Generative Adversarial Networks for Medical Imaging

CSYE 7245 Big Data and Intelligence Analytics

Vignesh Murali, Rishi Rajani, Rajeshree Kale

Abstract

In this project, we observe the performance of Generative Adversarial Networks for medical imaging. In healthcare, there are a lot of privacy concerns which prevent researchers from accessing private medical scans and images of patients. Hence, with the little medical image data we have we would like to use GANs to create synthetic medical images. We trained our GANs on malaria blood cells (2 categories: parasitized, infected) for 15,000 epochs. We tested these synthetic images on a very accurate (95% validation accuracy) ResNet CNN which was trained on our original dataset. 30% of our parasitized cells were accurately predicted and 99% of our uninfected cells were accurately predicted.

Introduction

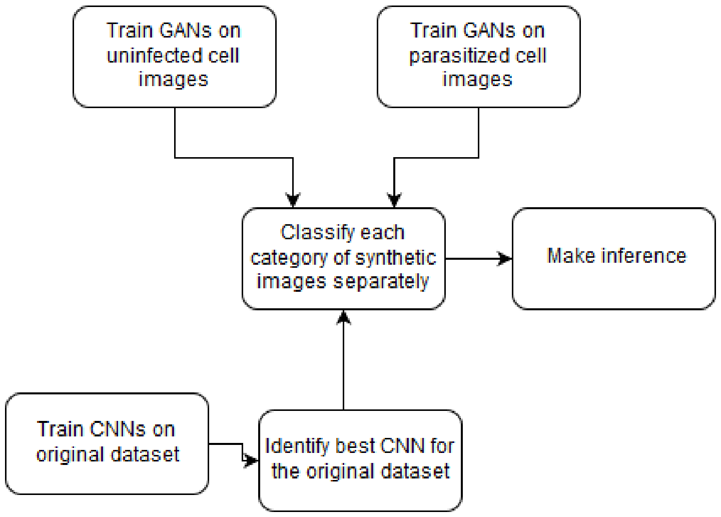
The healthcare industry provides its payers with great levels of security and privacy. This is obviously a very positive deed but from a research point of view it is disastrous. There is not enough medical data (such as blood cell scans, MRI scans etc.) for researchers to analyze and make bigger healthcare breakthroughs.

We aim to solve this problem by utilizing GANs (Generative Adversarial Networks). GANs can be taught to create data eerily similar to our own in any domain: images, music, speech, prose. They are robot artists in a sense, and their output is usually impressive.

GANs can possibly be used to solve the problem of lack of medical data and the synthetic images generated by them can ge used for research without privacy concerns as they are generated synthetically and do not belong to real human beings.

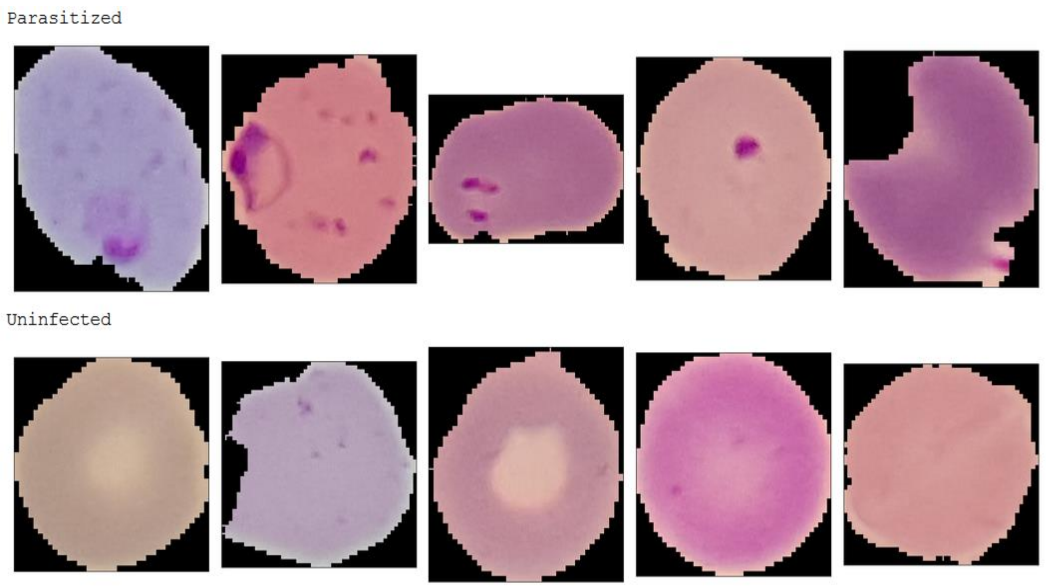
Methods

Our workflow is as follows:


Train GANs on uninfected cell images
Train GANs on parasitized cell images
Classify each category of synthetic images separately
Make inference
Train CNNs on original dataset
Identify best CNN for the original dataset

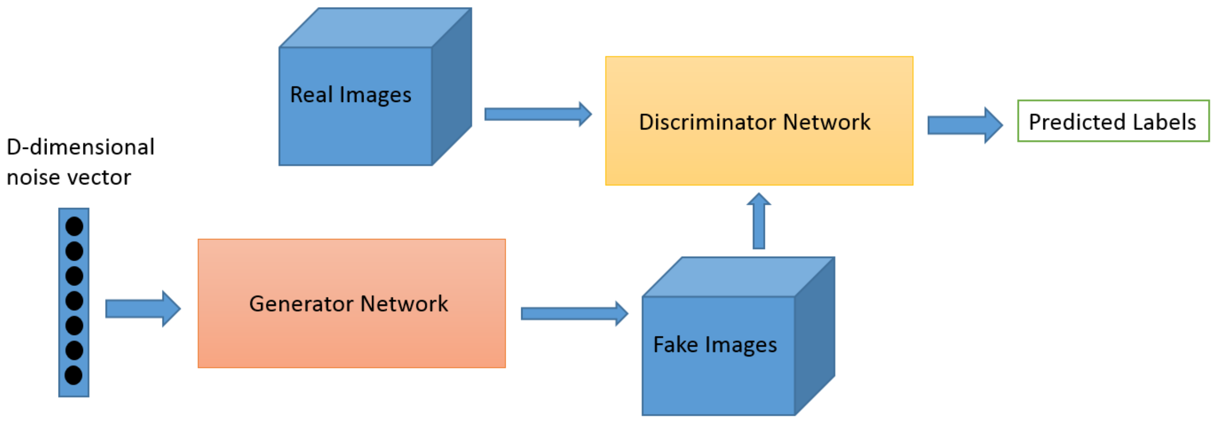
Dataset

We are using a dataset of labeled and preprocessed images to train and evaluate our model. NIH has a malaria dataset of 27,558 cell images with an equal number of parasitized and uninfected cells. A level-set based algorithm was applied to detect and segment the red blood cells.
Dataset : https://ceb.nlm.nih.gov/repositories/malaria-datasets/



We can clearly observe that the parasitized cells have purple plasmic infections inside them whereas the uninfected cells appear to be plain. So, the differences are clearly distinguishable.

GAN's Architecture

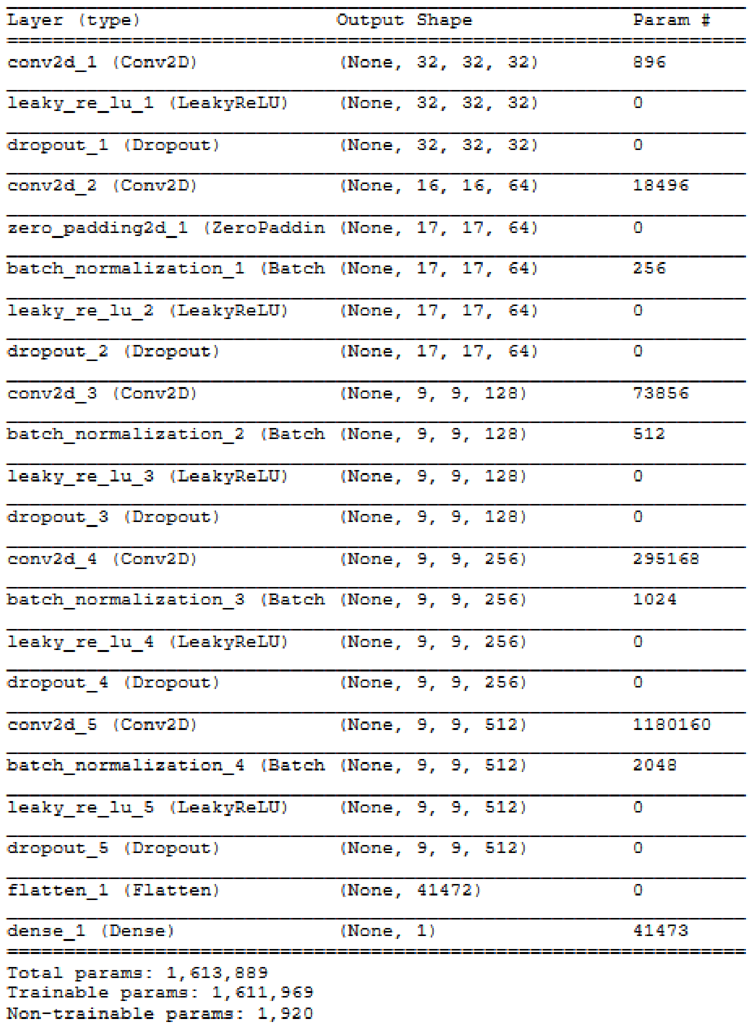


The above image was referenced from https://towardsdatascience.com/generative-adversarial-networks-gans-a-beginners-guide-5b38eceece24

High level overview of how GANs work:

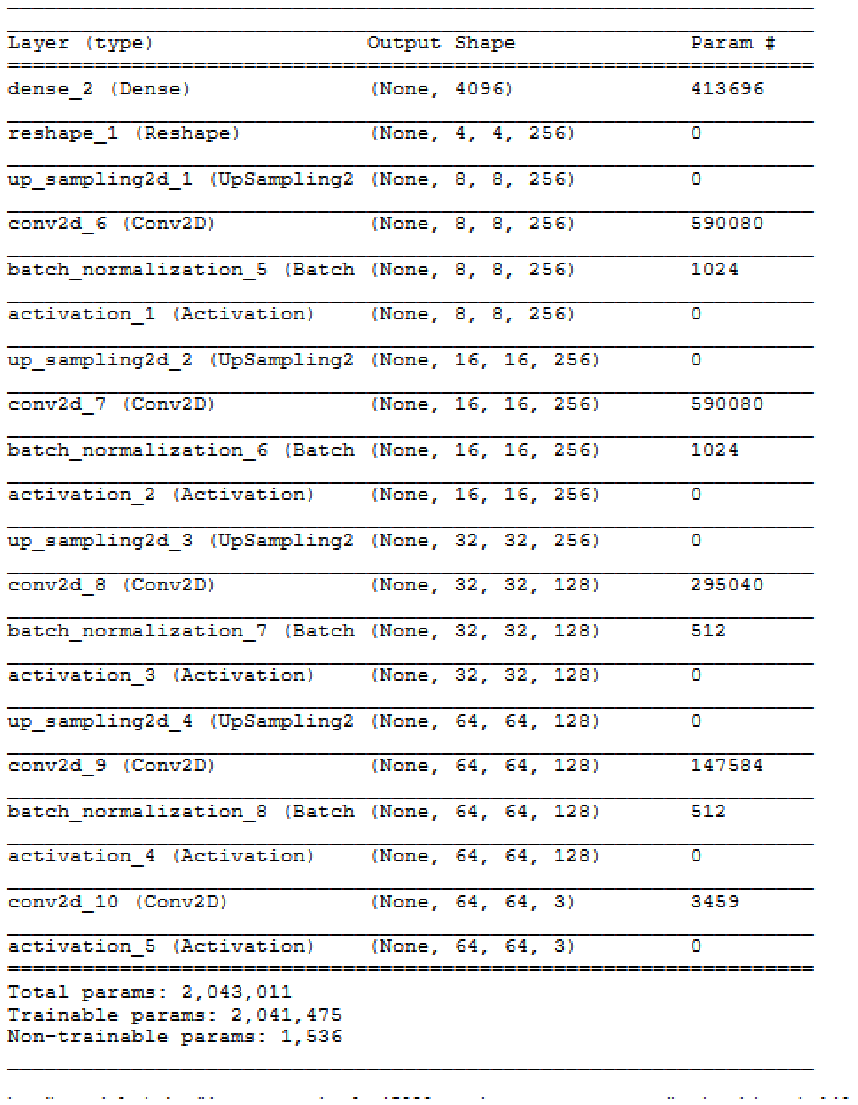
- The generator takes in random numbers and returns an image.
- This generated image is fed into the discriminator alongside a stream of images taken from the original dataset.
- The discriminator takes in both real and fake images and returns probabilities, a number between 0 and 1, with 1 representing a prediction of authenticity and 0 representing fake.
- There is a feedback loop between the generator and the discriminator.

Generator:



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 32, 32, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
zero_padding2d_1 (ZeroPaddin (None, 17, 17, 64)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 17, 17, 64)        256
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 17, 17, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 17, 17, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 9, 9, 128)         73856
_________________________________________________________________
batch_normalization_2 (Batch (None, 9, 9, 128)         512
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 9, 9, 128)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 9, 9, 128)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 9, 9, 256)         295168
_________________________________________________________________
batch_normalization_3 (Batch (None, 9, 9, 256)         1024
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 9, 9, 256)         0
_________________________________________________________________
dropout_4 (Dropout)          (None, 9, 9, 256)         0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 9, 9, 512)         1180160
_________________________________________________________________
batch_normalization_4 (Batch (None, 9, 9, 512)         2048
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 9, 9, 512)         0
_________________________________________________________________
dropout_5 (Dropout)          (None, 9, 9, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 41472)             0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 41473
=================================================================
Total params: 1,613,889
Trainable params: 1,611,969
Non-trainable params: 1,920
```

Discriminator:



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_2 (Dense)              (None, 4096)              413696    
_________________________________________________________________
reshape_1 (Reshape)          (None, 4, 4, 256)         0         
_________________________________________________________________
up_sampling2d_1 (UpSampling2 (None, 8, 8, 256)         0         
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 8, 8, 256)         590080    
_________________________________________________________________
batch_normalization_5 (Batch (None, 8, 8, 256)         1024      
_________________________________________________________________
activation_1 (Activation)    (None, 8, 8, 256)         0         
_________________________________________________________________
up_sampling2d_2 (UpSampling2 (None, 16, 16, 256)       0         
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 16, 16, 256)       590080    
_________________________________________________________________
batch_normalization_6 (Batch (None, 16, 16, 256)       1024      
_________________________________________________________________
activation_2 (Activation)    (None, 16, 16, 256)       0         
_________________________________________________________________
up_sampling2d_3 (UpSampling2 (None, 32, 32, 256)       0         
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 32, 32, 128)       295040    
_________________________________________________________________
batch_normalization_7 (Batch (None, 32, 32, 128)       512       
_________________________________________________________________
activation_3 (Activation)    (None, 32, 32, 128)       0         
_________________________________________________________________
up_sampling2d_4 (UpSampling2 (None, 64, 64, 128)       0         
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 64, 64, 128)       147584    
_________________________________________________________________
batch_normalization_8 (Batch (None, 64, 64, 128)       512       
_________________________________________________________________
activation_4 (Activation)    (None, 64, 64, 128)       0         
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 64, 64, 3)         3459      
_________________________________________________________________
activation_5 (Activation)    (None, 64, 64, 3)         0         
=================================================================
Total params: 2,043,011
Trainable params: 2,041,475
Non-trainable params: 1,536
_________________________________________________________________
```

Some pointers about our GAN architecture:

- Padding: 'same' means that output size in convolution layer remains same as input size.
- Batch normalization: Normalizes the output of each layer to combat internal covariate shift.
- Upsampling: It resizes the image back to its original state. The sole purpose of upsampling layers is to reduce computations in each layer, while keeping the dimension of input/output as before.
- We reshape after input layer of generator because CNNs require input in the shape

Training workflow:

- Normalize pixel values
- Set labels for real & generated [0 & 1] for the discriminator to identify images.
- Select random batch of images.
- Create random noise input based on batch size.
- Generator generates images using random noise input.
- Train discriminator on real images & on fake images so it can tell the difference.
- Generator is trained on combined model with feedback from discriminator to improve quality.
- Model is saved

After training our GANs we test these synthetic images on a ResNet CNN architecture which has been trained on our original dataset.

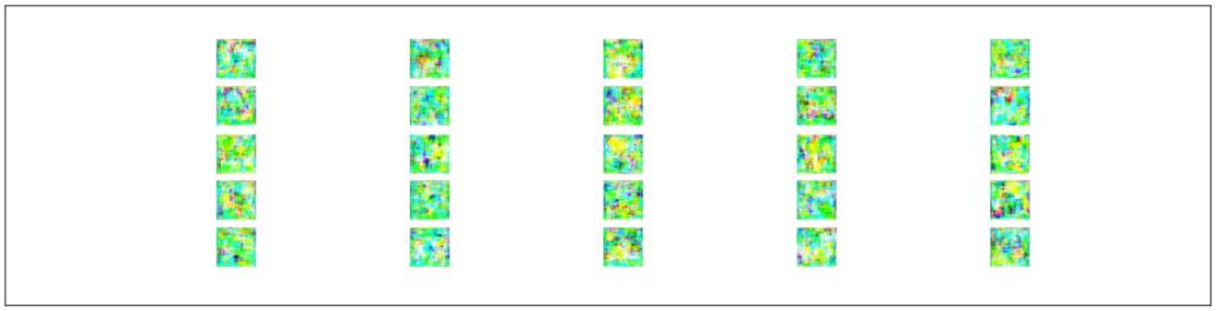
We initialize ResNet with the following parameters:

- Images are 64 x 64 x 3 (length, width, and depth—3-channel RGB images)
- 2 classes (Parasitized & Uninfected)
- Stages = (3, 4, 6)
- Filters = (64, 128, 256, 512)
- This implies that the first CONV layer (before reducing spatial dimensions) will have 64 total filters.

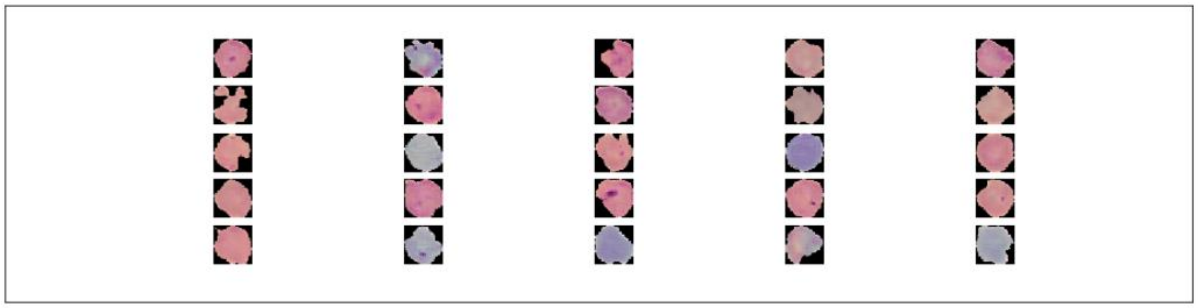
Results

GAN results parasitized:

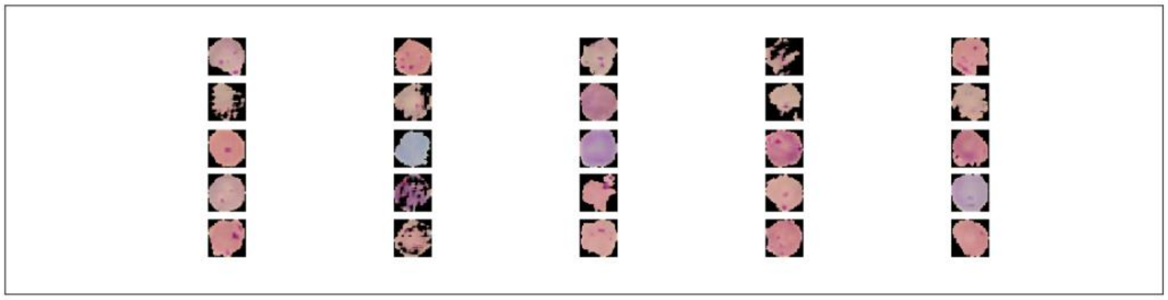
1st Epoch:



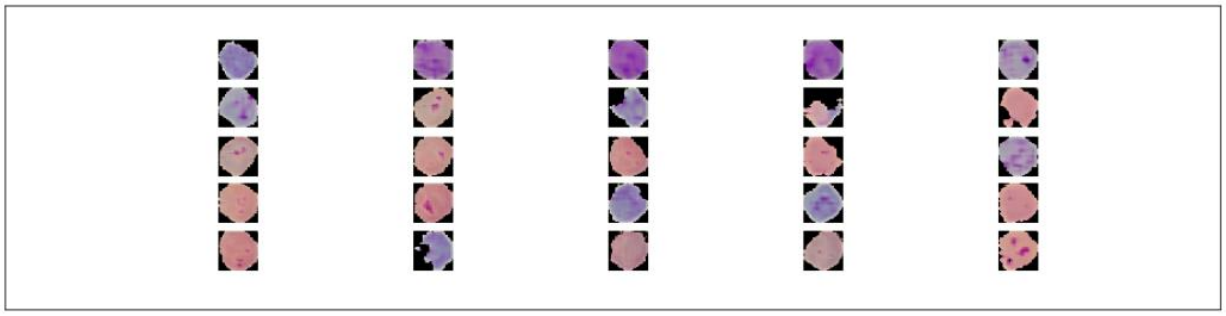
5000th Epoch:



10000th Epoch:



15000th Epoch:

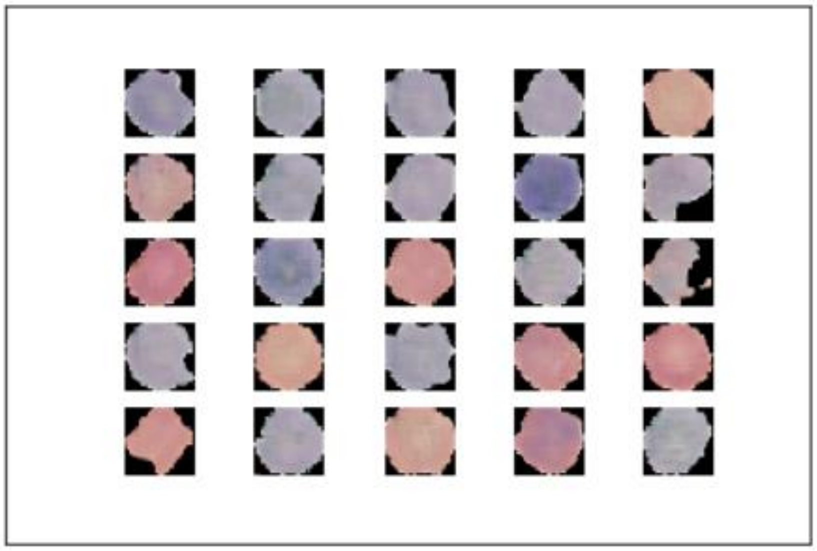


At the 15000th Epoch we can clearly see the plasmic infections inside each cell.

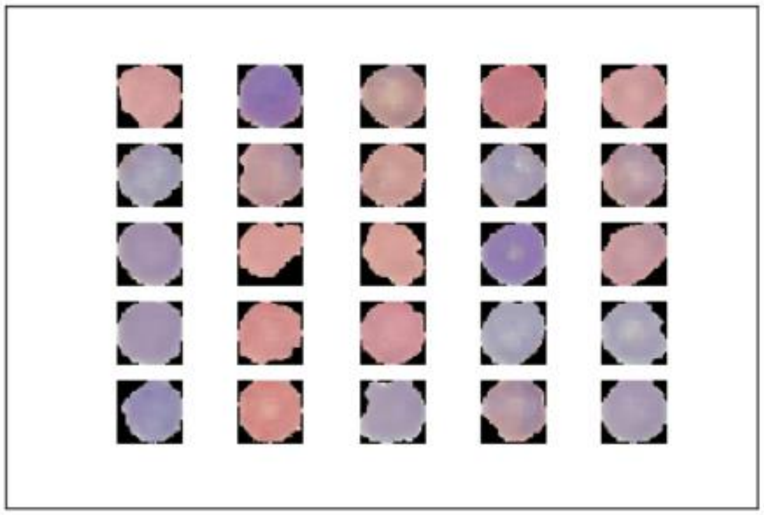
GAN results uninfected

1st Epoch:

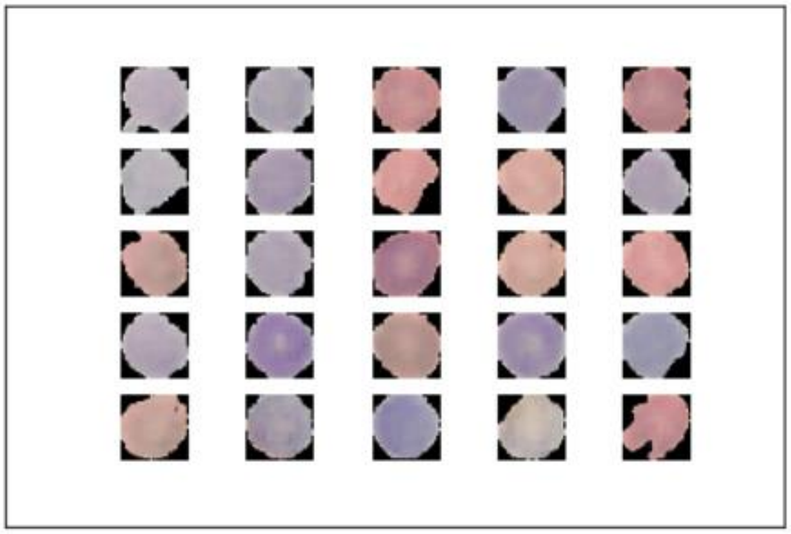
5000th Epoch:



10000th Epoch:

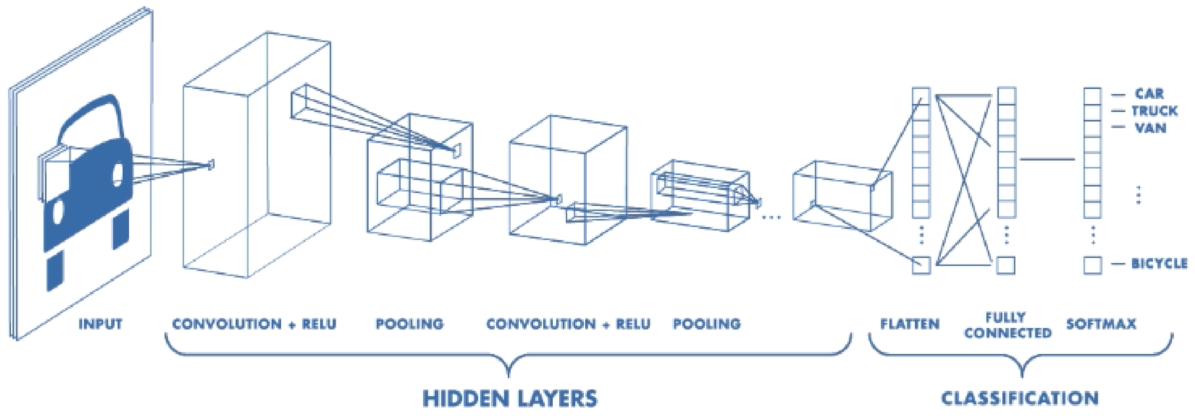


15000th Epoch:



Convolutional Neural Network

Convolutional Neural Networks are very similar to ordinary Neural Networks from the previous chapter: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply.



The above image was referenced from : https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

Model Parameters

- Dimensions of image for the custom CNN
- RGB
- Binary classification
- Modify based on the GPUs in your system
- Modify depending on the model convergence with your data

The CNN model architecture:

```
_________________________________________________________________
Layer (type)              Output Shape            Param #
=================================================================
=
conv2d_1 (Conv2D)           (None, 62, 62, 32)       896
_________________________________________________________________
activation_1 (Activation)    (None, 62, 62, 32)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 31, 31, 32)       0
_________________________________________________________________
conv2d_2 (Conv2D)           (None, 29, 29, 32)       9248
_________________________________________________________________
activation_2 (Activation)    (None, 29, 29, 32)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 32)       0
_________________________________________________________________
conv2d_3 (Conv2D)           (None, 12, 12, 64)       18496
_________________________________________________________________
activation_3 (Activation)    (None, 12, 12, 64)       0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 64)         0
_________________________________________________________________
flatten_1 (Flatten)         (None, 2304)             0
_________________________________________________________________
dense_1 (Dense)             (None, 64)               147520
_________________________________________________________________
dense_2 (Dense)             (None, 2)                130
=================================================================
=
Total params: 176,290
Trainable params: 176,290
Non-trainable params: 0
_________________________________________________________________
```

The Malaria Detection dataset using convolutional neural network gave an accuracy of 52.21% for 30 epochs.The next model gave an accuracy 53.31% with 100 epochs and the final model with 50 epochs gave accuracy of 52%. The model parameters used were:

- Activation Function: Rectified linear unit (ReLU)
- Cost function: categorical_crossentropy,binary_crossentropy
- No.of Epochs: 30,50,100

- Gradient estimation: Stochastic Gradient Descent

To improve the model we tuned the hyperparameters like: reducing the learning rate,adding number of layers,training the model on more number of epochs,adding the data augmentation. Also, number of neurons can be more complicated for a better fit.Along with that all the parameters mentioned above can also be changed in order to improve the model accuracy.

Results of CNN models

Number	Model Name	No. of Epochs	Accuracy
1	CNN Model 1	30	0.5221
2	CNN Model 2	100	0.5331
3	CNN Model 3	50	0.5200

The best CNN model was selected for building the ResNet Network which will then be used for testing the images on GAN's.

Model Architecture (ResNet)

We initialize ResNet with the following parameters:

- Images are 64 x 64 x 3 (length, width, and depth—3-channel RGB images)
- 2 classes (Parasitised & Unparasitised)
- Stages = (3, 4, 6)
- Filters = (64, 128, 256, 512)
- This implies that the first CONV layer (before reducing spatial dimensions) will have 64 total filters.

The CNN model was then built on ResNet network which consisted of:

- Stages = (3, 4, 6)
- Filters = (64, 128, 256, 512)

The model gave an accuracy of around 95% with only 4 epochs. This shows that the model has trained well and gave a very good accuracy with less number of epochs. It will be further used for the testing of GAN's.On those basis we can decide how correctly the network performs in classifying the images of Malaria Dataset cells.

Number	Model Name	No. of Epochs	Accuracy
1	CNN Model 1	30	0.5221
2	CNN Model 2	100	0.5331
3	CNN Model 3	50	0.5200
4	**Best CNN Model**	**30**	**0.5548**
5	ResNet Model	4	0.9489

Testing GANs on our trained ResNet model

Out of 1000 parasitized images, 300 were classified correctly.
Out of 1000 uninfected images, 991 were classified correctly.

Generated Images Parasitised:	300/1000
Generated Images Uninfected:	991/1000
Predicted as Parasites	30%
Predicted as unifected	99.1%

Discussion

We noticed that after predicting the labels of our synthetic images, 30% of the parasitized synthetic images were accurately predicted and 99% of uninfected images were classified as uninfected.

We can infer that our GAN architecture works extremely well with simpler images as witnessed with the uninfected images.

But with more complex images where the features of the image are not as easy to capture as seen in the parasitized images, the GAN architecture is not as effective.

With significant tuning of hyperparameters and further preprocessing, we can probably achieve better results with complex image structures as well.

Scope

Deep learning has made many advancements today especially in the medical industry today. However, we are still facing some problem with regard to availability and confidentiality
of unmarked data. Due to the lack of data it becomes difficult for us to apply different kind of deep learning networks or any machine learning algorithms to diagnose the disease.

Therefore, we decided to use GAN's to generate synthetic images of infected and uninfected blood cells and then classify them correctly. Before we get into the classification of our synthetic images, we train different CNN's on the original dataset to figure out the best model. Once we have the best model we use it on our synthetic data to check how effectively our GAN has generated the images.

We have tuned the GAN's to perform well and then tested it on our data. Additionally, we performed hyperparameter tuning to increase the effectiveness of our network but unfortunately the quality of the images deteriorated and were not acceptable. This approach was something that the team jointly decided and was not observed in any of the research papers we read.

Context

GANs were adapted from: https://gist.github.com/AI-Insider/41d187bec396aa0cd662820db4ae675c
This network was used for generating images of human faces in this example. We tweaked the code to fit our dataset.

ResNet code was adapted from:
https://github.com/gracelynxs/malaria-detection-model/blob/master/CNN/resnet.py
The resnet architecture was used. It was not pre-trained with weights.

Conclusion

The Malaria Detection dataset using convolutional neural network gave an accuracy of 55.48% for 30 epochs. The best CNN model was then built on ResNet network which consisted of the model that gave an accuracy of around 95% with only 4 epochs.

After noticing such high accuracy, the ResNet model was selected to evaluate our GANs.Upon predicting our GAN generated images on the trained ResNet model, we observe that 30% of our synthetic parasitized images were predicted accurately as parasitized and 99% of our uninfected images were predicted accurately as uninfected.

To conclude, it is noticed that GANs work significantly well for simpler images but in case of more complex images (like the Parasitized images which showed some plasmic infections within the cells), it gets difficult to efficiently capture minute features. However, we feel that this problem can be unraveled with a domain driven fine tuning approach.

References:

1. Dong, Yuhang, et al. "Evaluations of deep convolutional neural networks for automatic identification of malaria infected cells." *2017 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI). IEEE, 2017.*
2. Liang, Zhaohui, et al. "CNN-based image analysis for malaria diagnosis." *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). IEEE, 2016.*
3. Razzak, Muhammad Imran, Saeeda Naz, and Ahmad Zaib. "Deep learning for medical image processing: Overview, challenges and the future." *Classification in BioApps*. Springer, Cham, 2018. 323-350.
4. Kermany, Daniel S., et al. "*Identifying medical diagnoses and treatable diseases by image-based deep learning*." Cell 172.5 (2018): 1122-1131.
5. Frid-Adar, M., Diamant, I., Klang, E., Amitai, M., Goldberger, J., & Greenspan, H. (2018). GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. *Neurocomputin*g, 321, 321-331.
6. Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., ... & Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical image analysis*, 42, 60-88.
7. Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." *arXiv preprint arXiv*:1511.06434 (2015).