SWAN On-Premises for Machine Learning and Data Analysis on Medical Applications

Omar Zapata, Enric Tejedor and Pere Mato

CERN / EP-SFT

February 17th, 2020

Outline

- Ongoing KT Project
- SWAN On-Premises - ScienceBox
  - What is SWAN?
  - ScienceBox
  - GPU Support
  - Using CVMFS as a Software Source
- Medical Applications
  - Introduction
  - Medical data
  - 2D Images classification
  - 2D Images segmentation
  - Loss functions for segmentation
  - Some 2D results
  - 3D segmentation
  - 3D results with Unet
- R&D
  - 3D Unet with VAE
    - Model
    - Preliminary results
  - GANs
- Conclusions

Ongoing KT project

Ongoing KT project

- In partnership with All-In-Image, Israeli company specialized in image processing for medical diagnosis.

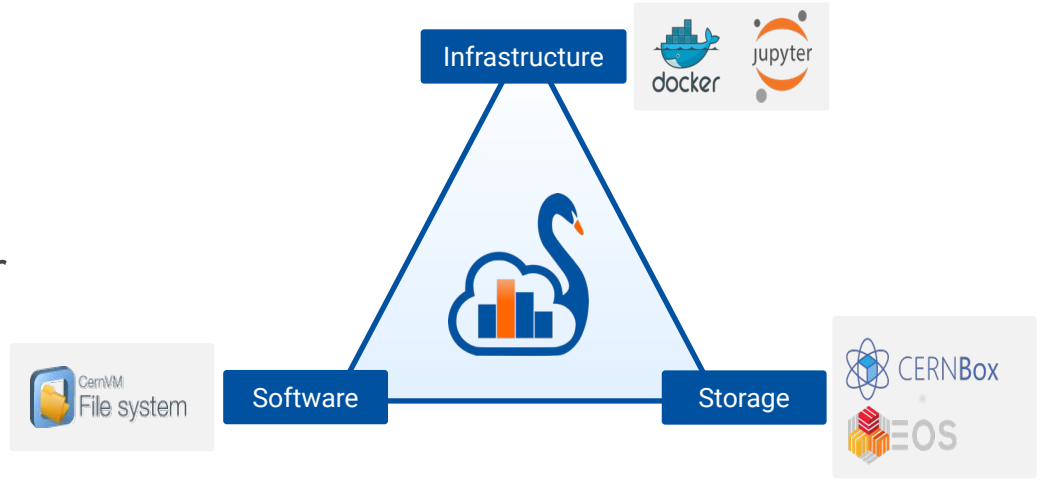
- Objectives
  - Give support to ScienceBox/SWAN to run machine learning frameworks on NVidia GPUs.
  - Create machine learning models for classification and segmentation of brain tumors.

SWAN On Premises - ScienceBox

What is SWAN?

SWAN (Service for Web based ANalysis) is a platform to perform interactive data analysis in the cloud.

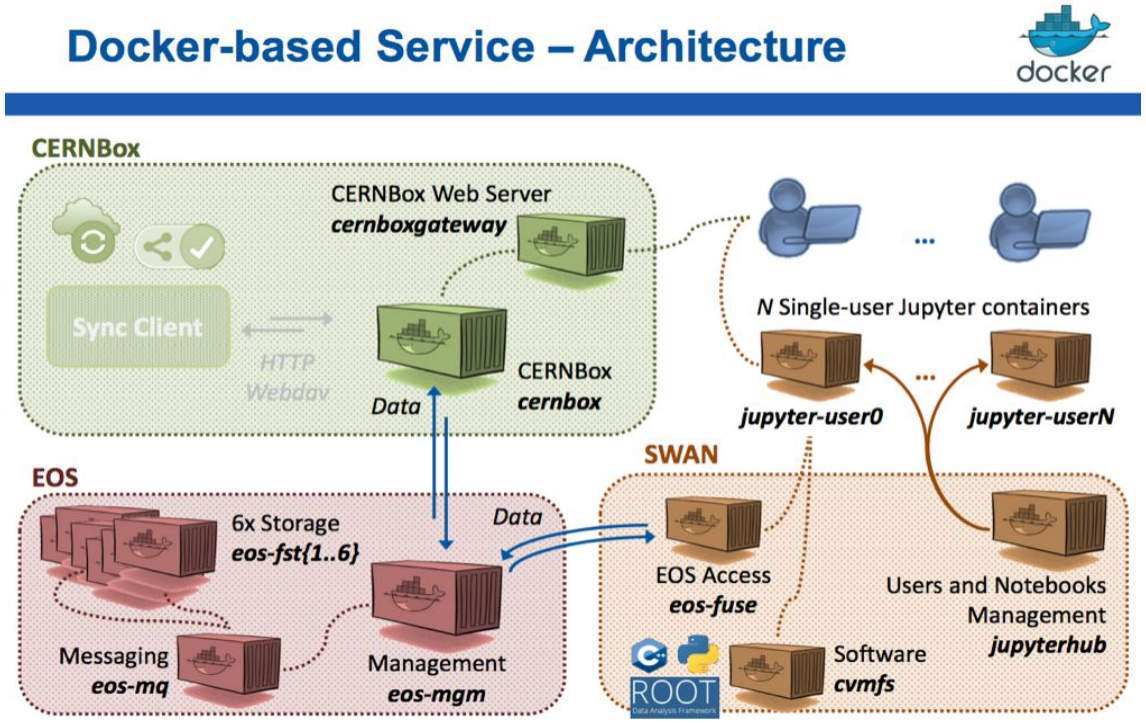
- Analyse data **without the need to install any software**
- Jupyter **notebook interface** as well as shell access from the browser
- Use **CERNBox** as your **home directory** and **synchronise** your local user storage with the cloud
- Access **experiments' and user data** in the CERN cloud (EOS)
- **Share your work** with your colleagues thanks to CERNBox
- **Document and preserve science** - create catalogues of analyses: encourage reproducible studies and learning by example
- Submit your jobs to **CERN Spark Clusters**



ScienceBox

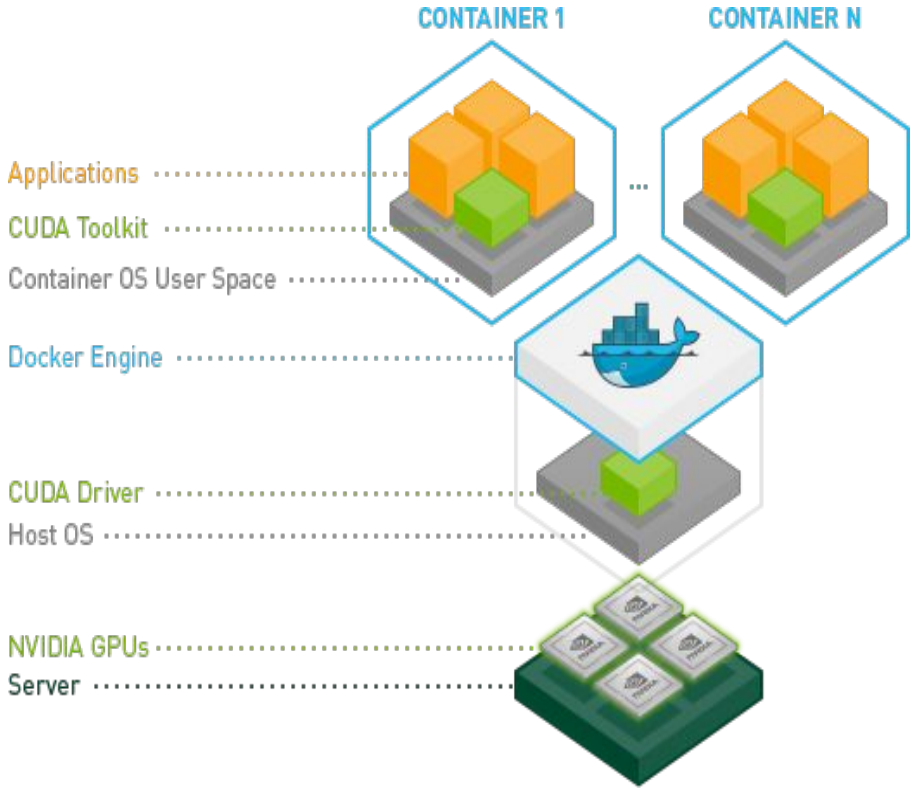
- SWAN On-premises (ScienceBox) is a packaged version of SWAN that can be easily installed in on-premises machines.
- ScienceBox can be deployed in private or public clouds like OpenStack, Amazon Web Services, Google Cloud Platform etc..
- It can even run in your personal computer or laptop.
- CERN credentials not required.

http://sciencebox.web.cern.ch/



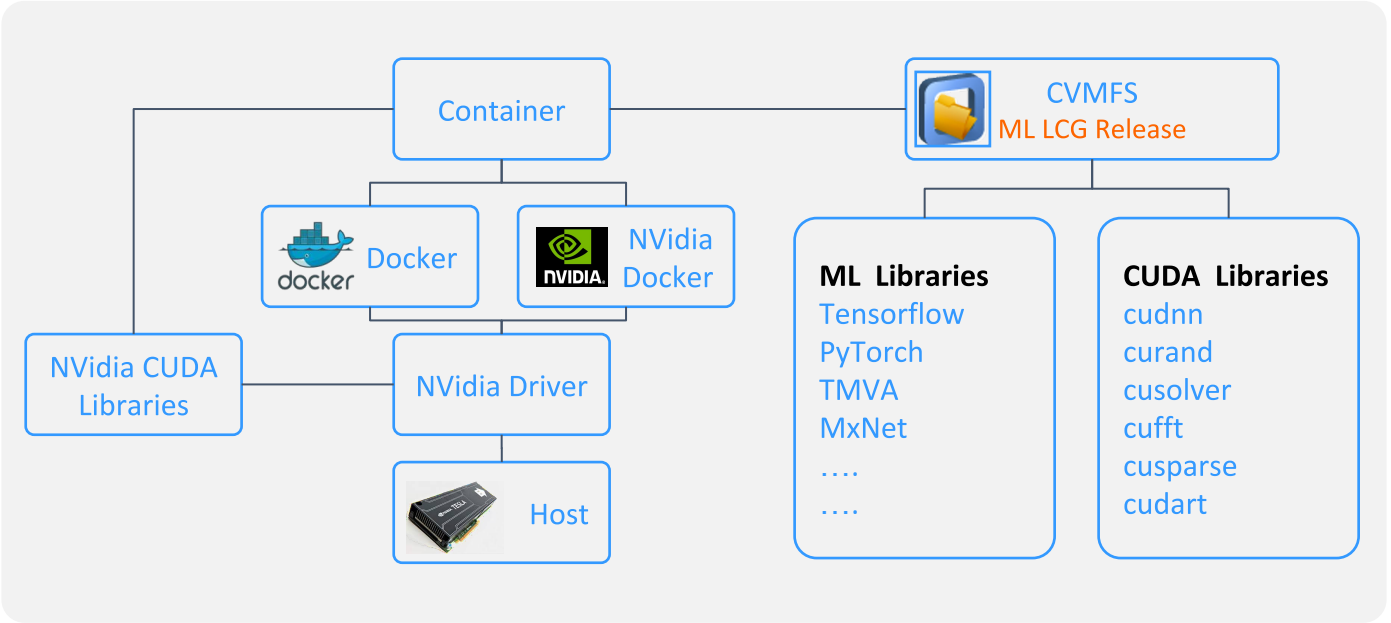
SWAN/ScienceBox GPU

- **Objective**: exploitation of GPUs from a SWAN interactive session
- Users can **attach a GPU** to their session
  - Creation of a container with NVidia GPU support
- Use of ML libraries that are Cuda-enabled to offload computations to the GPU
  - Libraries provided by a **CVMFS software stack**
- Support for docker and kubernetes
- Prototype server was deployed for testing purposes using a NVidia Tesla V100 PCIe 32GB



Using CVMFS as a Software Source

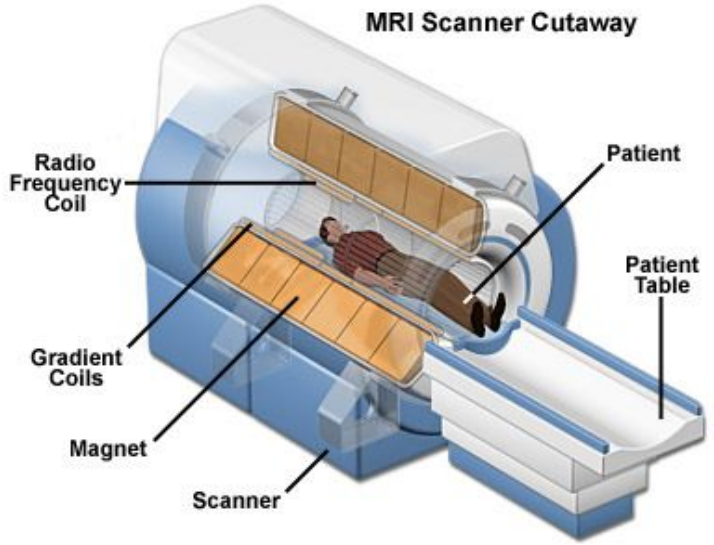
- A stack of software for machine learning was created
- Software distributed through LCG releases have:
  - CUDA dependencies like cudnn, curand, cublas etc.. except libcuda that is provided by the driver's installer in the host and it will be exported through a volume
  - Machine learning packages like Tensorflow, TMVA, PyTorch and MxNet will be included with GPU support enabled in the compilation.

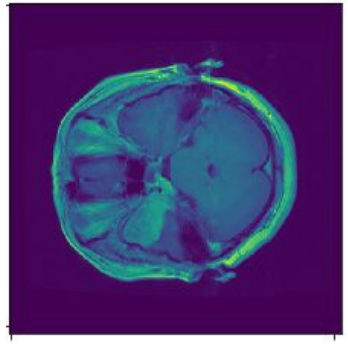


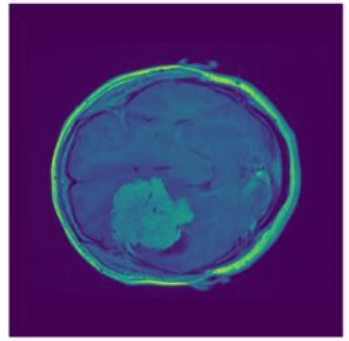
Use case: Medical applications

Medical applications introduction

- **Objective**: use MRI (Magnetic Resonance Images) technology with machine learning for:
  - Brain tumor classification
  - Brain tumor segmentation
- Our machine learning approach is using Convolutional Neural Networks (CNN) with specialized models like Deep Encoder/Decoder and Generative Adversarial Networks (GANs)
- Synergy with GPU-Enabled SWAN
  - Platform for ML studies

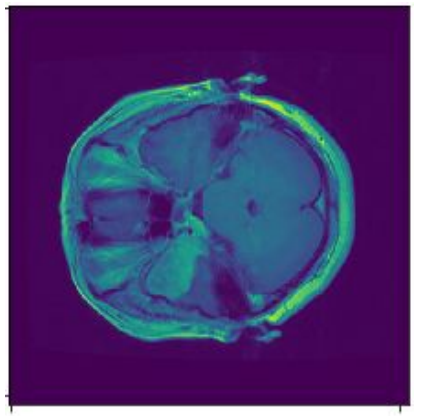




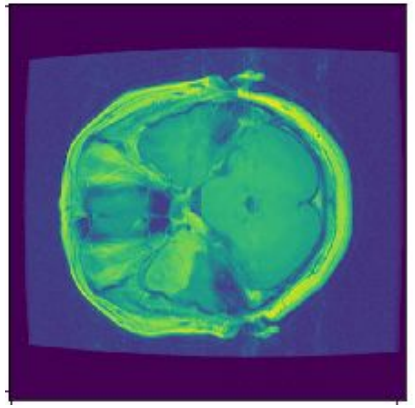


Medical data

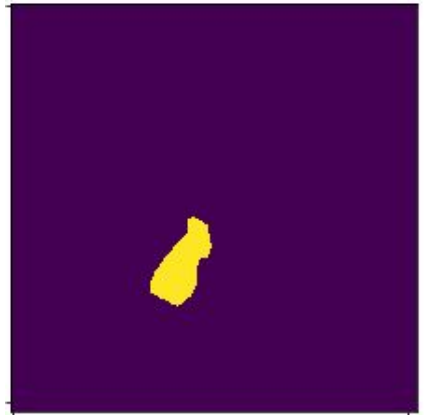
- Very restrictive for privacy policies.
- Requires a lot of preprocessing:
  - Noise removal: a lot of electromagnetic field captured in the background by the sensors. (usually produce hard numerical error propagation)
  - Skull stripping: required to improve the segmentation and classification of the tumors.
  - For segmentation, the specialist should create a mask (image with the region of the tumor) in order to train our machine learning models.
  - Requires image enhance algorithms to fix problems with calibration or poorly taken samples.
- 2D public dataset taken from [figshare]



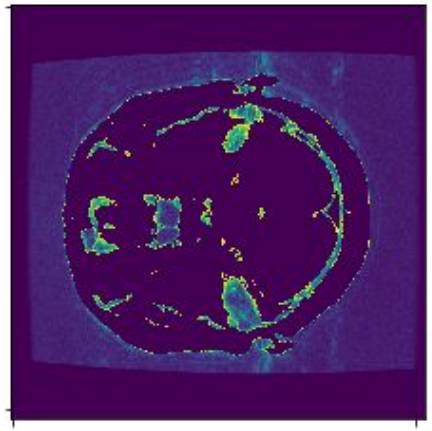
Raw Image



Revealed Noise



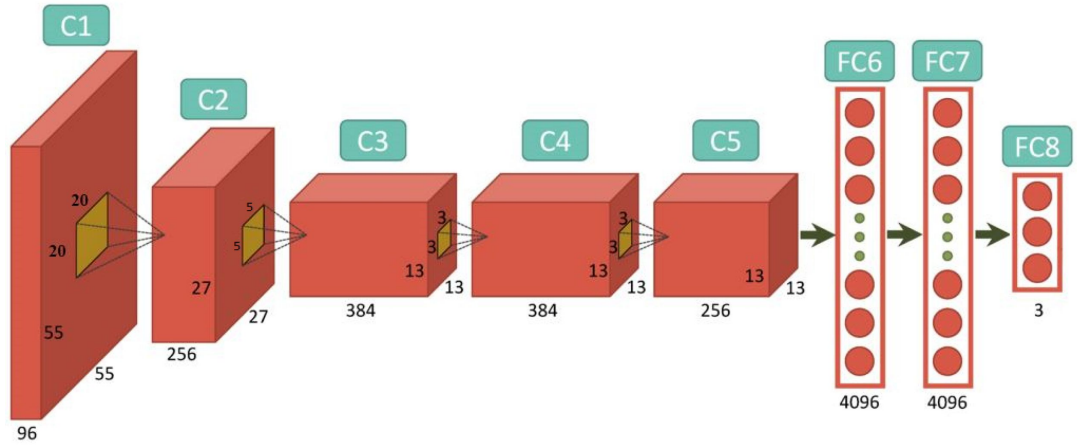
Mask



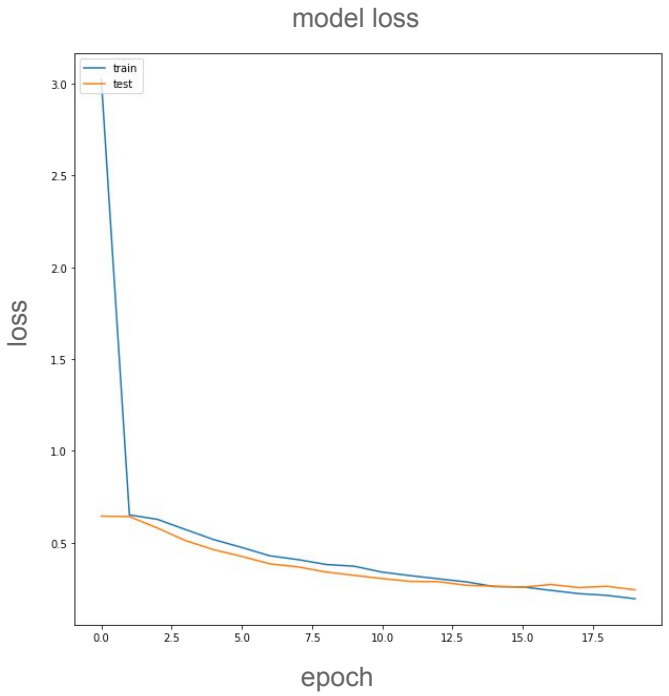
Noise removed

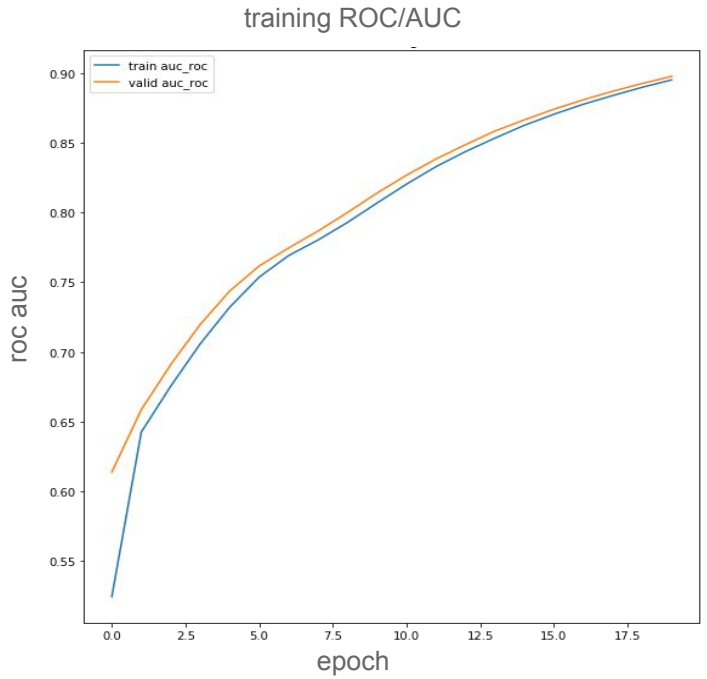
2D Images Classification

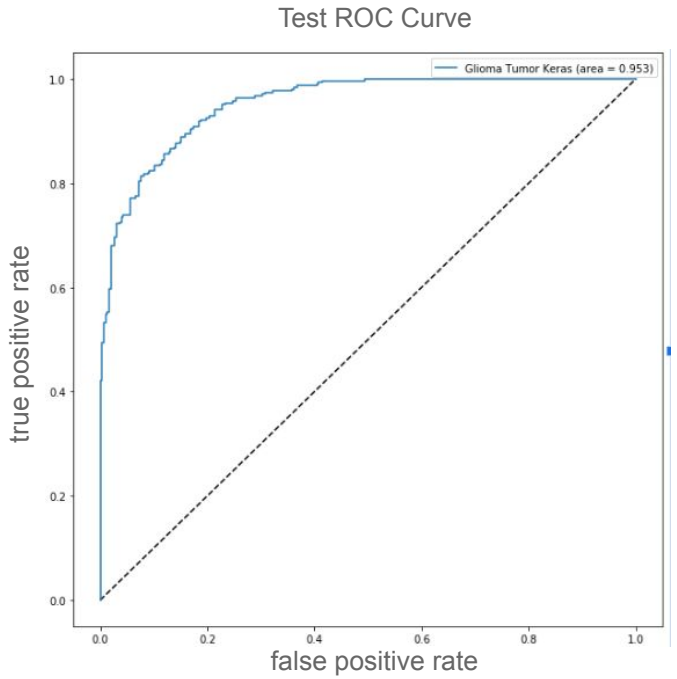
- Binary classification problem for tumors (Glioma or Not)
- 2D Convolutional Neural Network
  - Similar to an encoder with fully connected layers at the end
  - The loss function is binary cross entropy
- The hard part is to preprocess the data.



2D Images Classification



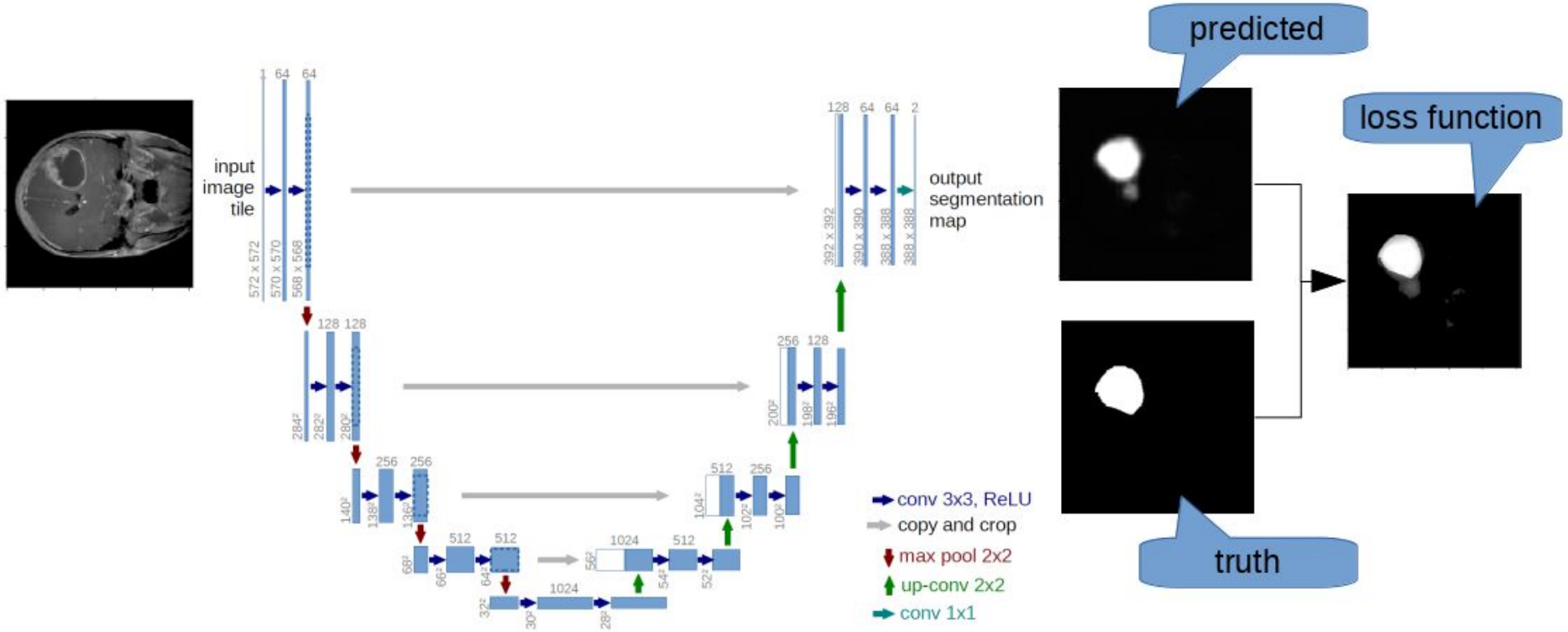




	Train	Validation	Test
Loss	0.2620	0.2997	X
Accuracy	0.8824	0.8617	X
AUC-ROC	0.85	0.86	0.953

2D images segmentation

Encoder/Decoder (Unet)



> https://link.springer.com/chapter/10.1007/978-3-319-24574-4_28

Loss functions for segmentation

- Intersection over union (fig 1)

$$IoU(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

- Dice (aka. Jaccard Index) (fig 2)

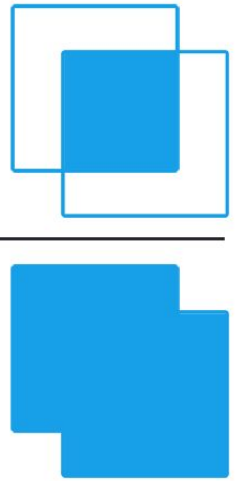
$$J(A,B) = \frac{2|A \cap B|}{|A| + |B|}$$ 2*overlap/total pixels

- Pearson CoC

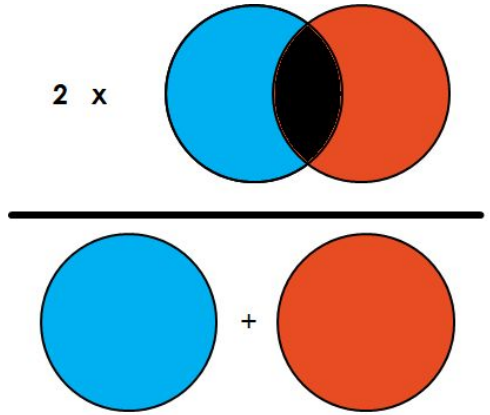
$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \qquad L(A,B) = 1 - \frac{r(A,B)+1}{2}$$

```
from tensorflow.keras import backend as K

def pearson_coc_loss(y_true, y_pred):
    x = K.flatten(y_true)
    y = K.flatten(y_pred)
    xm = K.mean(x)
    ym = K.mean(y)
    sumxy = K.sum((x-xm)*(y-ym))
    sumx = K.sum((x-xm)*(x-xm))
    sumy = K.sum((y-ym)*(y-ym))
    xn = sumxy/(K.sqrt(sumx)*K.sqrt(sumy))
    xn=(xn+1)/2  #normalization bewteen 0,1
    return 1-xn #maximization to positive correlation
```

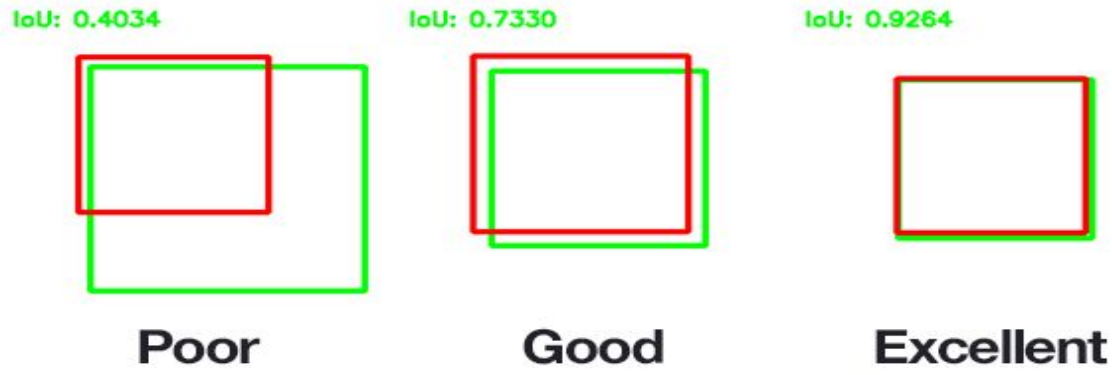
$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$



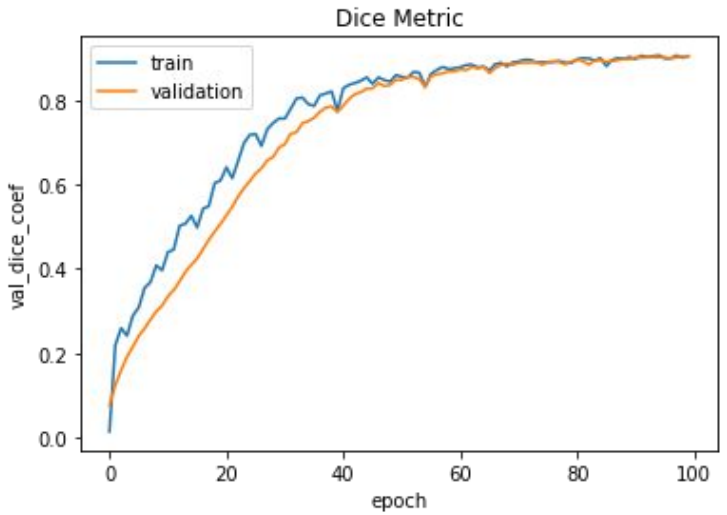
(fig 1)

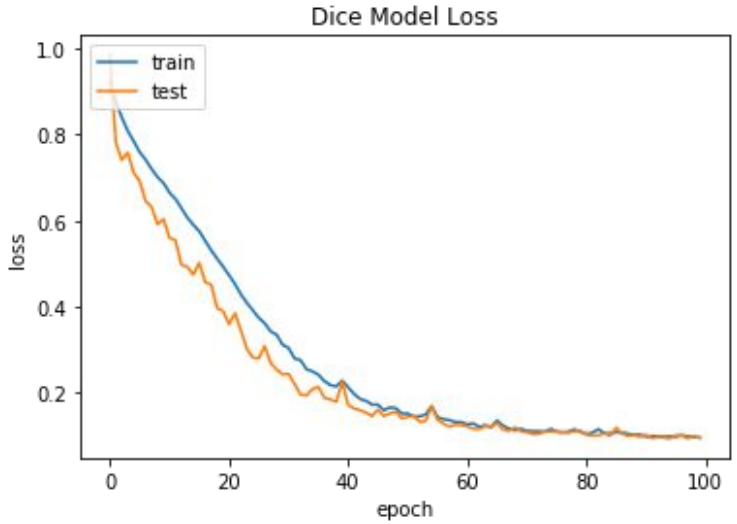


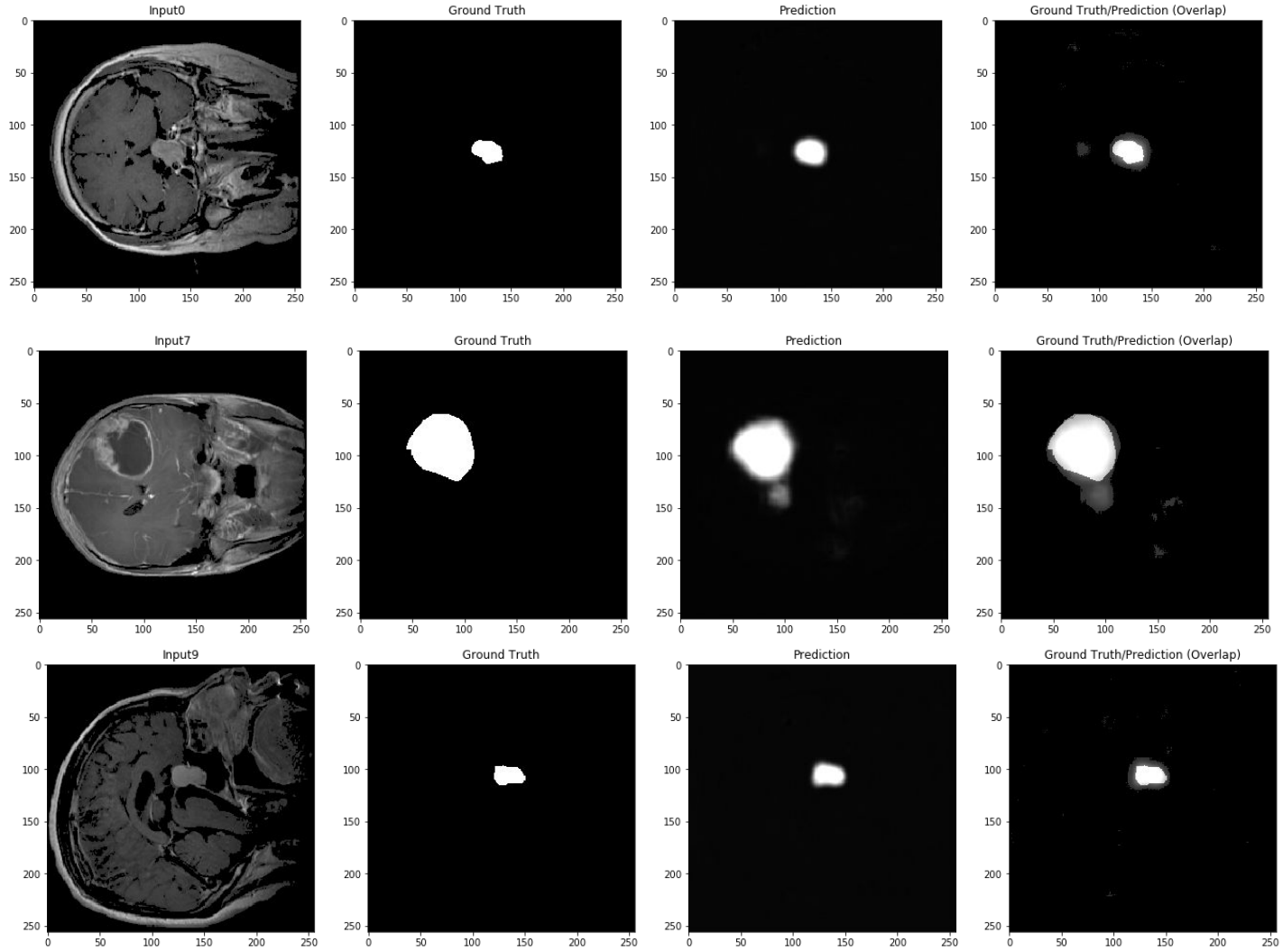
(fig 2)



Some 2D results

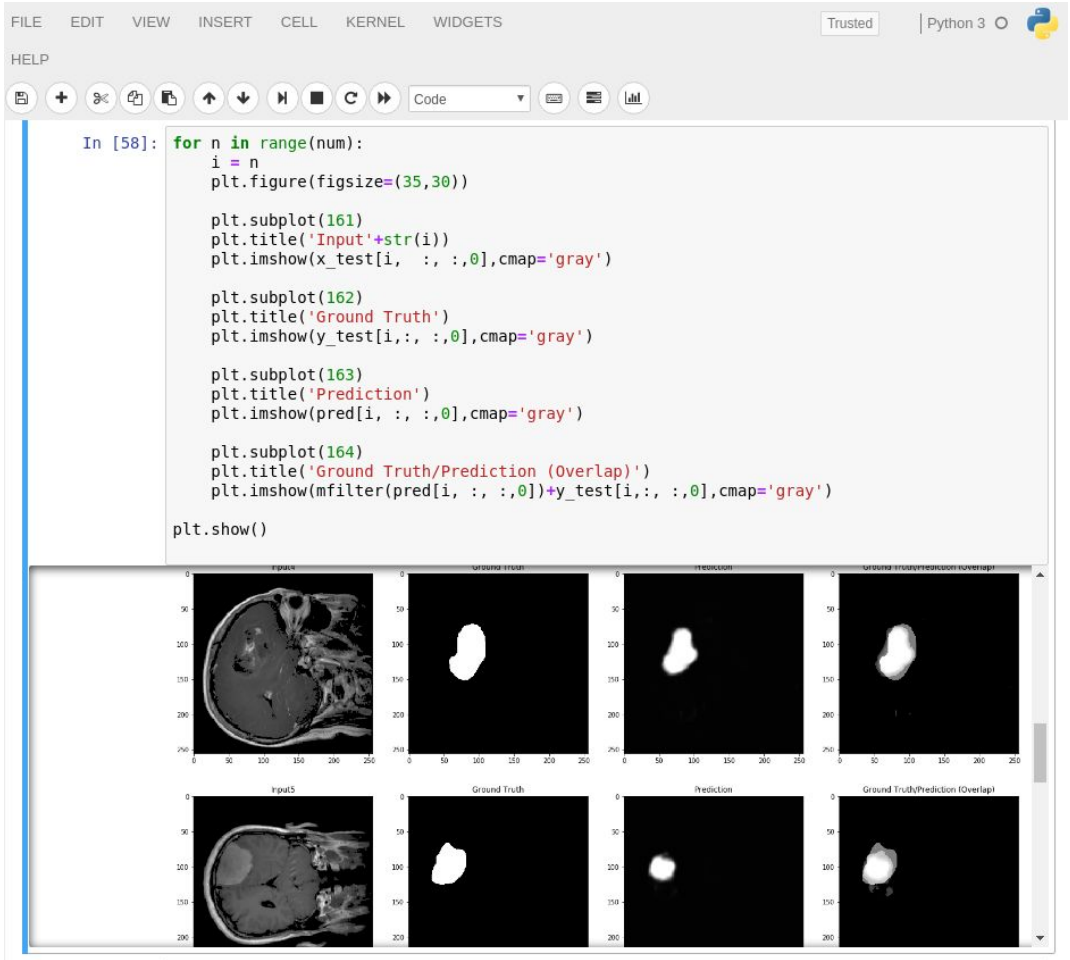






Using ScienceBox

- The medical application use case can be fully supported by ScienceBox
- We are using a ScienceBox deployment on a server equipped with a GPU
- Segmentation & classification of tumors
  - Notebook interface for iterative analysis
  - EOS/CERNBox to store the medical data
  - CVMFS to provide the Machine Learning libraries (tensorflow, keras, scikit-learn)
  - Massive performance gain by exploiting attached GPU



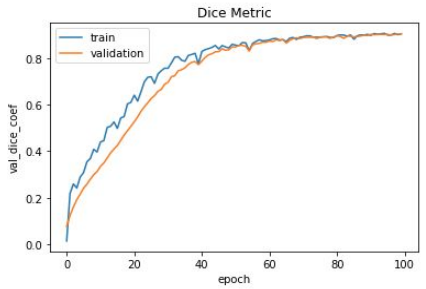
Using ScienceBox

```
Epoch 98/100
2438/2438 [==============================] - 182s 75ms/sample - loss: 0.0984 - dice_coef:
0.9016 - val_loss: 0.0936 - val_dice_coef: 0.9064
Epoch 99/100
2438/2438 [==============================] - 182s 75ms/sample - loss: 0.0966 - dice_coef:
0.9034 - val_loss: 0.0991 - val_dice_coef: 0.9010
Epoch 100/100
2438/2438 [==============================] - 182s 75ms/sample - loss: 0.0958 - dice_coef:
0.9042 - val_loss: 0.0952 - val_dice_coef: 0.9048
```

In [26]:
```
model.save('aug{}_{}_epoch{}'.format(num_of_aug,img_size,num_epoch))
model.save_weights('weights_{}_{}.h5'.format(img_size,num_epoch))
```

In [27]:
```
print(history.history.keys())
# summarize history for accuracy
plt.plot(history.history[list(history.history.keys())[3]])
plt.plot(history.history[list(history.history.keys())[1]])
plt.title('Dice Metric')
plt.ylabel(list(history.history.keys())[3])
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Dice Model Loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

```
dict_keys(['loss', 'dice_coef', 'val_loss', 'val_dice_coef', 'lr'])
```



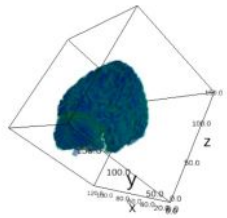

```
conv2d_55 (Conv2D)              (None, 256, 256, 64) 36928       activation_52[0][0]
batch_normalization_v1_53 (Batc (None, 256, 256, 64) 256         conv2d_55[0][0]
activation_53 (Activation)      (None, 256, 256, 64) 0           batch_normalization_v1_5
3[0][0]
conv2d_56 (Conv2D)              (None, 256, 256, 1)  65          activation_53[0][0]
==========
Total params: 34,535,745
Trainable params: 34,523,969
Non-trainable params: 11,776
```

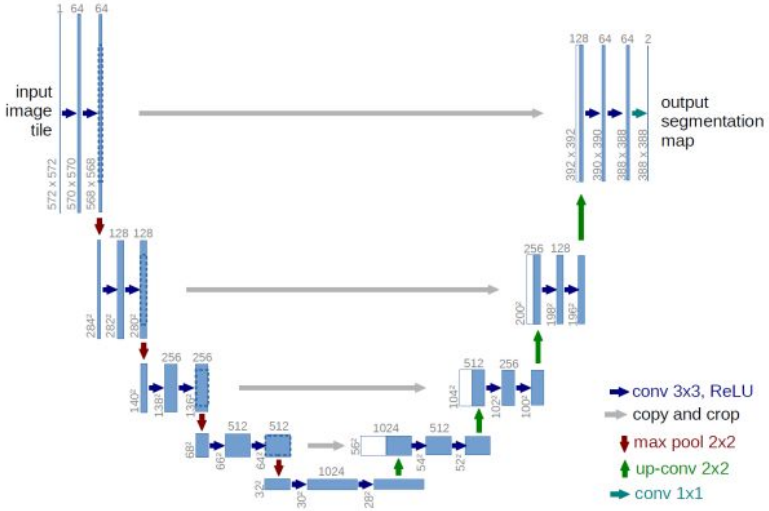
In [46]:
```
num_epoch = 100
batch_size = 16 # should be square

callbacks = [
    ReduceLROnPlateau(factor=0.01, patience=3, min_lr=0.00001, verbose=1)#,
]

history = model.fit_generator(train_datagen.flow(x_train, y_train, batch_size=batch_size), e
                              steps_per_epoch = (x_train.shape[0]//batch_size),
                              validation_data = val_datagen.flow(x_valid, y_valid, batch_size=ba
                              validation_steps = (x_valid.shape[0]//batch_size))#,
```

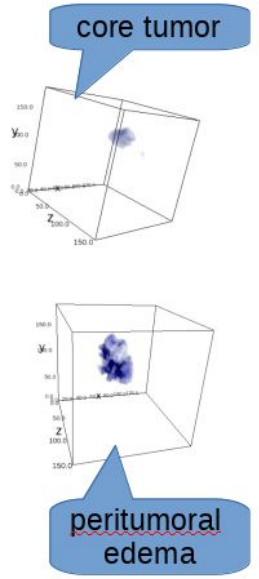
```
Epoch 1/1000
20/20 [==============================] - 7s 343ms/step - loss: 0.4724 - dice_coef: 0.0293
- mean_iou: 0.3969 - iuo: 0.0000e+00 - coc_metric: 0.0552
69/69 [==============================] - 67s 973ms/step - loss: 0.4461 - dice_coef: 0.038
6 - mean_iou: 0.3894 - iuo: 0.0000e+00 - coc_metric: 0.1078 - val_loss: 0.4724 - val_dice
_coef: 0.0293 - val_mean_iou: 0.3969 - val_iuo: 0.0000e+00 - val_coc_metric: 0.0552
Epoch 2/1000
```

3D segmentation

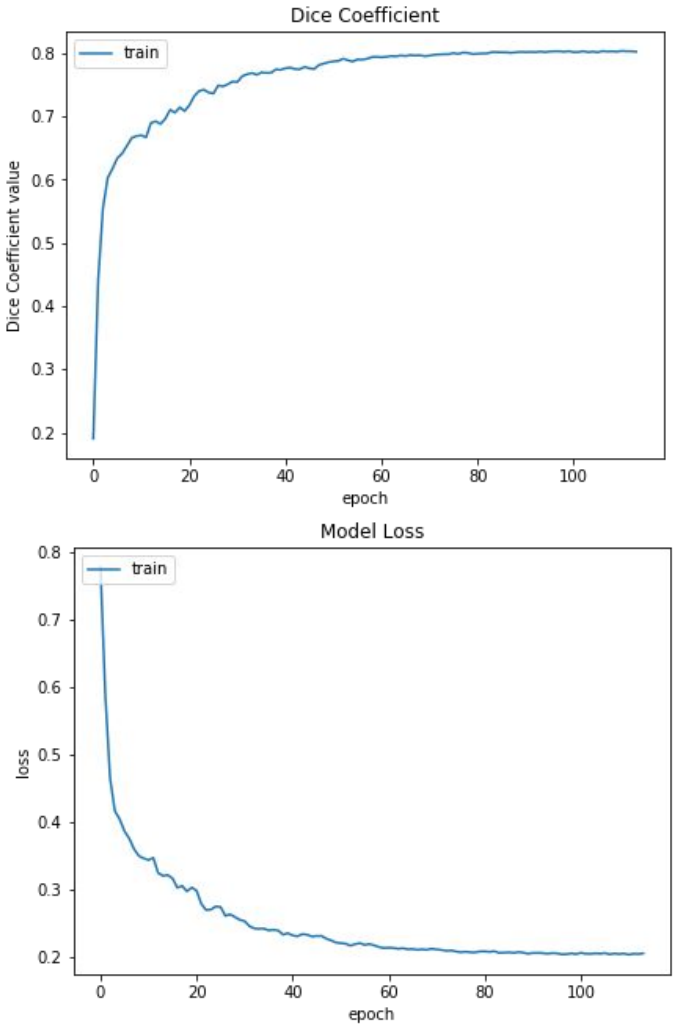
- Based on a new dataset provided by All-In-Image
- Better Quality (less noise)
- Skull Stripped (allows more accuracy)
- Multi-class segmentation (more complex)
  - Core tumor
  - Peritumoral edema
- Requires 3D convolutions instead 2D
  - More expensive in memory
  - Requires a lot of time to converge
- only 259 cubes of data for the training (vs 3000 images of the 2D dataset)

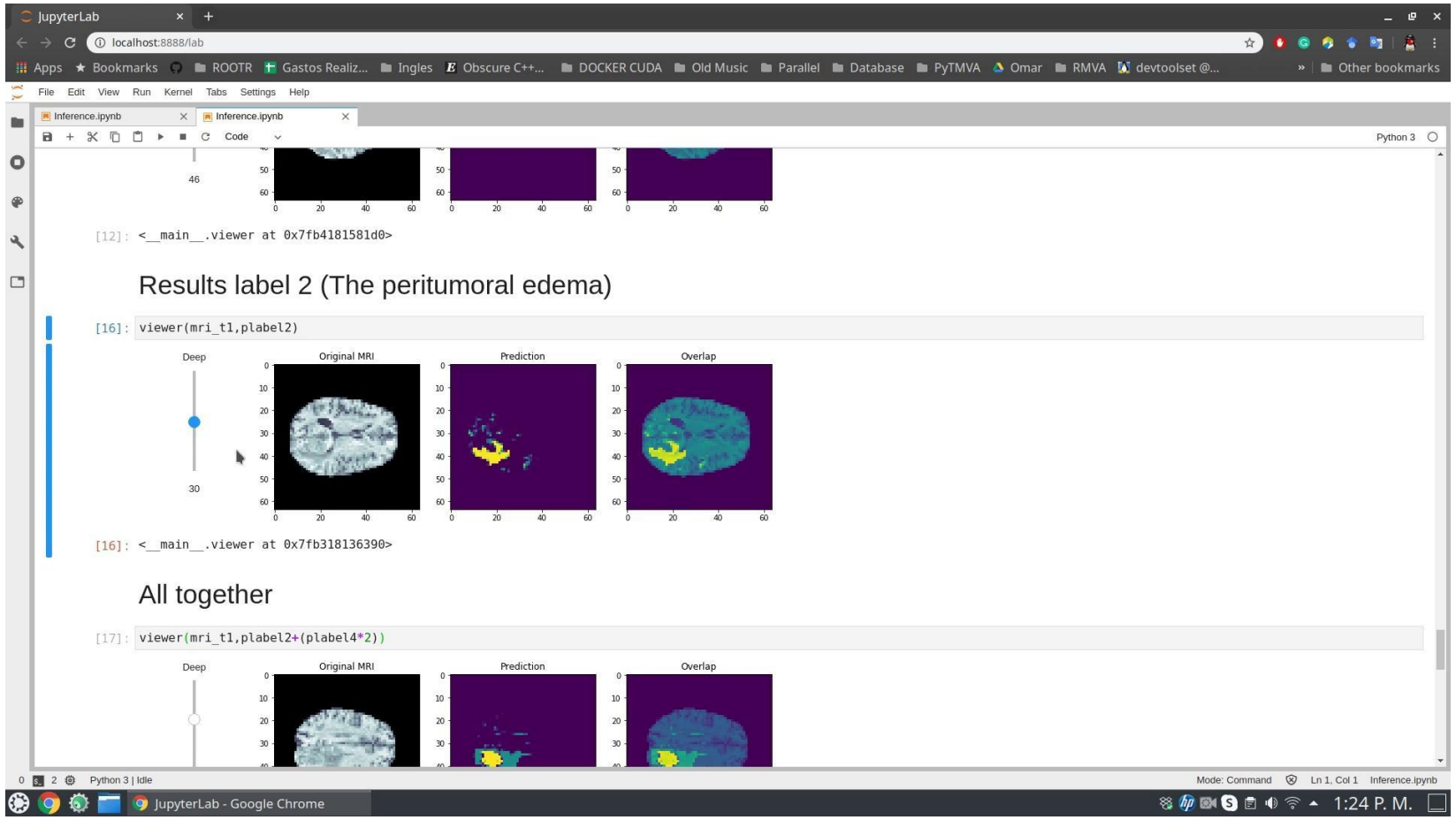






Results with Unet 3D





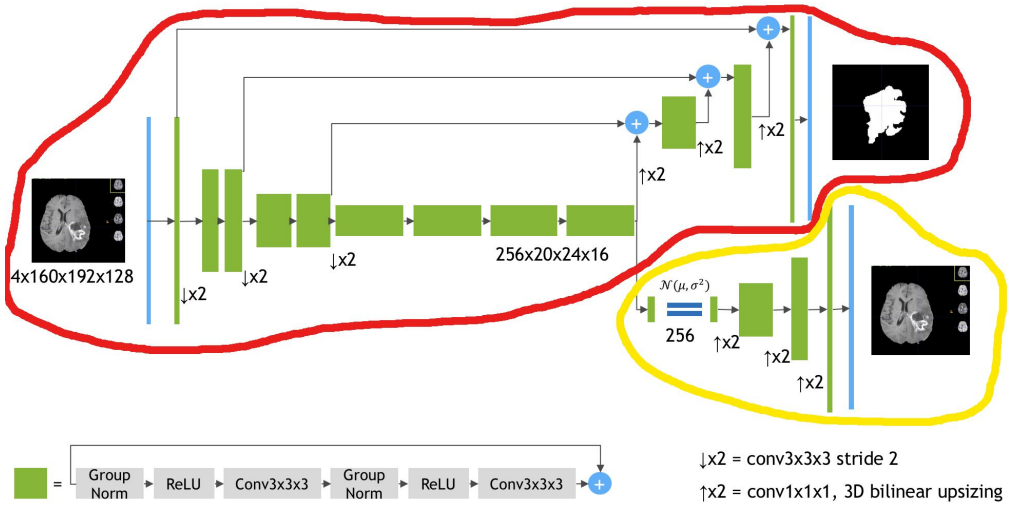
https://drive.google.com/file/d/1sOQDeiNeyVfhqAakBnaLqRVkWXoIxfk2/view?usp=sharing

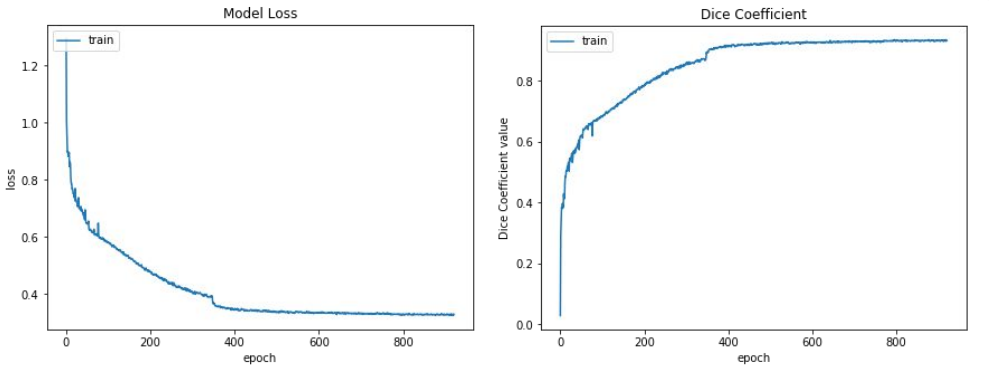
Sagittal View

Under R&D

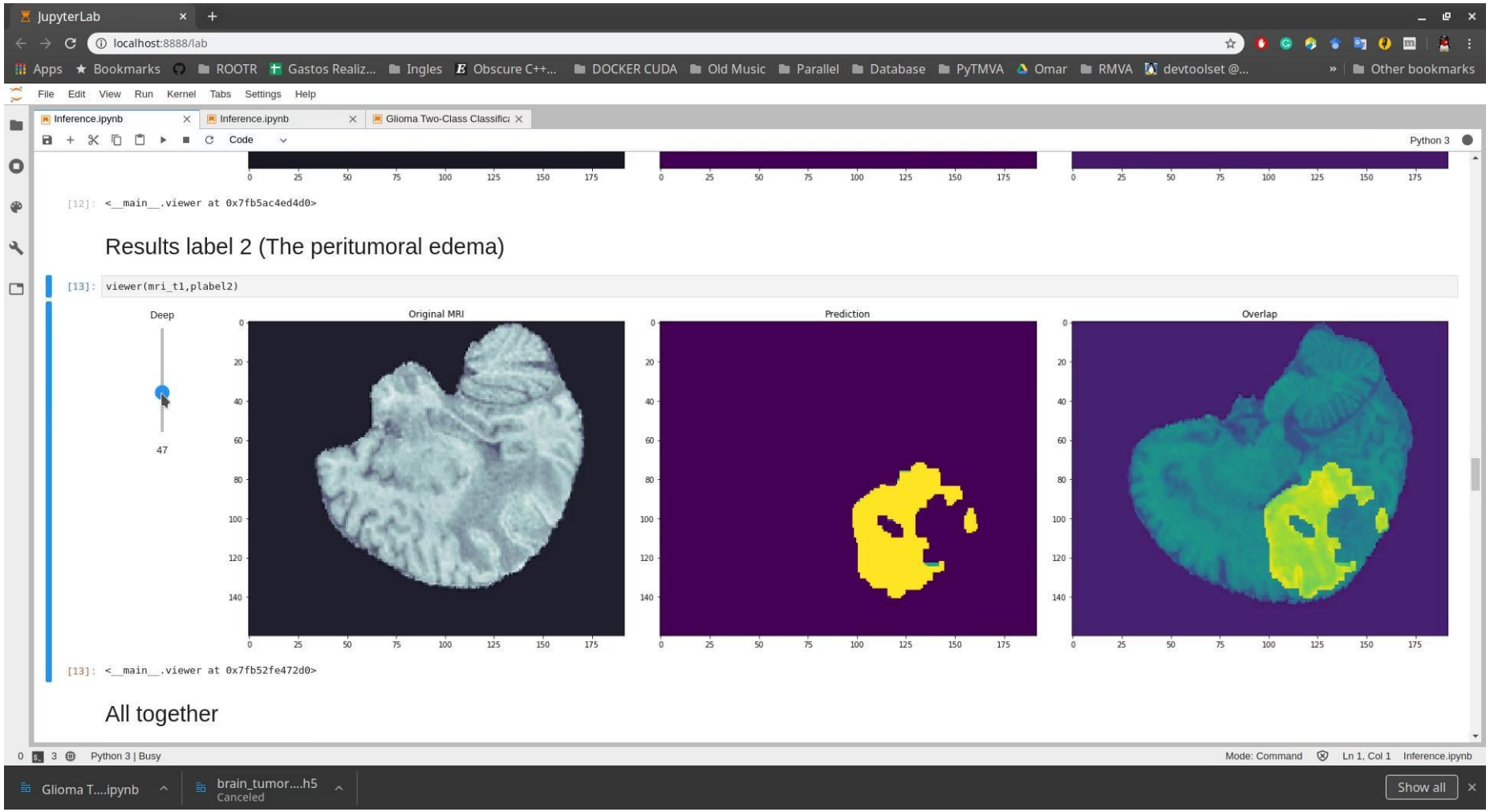
3D Unet-VAE

- Model based on this paper https://arxiv.org/pdf/1810.11654.pdf developed by NVidia
- Code taken from https://github.com/IAmSuyogJadhav/3d-mri-brain-tumor-segmentation-using-autoencoder-regularization
  - modified for:
    - our input data
    - to generate two masks (Core tumor, peritumoral edema)
- For this case the VAE is an structure that is encoding the normal distribution during the training to ensure the conservation of the properties of the data in the encoder.
- The VAE decoder is only a regularizer, it's not used to predict the mask.





3D preliminary results with Unet-VAE



https://drive.google.com/file/d/1JzDqAkhbI1DIU195u3usgXm1ivFPXqoh/view?usp=sharing

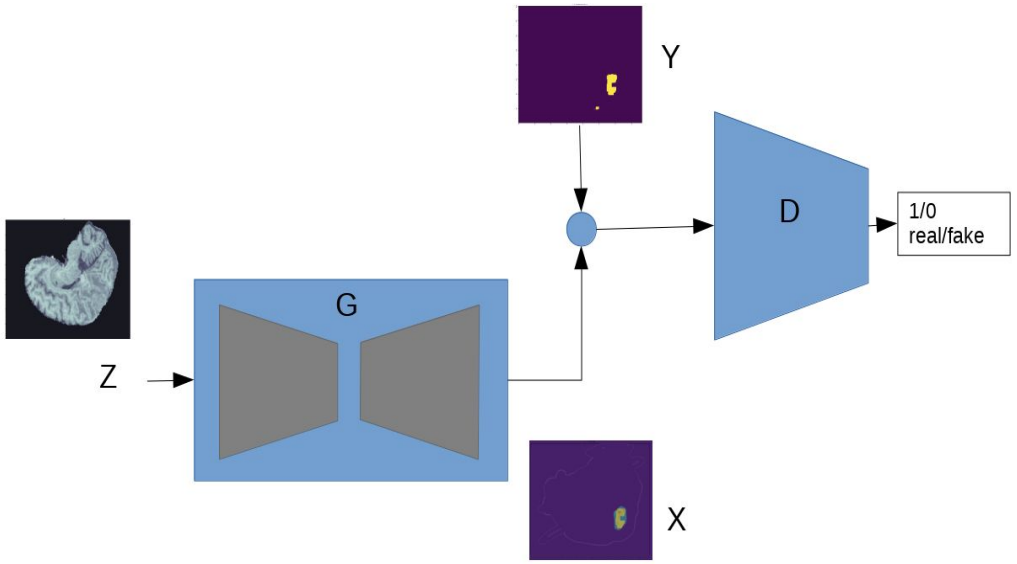
Coronal View

Generative Adversarial Networks (GANs)

- Two neural networks in a minmax game (Ian J. Goodfellow et al.)

$$\underset{G}{min}\,\underset{D}{max}V(D,G) = E_y Log(D(y)) + E_z Log(1 - D(G(z)))$$

- At the end the discriminator D is not able to say that X is false
- Z is not random noise is a MRI then this a cGAN (Conditional GAN)
- The mask is generated by G and the discriminator is not used in the inference.



Generative Adversarial Networks (GANs)

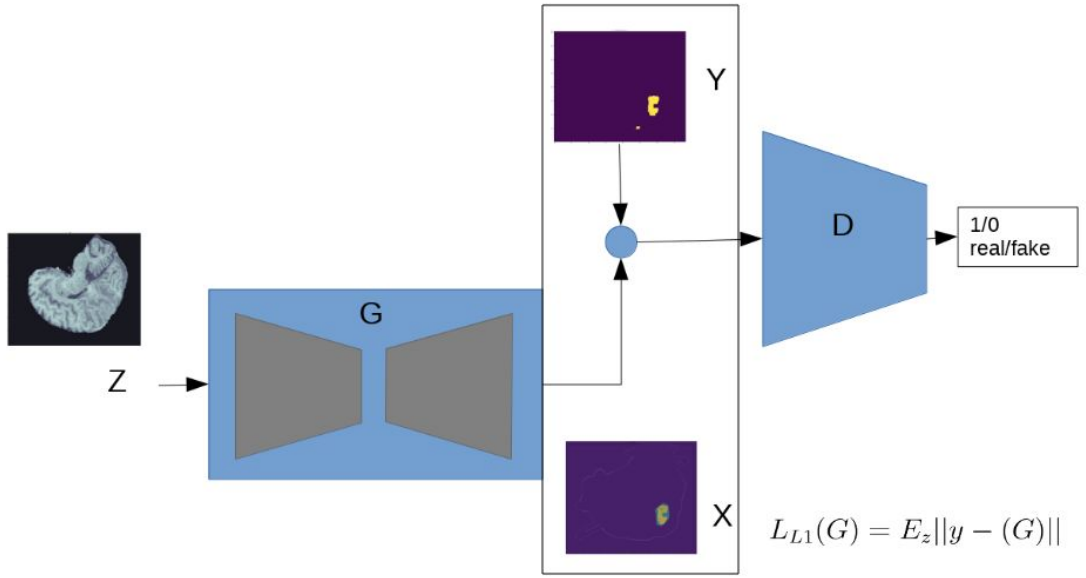
- I am working in a new model, inspired in the algorithm Image to Image translation, developed by [Berkeley AI Research, UC Berkeley]

$$\frac{min}{G}\frac{max}{D}V(D,G) = E_y Log(D(y)) + E_z Log(1 - D(G(z)))$$

- Requires a new regularizer
- The new loss is $L_{L1}(G) = E_z||y - (G)||$

$$G* = \frac{min}{G}\frac{max}{D}L_{cGAN}(D,G) + L_{L1}(G)$$

- What was implemented?
  - Generator is a 3D Unet
  - Discriminator is an Encoder with 3D convolutions
  - The regularizer is the dice loss.
  - Adapted to get the MRI and to generate the to masks



Generative Adversarial Networks (GANs)

- Not results YET!! The model is training at this moment and the dice coefficient is over 0.62
- This model is extremely expensive to train, I need to train two networks with 3D convolutions that requires a huge amount of memory and a lot of cuda cores.
- The model that I am currently training is for low resolution images and the number of filters in the convolutions were reduced. (This can affect the accuracy)

Conclusions

- Swan/ScienceBox now have support for GPUs
  - New stack for ML with GPU enabled packages
  - We can run machine learning models offloading the processing in a GPU
- Machine learning models for medical applications have been developed
  - For glioma classification in 2D images
  - For 2D and 3D segmentation
- This research would be useful for hospitals and medical centers to study the patients with brain tumors and it can may help to follow the progress in the treatment.
- The models and the techniques can be extended to do segmentation for other regions in the body.

Thank you

References

- MRI Images from https://figshare.com/articles/brain_tumor_dataset/1512427/5
- MRI Scanner image taken from https://nationalmaglab.org/education/magnet-academy/learn-the-basics/stories/mri-a-guided-tour
- https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf
- https://arxiv.org/abs/1611.07004
- https://www.mdpi.com/2076-3417/8/1/27