THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

DATS 6203 Section 10: Machine Learning II

Instructor: Dr. Amir Jafari

Final Project | Individual Final Report

Dylan Saez

December 6, 2021

Introduction to Personal Work

Introduction, Saez for Group

In this project, we experiment with several convolutional neural networks (CNNs) in order to find the model that best distinguishes between normal and leukemia blast (cancer) cells. Building a model to accurately classify which type of cell is shown to the computer is extremely important when it comes to diagnosing patients correctly in the medical field.

An overarching interest in becoming more familiar with CNN networks for image classification and applying machine learning skills to a healthcare-related question was the main motivation for choosing this application. The dataset used to answer this question consists of 15,135 images from 118 patients with two classes: Normal cell and Leukemia blast (cancer) cell. 10,661 of these images are used in the training set, 1,867 on our validation set; 2,586 are used on the test set kept by the hosts of the competition. The number of images translates to over 10 GB of data.

In order to show how the best model was selected, information will be provided on the numerous experiments performed consisting of different image classification models, their results, and how they fell short compared with the best model.

PyTorch, a machine learning framework, is used to implement the networks built since there are pretrained networks that we will use as a baseline comparison. F-1 Score and Cohen will be used to measure the performance of the network.

Models experimented on this dataset include VGGNet16, VGGNet19, EfficientNetB7, EfficientnetB3, EfficientnetB4, EfficientnetB5, EfficientnetB0, GoogleNet, Densenet161, DenseNet121, DenseNet169, and ResNet152.

To obtain sufficient background on applying the chosen networks, papers have been included to describe the original intention of the neural network, compare to baseline scores, and describe any features modified for this specific problem.

Description of the Dataset, Saez for Group

As stated above, the dataset consists of 15,135 images from 118 patients with two classes: Normal cell and Leukemia blast (cancer) cell. The number of images translates to over 10GB of data.

According to the section titled, *About this dataset* on the Kaggle competition where this application was introduced, “These cells have been segmented from microscopic images and are representative of images in the real-world because they contain some staining noise and illumination errors, although these errors have largely been fixed in the course of acquisition”. Staining noise indicates a better visualization of the different cells by negative staining, an important tool for the study of biological molecules. The paper where negative staining is used in preparation for visualization modification, Ohi, Li, Cheng, and Walz (2004) indicates, “Negative staining, the embedding of a specimen in a layer of dried heavy metal solution, was introduced early on a as a quick and easy specimen preparation technique that significantly increases the specimen contrast.” On the other hand, illumination errors are removed to “support colour-based image recognition and stable tracking (in and out of shadows),” Finlayson (2018).

This dataset can be trusted to be an accurate representation of Normal and Leukemia blast (cancer) cells since it was reported in the Kaggle competition that an expert oncologist annotated the ground truth labels of the data.

Details of Models Run

Saez Experimentation Overview

Identical to Stefani’s statement, “After Chris’s finalized initial code base model script, I took to experimenting with different pretrained architectures provided by the torchvision models package.”

Before implementing the different VGGNet and EfficientNet models, I gathered some background information on these two networks by reviewing the papers that introduced them.

The CNN model, VGG Network, was proposed by Karen Simonyan and Andrew Zisserman in their paper, "Very Deep Convolutional Networks for Large-Scale Image Recognition". The main idea from this paper is that adding more (convolution) layers, the model performance increases. Specifically, increasing depth (layers) using an architecture training with 224x224 RBG images and 3x3 kernel (convolution filters), the performance of the model increases. Using non-linear activations in between layers allows the function to become more "discriminate" and decreases the number of parameters by a 3x3 non-linear decomposition. This paper also introduced a range of depth 16-19 for layers.

Another CNN model, EfficientNet, was introduced by Mingxing Tan and Quoc V. Le in their paper, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". Model scaling deals with three main components: depth, width, and resolution to improve the performance of the model.

Depth: the number of layers in a network.

Width: the number of neurons in a layer, filters in a convolutional layer

Resolution: (height)(width) of the input image.

By scaling up on the baseline models MobileNets and ResNet, Tan and V. Le demonstrate the effectiveness using a compound coefficient of depth, width, and resolution.

Model 1: VGGNet16

Inspired by an attempt posted to this Kaggle Competition and previous work using this model in a completely different application, I tested VGGNet (16) and VGGNet(19). I did not attempt the other versions since my scores were low and I wanted to move onto different models. For data augmentation, I applied random horizontal and vertical flips with probability of 0.5. The input shape was the same as that of the notebook's (128, 128, 3). The model was in the mid-high 50's. I used 20 epochs, the notebook used 500. Batch size was 512, learning rate was 0.001, and no class imbalance code was written. Accuracy scores ranged from mid-high 50's for VGGNet16, with a low accuracy score of 54.9% to 58.3%.

Model 2: VGGNet19

With 20 epochs, I reached the high 50's: with a low accuracy score of 57.4% % and a high of 59.2%. The same data preparation from (16) applies here. As stated above, I did not attempt the other VGG versions since my scores were low and I wanted to move onto different models. Surprisingly, for the other application I applied this model to, I received scores in the high 70's. I would hypothesize that my score was so low because of the data augmentation - there may be a lot more to add in this section of the code.

Model 3: Efficientnet_B4

Chris tested the largest EfficientNet Model (B7) and Stefani tested the EfficientNet-B0 (baseline network). I tested two different iterations of the network (B4) and (B5). The main idea of EfficientNet is tuning a model scaling with the three components: depth, width, resolution to achieve maximum model performance. For data augmentation, I applied horizontal flip, vertical flip, brightness, and contrast specifications. The input size of the image was (300,300). Batch size was 32, epochs assigned to 7, and learning rate of 0.001. For class imbalance, I used oversampling for the minority class. Accuracy scores ranged from mid-60's for EfficientNet(B4) with a low accuracy score of 62.3% and a high of 63.8%.

Model 4: Efficientnet_B5

Following the same specifications as B4: With 7 epochs, I reached up to low-60's on the test set, with a low accuracy score of 61.1% and a high of 62.8%.

Results, Summary, Conclusions

The worst of my models was vgg_net_16. The best of my models was efficient_net_b4. The next time I try to run these models, I would include more data augmentation, transformers, embedding, class imbalance, and other pre-processing items as the professor suggested in class.

Calculation: Percentage of Code

Using the pre-trained models, I wrote about 10 lines of code out of the models. Data augmentation, vgg specifications, and modifying other lines such as batch, epochs, learning rate, etc.

References

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks (2020)
Very Deep Convolutional Networks for Large-Scale Image Recognition Karen Simonyan, Andrew Zisserman (2014)
Kaggle Competition Link
VGGNet Image Link
EfficientNet Image Link (Slide 7)

Code

https://pytorch.org/vision/stable/models.html