to image
        Apply color jittering to image
        Add noise to image
        Convert noisy image back to image format
        Apply PCA jittering to image
        Apply random rotation and noise to image
        Apply horizontal flip and PCA jittering to image
        Apply crop, shift, and noise to image
        Apply horizontal flip, rotation, and color jittering to image
    return augmented_images, augmented_labels
```

3. **Network Configuration**: Modify the AlexNet architecture by replacing the final fully connected layer with one that has two output nodes (assuming binary classification), followed by a SoftMax layer for classification.
4. **Training**: Train the modified network using stochastic gradient descent with momentum (SGDM), with a mini-batch size of 30, an initial learning rate of 1e-4, for 10 epochs. Use k-fold cross-validation (with 5 folds) to ensure robust evaluation and to mitigate overfitting.

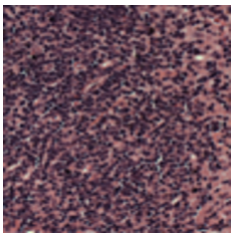
3. Results

We evaluated our method using a 5-fold cross-validation scheme. The network's performance was assessed by the accuracy of classifying the test images in each fold. The following figures show

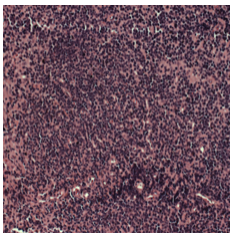
examples of the augmentation techniques applied to the training images:



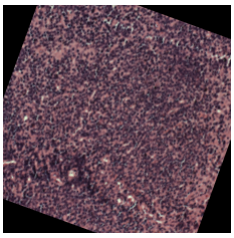
1. Original Image



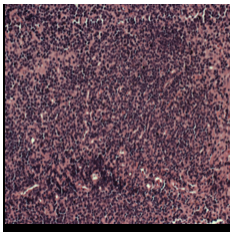
2. Cropped



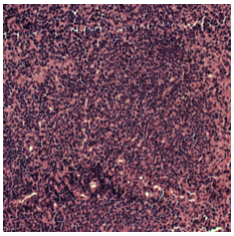
3. Flipped



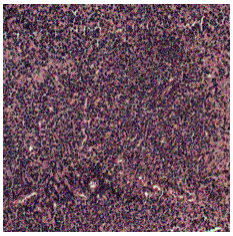
4. Rotated



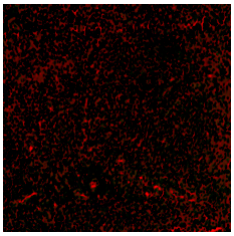
5. Shifted



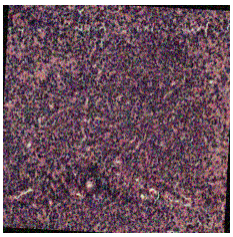
6. Color-Jittered



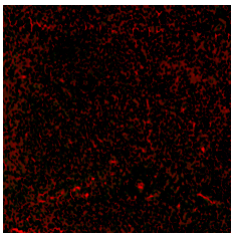
7. Noisy



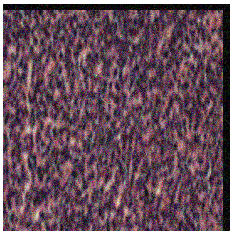
8. PCA-Jittered



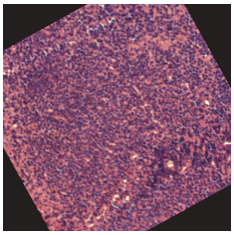
9. Rotated & Noisy



10. Flipped & PCA-Jittered



11. Cropped, Shifted & Noisy

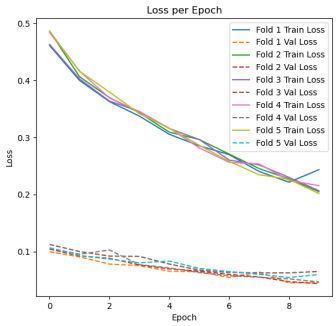


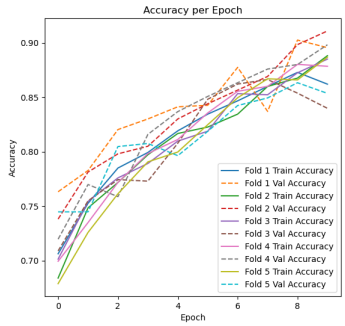
12. Flipped, Rotated & Color-Jittered

The average accuracy and loss achieved across all folds were [0.8104] and [0.3147] respectively, demonstrating the effectiveness of our approach. Table 1 provides a summary of the classification accuracy for each fold.

Table 1: Classification Accuracy & Loss

Fold	1	2	3	4	5
Fold Accuracy	0.8138	0.8091	0.8113	0.8125	0.8050
Fold Loss	0.3128	0.3161	0.3136	0.3163	0.3147





Discussion

The implementation of data augmentation techniques significantly improved the network's ability to generalize, resulting in higher classification accuracy. Future work may involve exploring different network architectures or fine-tuning hyperparameters to further enhance performance.

References

- DING, J., CHEN, B., LIU, H., AND HUANG, M. 2016. *Convolutional Neural Network with Data Augmentation for SAR Target Recognition*. IEEE Geoscience and Remote Sensing Letters, 13(3).
- KRIZHEVSKY, A., SUTSKEVER, I., and HINTON, G. E. 2012. *ImageNet Classification with Deep Convolutional Neural Networks*. Communications of the ACM, 60(6), pp. 84-90.
- NANNI, L., BRAHNAM, S., GHIDONI, S., and MAGUOLO, G. *General Purpose (GenP) Bioimage Ensemble of Handcrafted and Learned Features with Data Augmentation*. IEEE Transactions on Journal Name, Manuscript ID.
- JIA, S., WANG, P., JIA, P., and HU, S. *Research on Data Augmentation for Image Classification Based on Convolution Neural*

Networks. Electronic & Information College, Dalian Jiaotong University, Dalian, China.

Appendices

Appendix A: Augmentation Techniques

- Horizontal Flipping
- Random Rotation
- Random Cropping
- Shifting
- Colour Jittering
- Adding Noise
- PCA plus Jittering

Appendix B: Code Repository

The code used for the experiments and analysis in this paper is available on GitHub. The repository includes the data preprocessing scripts, model training code, and evaluation metrics. Detailed instructions on how to replicate the experiments are also provided in the README file.

The repository can be accessed at the following URL: https://github.com/Jacoporighe/DeepLearning/tree/main/PyTorch