Capstone Project Report

Author: Li Yuan

Table of Contents

Executive Summary

Nowadays, more and more unlabeled medical images are generated either in research teams or hospitals. These institutions need a labeled dataset to analyze the downstream tasks and research. However, manually labeling millions of medical images requires enormous time and human labor to finish with some errors. My goal is to design an automatic annotation black box/model to help the labeling process without human labor. I want to apply and test the ability of state-of-art techniques connecting text with image, zero-shot classification models to predict or annotate unlabeled medical images, MedMNIST dataset, which is a benchmark dataset used to test new algorithms or new methods. The challenge is that the general ImageNet zero-shot models are trained based on general images in life instead of medical areas. Thus, the CLIP model solely performs poorly on granular-level labels like malignant, normal, benign, etc. There are two aspects to address this challenge. The first aspect is to predict higher-level medical images like chest and breast. The second aspect is to combine the zero-shot CLIP model with training some traditional CNN models on the labeled dataset by CLIP. My result is that the combination method significantly increases the prediction accuracy from 80% to 95%, and it can correct some mislabeling images produced by solely the CLIP model. My new model improves the annotation accuracy on average by 6% and the F1 score on average by 8%. This shows our new model works, improves the existing model, and can be used in production. The next step is once the OpenAI releases and publishes their zero-shot CLIP model codes, and I will plan to use their codes to train

a domain adaption zero-shot CLIP model to the medical fields from scratch by using text captions paired with images. Finally, I will test my customized zero-shot clip model on the MedMNIST benchmark dataset.

Introduction/background

Because millions of unlabeled medical images are generated in hospitals, waiting for annotations before being analyzed downstream, manually labeling them is too time-consuming. Also, sometimes, the labels or classes that we want to predict are not in all pre-trained models, or some are not. At this time, we come across zero-shot, one-shot, or few-shot transfer learning. The state-of-the-art CLIP model connects text and images. The clip model efficiently learns visual concepts from natural language supervision. The clip model can be applied to any visual classification task by simply giving the visual labels to be recognized. The clip model is trained on an extensive range of natural language supervisions and images on the internet. The general clip model performs better than ImageNet Resnet101 on ImageNet, ImageNet v2, ImageNet Rendition, ObjectNet, ImageNet Sketch, ImageNet Adversarial.

Data

MedMNIST v2: A large-scale, lightweight 2D and 3D biomedical image classification benchmark. This benchmark dataset was collected and processed by eight researchers from six institutions, including universities and hospitals. A formal paper about this dataset was published in 2021: https://arxiv.org/abs/2110.14795 on the arXiv. The GitHub Repository containing the guides and instructions on using this dataset has been developed and supported. A python package called medmnist, has been developed as well. You can download each 2D and 3D respectively by clicking this website: https://zenodo.org/record/5208230 and also download all datasets automatically by setting the download parameter in the package equal to True. The whole dataset is straightforward to access. The researchers are now still actively supporting and developing this dataset and its GitHub Repo. Although the dataset is not proposed for clinical use, it has many advantages for research purposes, such as being diverse, standardized, lightweight, and educational. The diverse dataset provides users with different organ images and tasks like binary/multi-class classification and regression problems.

Methodology

My goal is to automatically annotate the millions of unlabeled medical images without too much human involvement or manually labeling. I will use the zero-shot clip model and convolutional neural network to tackle this problem. The whole process involves two procedures. I first split the data which we want to annotate into two parts. The first part of the data will be annotated by a zero-shot clip model choosing the highest probability label among all. Then I will use a convolutional neural network to train the first part of the data with their predictions from the zero-shot clip model as their CNN ground truth labels. Eventually, I will evaluate this trained CNN model on the second part of the data with their truly ground labels. Once I have this best-tuned CNN model, I can leverage this CNN to annotate new unlabeled medical images. In this way, I can show that using two methods together can achieve higher accuracy on annotation than only using the zero-shot clip model.

Zero-shot clip model

The clip is the short name for Contrastive Language Image Pretraining. As the name suggests, the clip model is trained to pair the natural language concepts with the image concepts. These are awe-inspiring and inspirational works because before this idea and clip model, people always trained and predicted labels on observed/seen labels in computer vision. However, as demand increases, many unseen/unobserved categories need to be recognized in images. Clip models want to realize the prediction and pair the image concepts with unseen labels.

The overall architecture of clip models is shown in the Figure 1:

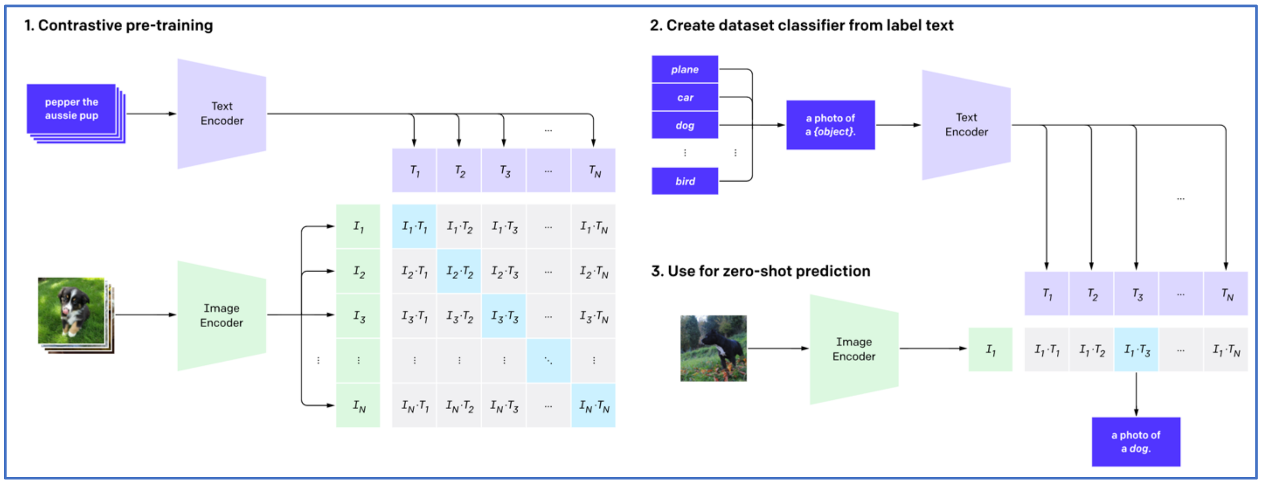


Figure 1

The model contains three phases. The first phase is to train two encoders. The first is the text encoder which extracts and learns the text representations in T1, T2, …, TN; the second is the image encoder which learns visual representations in I1, I2, …, IN. Then the dot product or other similarity score between paired text representations and visual representations is maximized while non-paired text representation and visual representations are minimized. After this optimization, captions that warp the unseen labels are fed into the text encoder to produce each text representation for each class, and the unrecognized image is fed into the image encoder to produce its visual representation. Finally, computing cosine or other similarity scores between this visual representation and each class text representation gives the highest score, and its corresponding class will be chosen as the prediction label for the new image. The above is the overall process for training and prediction. The trained clip model performs well on some standard datasets, like 90.1% accuracy on FOOD101 data, 90.2% accuracy on SUN397, and 89.0% accuracy on YOUTUBE-BB. There are some advantages of this kind of zero-shot model. Firstly, the CLIP is highly efficient. The second one is CLIP is flexible and general because the CLIP is trained on a wide range of visual and text datasets.

Convolutional Neural Network

The following architecture I apply is the convolutional neural network. This is a famous artificial neural network in image analysis like image classification, segmentation, and natural language processing. There has been too much research and applications in CNN in recent years because it is compelling and valuable in image analysis, including video analysis and signal analysis.

As the below figure suggests, the input digit image 2 is the input layer: (28*28*1). This CNN architecture contains two convolutional layers, two max-pooling layers, and two fully connected layers. The first convolutional layer has n1 filter/kernels, each of which is five by five. Each filter slides over the width and height of the input layer and computes the dot product between the selected input layer entries and this filter map. Once this filter slide over all possible width and height, it will produce a 2-dimensional activation map. If the first convolutional has n1 filters, then before the next layer, there are n1 activation maps. Each filter map will learn the different visual patterns of the input layer, such as edges of the orientation, the colors, etc. In the below example, the input layer only has one channel (1-dimensional depth); however, in reality, most images have three channels (3-dimensional depth). In this case, the filter map should also be 3-dimensional depth to make each activation map only 2-dimensional space. In other words, the depth of the filter map is always equal to the depth of the input layer. The most crucial property of CNN is the local connectivity and shared weights because the image dimension is vast, making the full connectivity impractical. The same filter will slide over all widths and heights of the input layer, so weights from this filter map will be used to compute the dot product with the receptive field of the input layer many times. The full connectivity and shared weights can significantly reduce the computational cost and learn the local and different visual patterns. The filter map size usually is small, 3 by 3 and 5 by 5 are the most common ones. The next most common layer is the Max-Pooling; here is 2 by 2 max-pooling map with 2 strides.

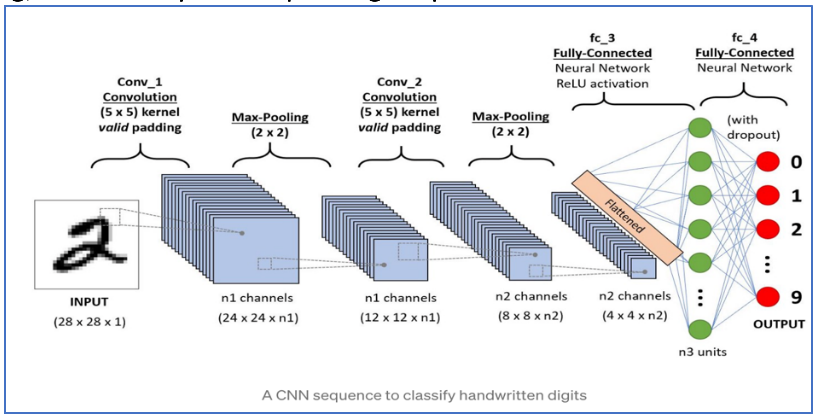


Figure 2

The above picture Figure 2 shows the typical local connectivity and shared weights in one convolutional layer. The input has three channels and one padding, there are two filters, each of which is 3 by 3 by 3, so the output volume has two activation maps.

In practice, people don't need to design a CNN architecture from scratch. People first should look for the current state-of-the-art pre-trained models, like AlexNet, VGGNet, and ResNet, and use these pre-trained models to fine-tune their dataset. The best CNN model on ImageNet usually works better than people's customized model.

My workflow of combinational two above models

Our problem is that we want to automatically annotate the medical images without manually labeling them. So, I will apply one of the best zero-shot clip models to predict the unseen image labels. However, these prediction labels only have 80% accuracy. Thus, the second stage is to train a custom CNN model to improve this prediction accuracy to 95% accuracy.
There are currently eight pre-trained zero-shot models: ['RN50', 'RN101', 'RN50x4', 'RN50x16', 'RN50x64', 'ViT-B/32', 'ViT-B/16', 'ViT-L/14']

After some experiments and comparisons, the Vision Transformers-L/14 (ViT-L/14) performed best among others. ViT-L/14 was chosen to be used. It was trained on ImageNet containing around 14 million images with 21,843 classes, then fine-tuned on another ImageNet at resolution 224 by 224. Next, we can load this huge pre-trained vision transformer model to have a look at its overall sketch. We can observe 427 million model parameters, and the input resolution is 224, consistent with the pre-trained ImageNet resolutions; the context length 77 refers to the maximum text size fed into the model. Training over 427 million model parameters requires a lot of computational resources and time. Luckily, we don't need to train this kind of enormous model. All we need is just apply this model to medical images in the following.

I will use two benchmark datasets as an illustration of my method. The first one is Chest MNIST which contains 78,468 training samples, 11,219 validation samples, and 22,433 test samples. The second one is Breast MNIST which contains 546 training samples, 78 validation samples, and 156 test samples. I chose all 546 breast training samples and randomly chose 546 chest training samples to form a mixture of two benchmark datasets. The following picture Figure 3 is nine random images with their true labels selected from this merged dataset.

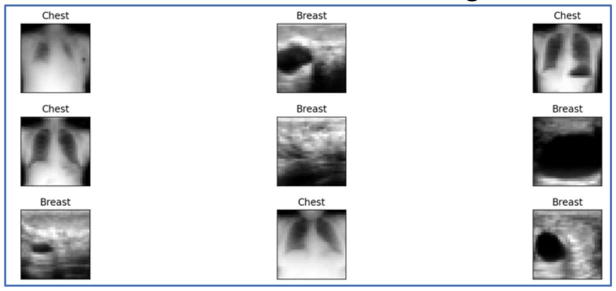


Figure 3

Next, I will create two captions encompassing these two labels, chest and breast. "This is a photo of the chest, and it's a type of organ," and "This is a photo of the breast, and it's a type of organ." For the CLIP model, it's typically to write a short caption or sentence capturing the labels then feed these captions into the text encoder model to produce normalized text features.

Then, I will convert all NumPy-array images to tensor format so that the PyTorch-based framework CLIP vision transformer model can process and move all tensors to GPU Cuda memory to speed up computation efficiency. After this data preprocessing, I use an image encoder to create all image features for all images. Lastly, I will use the cosine similarity to compute the similarity between all previous text features and their associated image features to find the max probability label score, then assign that label to each prediction.

```
SS = data.shape[0]
original_images = [Image.fromarray(data[j]) for j in range(SS)]

images = [preprocess(j) for j in original_images]
image_input = torch.tensor(np.stack(images)).cuda()

with torch.no_grad():
    image_features = model.encode_image(image_input).float()

text_probs = (100.0 * image_features @ text_features.T).softmax(dim=-1)
top_probs, top_labels = text_probs.cpu().topk(1, dim=-1)
```

After I get the prediction labels, I can compute the prediction accuracy by two lines of code. Eventually, by using this vision transformer CLIP model, I got around 80% accuracy on the 1092 training samples in binary classification.

Once I have these prediction labels, I will train a convolutional neural network on these prediction labels to improve their accuracy. The magic and powerful thing are that this trained CNN model from scratch can correct some mislabeling in the CLIP model predictions and then improve the accuracy a lot.

First, I must design this convolutional neural network. Since the breast and chest medical images are relatively small, 28 by 28, and only two classes are waiting to be recognized. The CNN model doesn't have to be deep; namely, a shallow CNN model is capable of classifying binary cases. So, I designed two convolutional layers followed by two of the most common max-pooling layers. The ReLU non-linear activation functions will be applied after each convolutional layer. After these, a flattened layer from images will go through three fully connected linear layers companies by three times dropout operations to overcome overfitting issues. Finally, the two-neuron output layer will be left to produce the probability score. This design architecture is shallow and straightforward but performs very well. We don't need a substantial pre-trained CNN model like AlexNet, and VggNet to achieve the same effect. The following picture Figure 4 displays the details of my customized Convolutional Neural Network.

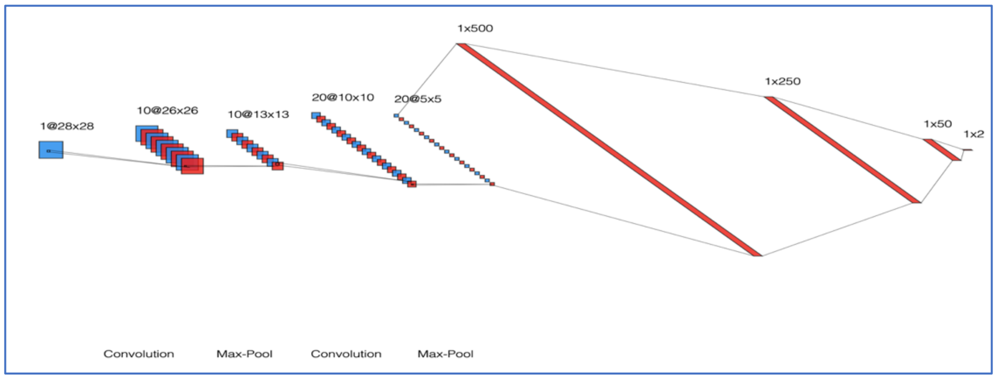


Figure 4

The following code snippet is the PyTorch-based convolutional neural network I used to train, which is the same as described above.

```
class bmodel(nn.Module):
    def __init__(self):
        super(bmodel, self).__init__()
        self.conv1 = nn.Conv2d(1, 10, 3)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.conv2 = nn.Conv2d(10, 20, 4)
        self.fc1 = nn.Linear(20 * 5 * 5, 250)
        self.fc2 = nn.Linear(250, 50)
        self.fc3 = nn.Linear(50, 2)
        self.dropout = nn.Dropout(0.5)
```

```
def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1)
        x = self.dropout(x)
        x = F.relu(self.fc1(x))
        x = self.dropout(x)
        x = F.relu(self.fc2(x))
        x = self.dropout(x)
        x = self.fc3(x)
        return x
```

Usually, we will move all our data and model to the GPU Cuda memory to speed up the computations and make use of our available resources. Cuda is Compute Unified Device Architecture, which can achieve parallel computing. It will improve your learning speed in your parameter update by using GPU rather than CPU.

After reading in the previously merged dataset with 546 chest images, 546 breast images, and their zero-shot CLIP predictions, I am going to randomly split this merged dataset into 80% training and 20% validation set. Eventually, I have 873 training images of size 1 by 28 by 28 and 219 validation images of size 1 by 28 by 28.

Next, I will transfer the images from NumPy to Tensor by the .from_numpy() function and place the images in the data loader to shuffle data and extract the batch size of the image to update the gradients by using DataLoader() function.

As we know, the most important hyperparameter in deep learning is the gradient update method, learning rate, and the type of loss function. In my case, because it's a binary classification problem, the most suitable loss function is the cross-entropy function. I choose the general conventional gradient update method, Stochastic Gradient Descent, and set the learning rate 0.001, momentum equal to 0.9 like the following:

```
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)
```

Once I set up all these environments and image processing, I train the convolutional neural network. In the training function, I use the validation data loader for the maximum number of epochs with no improvement in validation loss for early stopping. I set this argument to 10 and the maximum number of training epochs to 20. Then I start the training process defined by the above parameters. After this training process, I plot the following training and validation loss, and accuracy curves against each epoch shown in the Figure 5:

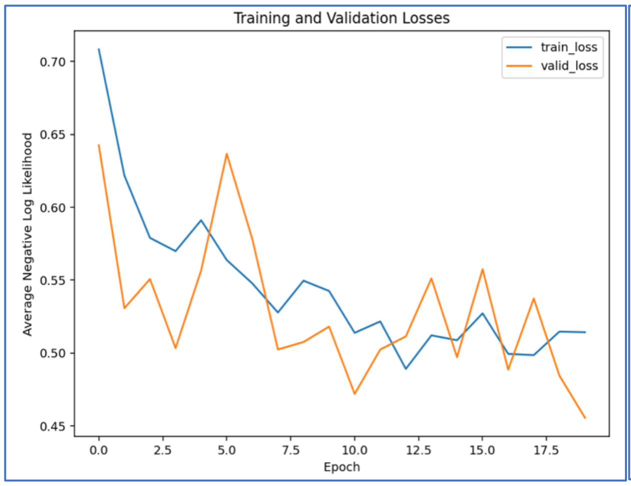


Figure 5

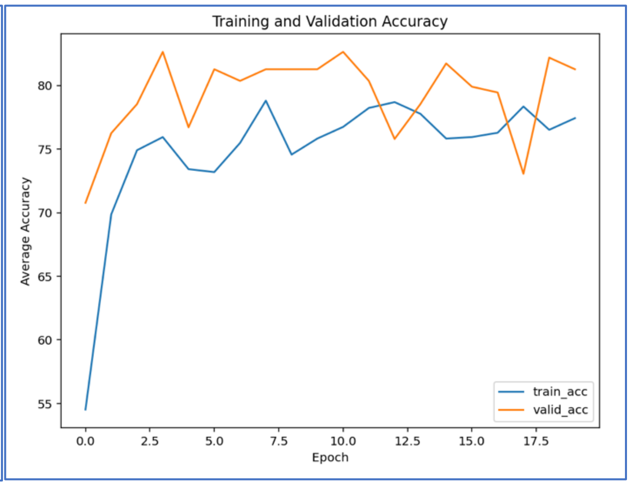
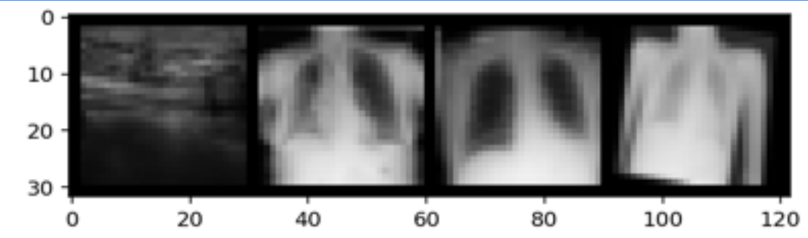


Figure 5

As we can observe from these two plots, training and validation loss continue decreasing from the first epoch until the last one on average. However, the curves oscillate a lot. The training accuracy and validation accuracy continue to increase after 5 epochs and keep at a flat level. The validation accuracy is greater than the training accuracy for most epochs except 12.5 and 17.5. My two convolutional layer CNN model performs well on this task. Next, I will evaluate how this combination of the zero-shot CLIP model and CNN model improve the accuracy of the held-out test set. First, let's get the prediction on the validation test before computing generalized out-of-sample accuracy.



```
GroundTruth:  breast chest breast breast
Prediction:  breast chest chest chest
```

Figure 6

As we can observe from the above picture, Figure 6, the ground truth labels are the zero-shot CLIP model predictions. Obviously, the third and fourth images should be the chest. However, the zero-shot classifies them into the breast. However, the powerful and magic thing is this CNN-trained model can correct the mislabeling in the zero-shot predictions. As we see, the third and the fourth images are classified as chest correctly.

Finally, I will evaluate this CNN model's performance on the held-out test set. I choose all 234 breast images and random 234 chest images from the larger pool. In total, 468 images are there. After creating the true labels for this merged test set, I will compare true labels against the CNN model prediction. I get 95.72% accuracy on the 468 test images. This shows that my idea and new method work perfectly on this binary classification medical image problem. This CNN model can be used to automatically annotate the newly generated chest and breast images without any human manually labeling.

Results

The general zero-shot CLIP model doesn't perform or classify well on a lot of medical images. The research team which published this model conducted a test on one kind of medical image dataset, PATCHCAMELYON, which is a benchmark new and challenging binary image classification dataset with 327.680 colored histopathologic scans of lymph node sections. It only shows 22.8% accuracy on this binary classification problem. However, this doesn't mean that this CLIP model will perform poorly or fail on all kinds of medical image tasks. I tried this model to classify chest and breasts binary classification task. The reason why I chose these two datasets is chest images are significantly different from breast images. My assumption that the CLIP model fails on the histopathologic images is that the task is to detect if there are many tumor cells in the lymph. This detection or classification requires a lot of segmentation or much granular classification because of cell-level recognition. As we know, the zero-shot CLIP model is not trained on a specific medical image dataset but instead on general ImageNet or other datasets, so it is not capable of classifying very granular medical images. However, my example is different from the above story. As we can observe from the above images, chest images and breast images are quite different in a lot of parts, including orientation edges and intensity areas. This makes the zero-shot CLIP model feasible and possible to classify.

The first stage is to apply the zero-shot CLIP model to the mixture dataset of chest and breast to get all predictions with **80.49%** accuracy. However, this is not a satisfactory result in terms of binary classification tasks. So, the second stage is to train a convolutional neural network on this labeled mixture dataset. This CNN model assumes predictions from zero-shot CLIP as "ground truth" to train and validation. The magic and powerful thing are that the trained CNN model can correct the mislabeling and improve the held-out test set accuracy to **95.72%** as compared to the **82.05%** accuracy solely predicted by the zero-shot model. This outcome is very impressive and amazing. After these two stages, the trained CNN model can be used to annotate the chest or breast images for production automatically. Then the labeled medical images can be analyzed in the downstream tasks for more research results and uses.
To show generalized improvement, I also tested my new model 30 times on different random training and test set and observed the results. The upper boxplot shows the difference between the accuracy of the new model and the accuracy of CLIP only. The lower boxplot is the difference between the F1 score of the new model and the F1 score of the CLIP only. As we can observe below in the Figure 7, the new model improves the annotation accuracy on average by 6% and the F1 score on average by 8%. This shows our new model works, improves the existing model, and can be used in production.

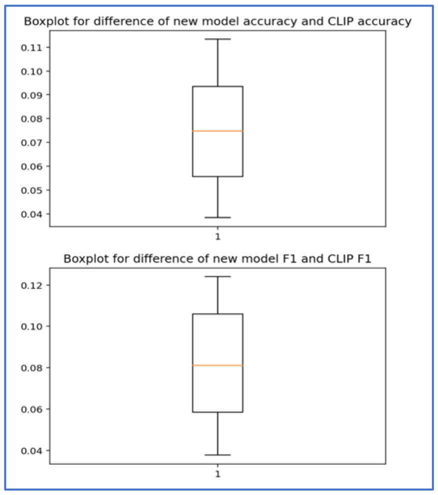


Figure 7

Conclusion/Next Steps

Employ MonAi to zero-shot CLIP model. There are some pre-trained medical image deep learning models, like in the monai, open medical network for Artificial Intelligence, which provides domain-optimized foundational capabilities for developing healthcare imaging training workflows in the native PyTorch language. In the future, I will plan to employ Monai pre-trained medical image models in the current zero-shot CLIP architecture, then apply them to the MedMNIST image dataset to repeat my experiments to see if these domain adaptation models significantly improve the accuracy because current zero-shot learning is trained on ImgaeNet which is far off medical fields. In addition to the zero-shot model, I can also apply and test one-shot and N-shot models to this problem.

Train the CLIP model on my dataset. Currently, openAI doesn't release or publish its zero-shot CLIP training code. Once the open AI releases and publish its training codes which are used to train their zero-shot learning models between images and texts, I will plan to use those codes to train on all medical dataset in the MedMNIST benchmark dataset and fine-tune them. Then I can have a medical domain adaptation zero-shot model. Then I test and apply them to some other medical dataset to annotate them and evaluate the results. In this case, the results should be much better than the general ImageNet-based zero-shot performance.

Expand a binary annotation task to a multiple category task. So far, I only tested one pairwise dataset between chest and breast, but I didn't test multiple category tasks and all pairwise datasets in the MedMNIST and all different kinds of CNN models. So, I will plan to test more pairwise datasets, like Blood and Derma, and use different kinds of CNN models, for example, adding a more convolutional neural network, changing output channel size, filtering size, and adding more linear layers and dropout layers. There are many different kinds of CNN designs I can try. What's more, it is to change the gradient update method. I used the stochastic gradient descent method, SGD, but I can try Adam, which is a more popular and influential way of updating gradients.

Citation

1. Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., … Sutskever, I. (2021). *Learning Transferable Visual Models From Natural Language Supervision*. doi:10.48550/ARXIV.2103.00020
2. O'Shea, K., & Nash, R. (2015). *An Introduction to Convolutional Neural Networks*. doi:10.48550/ARXIV.1511.08458
3. Yang, J., Shi, R., Wei, D., Liu, Z., Zhao, L., Ke, B., … Ni, B. (2021). *MedMNIST v2: A Large-Scale Lightweight Benchmark for 2D and 3D Biomedical Image Classification*. doi:10.48550/ARXIV.2110.14795