Deep learning PyThorch project

Creation of artificial images (i.e. data augmentation step) for Lymphoma diagnosis by fine tuning the popular stable diffusion model.1**. Abstract**

The classification of medical images poses a significant challenge due to the limited number of available examples for setting accurate classification parameters. In this research, we aim to address this problem by implementing an algorithm that generates artificial generated images derived from the original dataset. Our study focuses on the classification of lymphoma, a type of malignant lymphatic cancer. Our dataset is comprising 375 images categorized into three distinct classes: CLL (chronic lymphocytic leukemia), FL (follicular lymphoma), and MCL (mantle cell lymphoma).

To augment the dataset and improve classification accuracy, we propose the use of artificial images created through fine-tuning stable diffusion. These artificial images, combined with the original dataset, will be utilized in training a convolutional neural network (CNN) to classify the medical images into the three aforementioned lymphoma classes. The integration of artificial images into the training process aims to enhance the model's ability to generalize and accurately classify unseen instances. Additionally, we explore the effectiveness of classical data augmentation techniques such as Discrete Cosine Transform (DCT) as a benchmark for comparison.

By leveraging stable diffusion, our algorithm generates synthetic images that exhibit characteristics similar to the original lymphoma images. These artificially generated images are expected to introduce additional variations, augmenting the dataset and enabling the CNN to learn robust and discriminative features for lymphoma classification.

The effectiveness of our proposed approach will be evaluated through extensive experiments, comparing the performance of the CNN trained on the original dataset alone with the performance of the CNN trained on two combined datasets: one consisting of the original and artificial images, and another comprising the original and DCT-augmented images.

We anticipate that the integration of artificial images will lead to improved classification accuracy and demonstrate the potential of data augmentation techniques in addressing the challenge of limited training examples for medical image classification.

Explanation of existing data augmentation techniques

Data augmentation techniques are commonly used in image classification tasks to artificially increase the size of the training dataset and improve the model's ability to generalize. One such technique is Discrete Cosine Transform (DCT), which is widely used in image compression but can also be applied for data augmentation.

DCT-based data augmentation involves applying the Discrete Cosine Transform to an image, manipulating the transformed coefficients, and then applying the inverse DCT to obtain the augmented image. This process introduces variations to the image while preserving its visual content. We generated datasets with two DCT methods that will act as benchmarks for evaluating our novel idea of creating artificial images with a fine-tuned stable diffusion.

Simple explanation of CNN

We first had to started by developping a Convolutional Neural Network (CNN) model for our specific classification task. To achieve this, we employed transfer learning, a popular technique in deep learning, using the well-known AlexNet architecture as our base model. The training of the model will be done on three different datasets. The original one, the one augmented with classical DCT techniques and lastly with the artificial generated one from the fine-tuned stable diffusion model. The first and second trainings will be used as benchmarks to measure our results from the training with our artificially generated images.

Transfer learning involves leveraging pre-trained models that have been trained on large-scale datasets and have learned general features from diverse images. By using a pre-trained model as a starting point, we can benefit from the learned representations and adapt them to our specific task, thus saving significant computation time and resources.

To tailor the model to our specific classification problem, we made modifications to the final layers of the AlexNet model. We replaced the last fully connected layer with a new layer that matches the number of classes in our dataset. This allows the model to output predictions based on our target classes. Additionally, we added a log-softmax activation layer to normalize the predicted class probabilities.

To expedite the training process, we froze the parameters of the pre-trained layers, ensuring that they would remain fixed during training. By doing so, we focused our optimization efforts on updating the parameters of the newly added layers only. This strategy not only reduces training time but also helps prevent overfitting by retaining the valuable features learned by the pre-trained model.

To train and evaluate the model, we employed an Adam optimizer, a popular choice for optimizing neural networks, and used the negative log-likelihood loss criterion, suitable for multi-class classification tasks.

During the training process, we iterated over a specified number of epochs and performed both training and validation steps within each epoch. We calculated the loss and accuracy for each batch, updating the model's parameters accordingly. After each epoch, we computed the average loss and accuracy for both the training and validation sets.

The trained model and the history of the training process, including the loss and accuracy metrics, were saved for further analysis and evaluation. The resulting model can be used for inference on new images, enabling accurate predictions for the given classification problem.

Explaination of how stable diffusion works

In GANs, a large generator network is trained on a dataset to produce images. Another network, the discriminator, will have to distinguish between real and fake images. Training involves providing both real and fake images to the discriminator, which learns to differentiate them. However, GANs can be challenging to train due to issues like mode collapse, where the generator produces the same image repeatedly, and the lack of incentive for the network to generate diverse and interesting images.

In diffusion models, the process is broken down into smaller steps. The initial image is gradually distorted by adding noise at each step. The amount of noise added can follow a linear schedule, where the same amount is added at each step, or a non-linear schedule, where the noise amount varies. This noise addition process can be controlled to create different levels of noise in the image.

The training algorithm of a diffusion model, aims to deploy a network that can reverse this noise addition process and recover the original image. Rather than directly predicting the original image, the network predicts the noise that was added at each step. By subtracting this predicted noise from the noisy image, an estimate of the original image can be obtained. This estimation process is performed iteratively, gradually reducing the noise in each step until the network reaches an image that resembles the original.

The complexity of diffusion models increases when it comes to guiding the generation process. By conditioning the network on additional information, such as a caption, it becomes possible to direct the generation towards specific concepts or ideas. The text can be embedded using techniques like transformer embeddings similar to those used in GPT models.

3. Project idea

Use AI generated image from fine tuning stable diffusion as a data augmentation technique to create new images. Small explanaton of the different fine tuning techniques for stable diffusion.

A Stable Diffusion model can be decomposed into several key models:

A text encoder that projects the input prompt to a latent space. (The caption associated with an image is referred to as the "prompt".)

A variational autoencoder (VAE) that projects an input image to a latent space acting as an image vector space.

A diffusion model that refines a latent vector and produces another latent vector, conditioned on the encoded text prompt

A decoder that generates images given a latent vector from the diffusion model.

Either you fine tune the diffusion model (what we did) or another fine tuning technique which is called textual inversion. Textual inversion works by learning a token embedding for a new text token, keeping the remaining components of Stable Diffusion frozen.

These are the different steps of fine tuning the diffusion model of stable diffusion :

An input text prompt is projected to a latent space by the text encoder.

An input image is projected to a latent space by the image encoder portion of the VAE.

A small amount of noise is added to the image latent vector for a given timestep.

The diffusion model uses latent vectors from these two spaces along with a timestep embedding to predict the noise that was added to the image latent.

|  |  |  |  |
| --- | --- | --- | --- |
| Artificial Generated |  | Loss | Accuracy |
| Epoch 1 | Training |  |  |
|  | Validation |  |  |
| Epoch 2 | Training |  |  |
|  | Validation |  |  |
| Epoch 3 | Training |  |  |
|  | Validation |  |  |
| Epoch 4 | Training |  |  |
|  | Validation |  |  |
| Epoch 5 | Training |  |  |
|  | Validation |  |  |

A reconstruction loss is calculated between the predicted noise and the original noise added in step 3.

|  |  |  |  |
| --- | --- | --- | --- |
| Base images |  | Loss | Accuracy (%) |
| Epoch 1 | Training | 1.2144 | 34.49 |
|  | Validation | 1.1670 | 48 |
| Epoch 2 | Training | 1.0146 | 50.67 |
|  | Validation | 0.9904 | 54.67 |
| Epoch 3 | Training | 0.9267 | 55.18 |
|  | Validation | 0.9801 | 48 |
| Epoch 4 | Training | 0.8704 | 56.19 |
|  | Validation | 0.9946 | 44 |
| Epoch 5 | Training | 0.8630 | 60.87 |
|  | Validation | 0.9118 | 56 |

Finally, the diffusion model parameters are optimized w.r.t this loss using gradient descent.

Note that only the diffusion model parameters are updated during fine-tuning, while the (pre-trained) text and the image encoders are kept frozen.

4. Realisation

Have a simple CNN

* See how it performs without DA
* How it performs with DCT image transforms

Have stable diffusion fine tuned with our small image dataset

* Train CNN with new Artificial images generated

5. Results

Compare result of 3 methods

AI dataset :

Epoch: 1/5

Epoch : 001, Training: Loss: 1.1206, Accuracy: 41.8831%,

Validation : Loss : 1.0546, Accuracy: 50.6667%, Time: 11.8292s

Epoch: 2/5

Epoch : 002, Training: Loss: 0.9651, Accuracy: 50.9740%,

Validation : Loss : 1.0060, Accuracy: 50.6667%, Time: 10.0452s

Epoch: 3/5

Epoch : 003, Training: Loss: 0.9296, Accuracy: 55.1948%,

Validation : Loss : 0.8987, Accuracy: 57.3333%, Time: 9.9192s

Epoch: 4/5

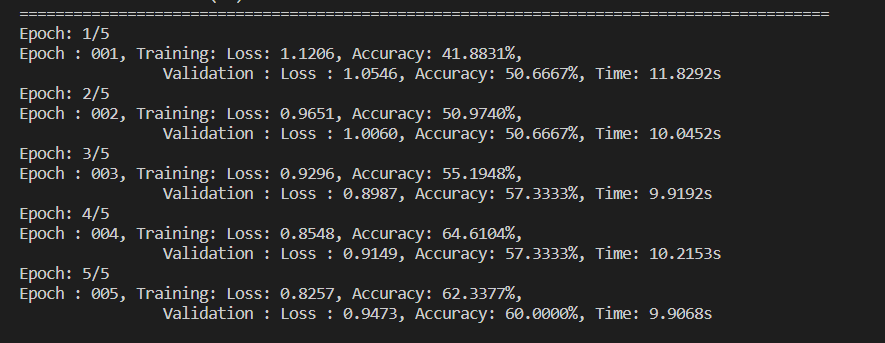
Epoch : 004, Training: Loss: 0.8548, Accuracy: 64.6104%,

Validation : Loss : 0.9149, Accuracy: 57.3333%, Time: 10.2153s

Epoch: 5/5

Epoch : 005, Training: Loss: 0.8257, Accuracy: 62.3377%,

Validation : Loss : 0.9473, Accuracy: 60.0000%, Time: 9.9068s



6. Conclusion

What was positive

What was in our way

What can be done it the future