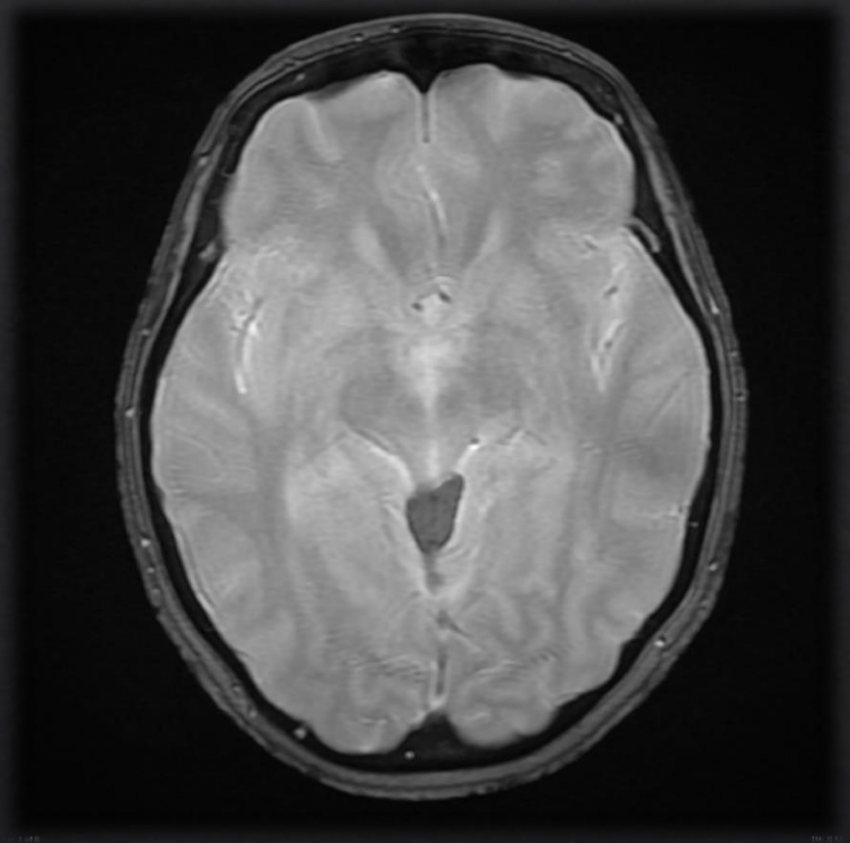


BrainScan'19

David García, Aitor Jara, Eduard Mainou and Núria Sánchez
Advisor: Santi Puch

Artificial Intelligence with Deep Learning
UPC School
July 2019

Contents

- Setting the task
- Preprocessing
- Implementation
  - Dataset
  - Train.py
  - Test.py
- Experiments
- Demo
- Q&A

Setting the task

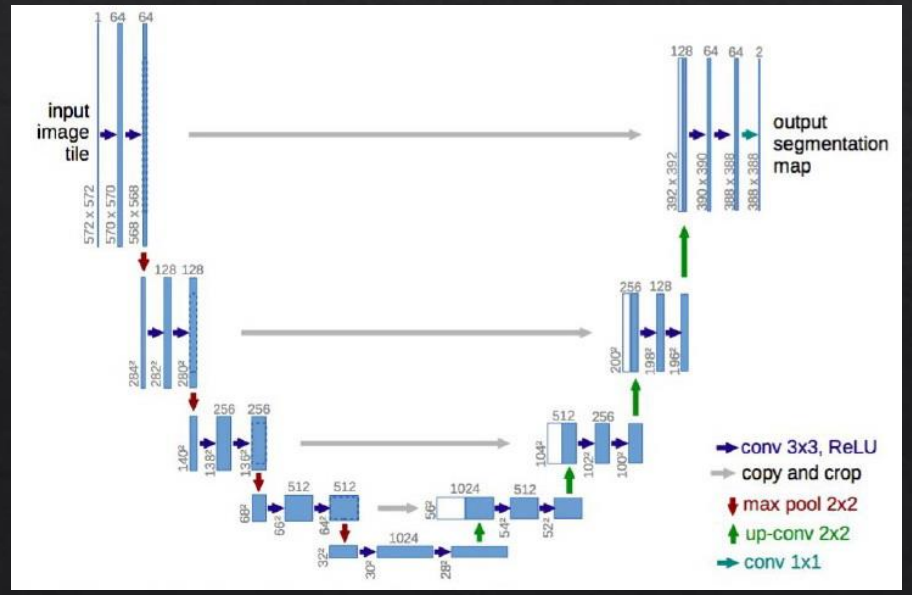
- **GOAL**: Segmentation of images displaying a brain with a tumour.

Setting the task

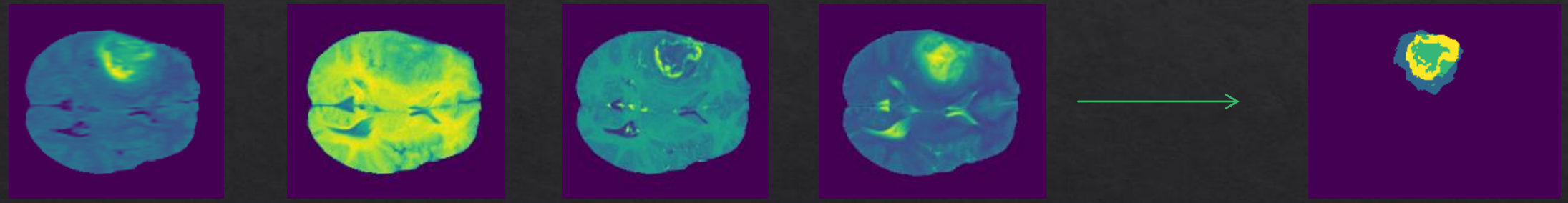
- **GOAL**: Segmentation of images displaying a brain with a tumour.
- **DATASET**: BRATS challenge
  - 484 Training Patients & 266 Testing Patients
  - Information stored in a single patient: Four 3D images + One 3D image (ground truth)
  - Segmentation: Voxel-level classification in the set $\{0, 1, 2, 3\}$.

Setting the task

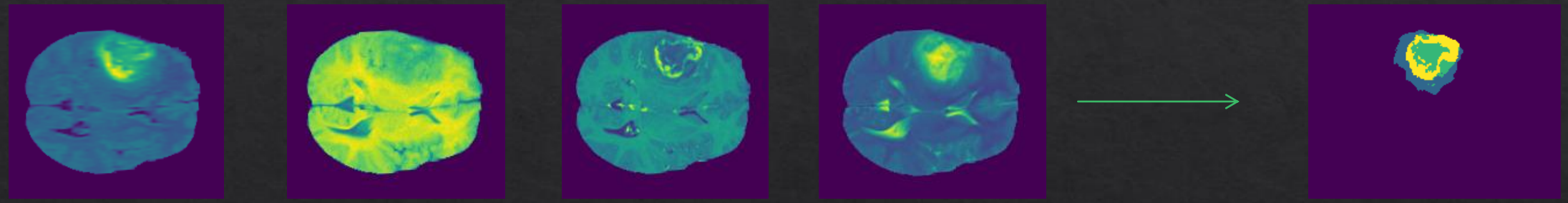
- **GOAL**: Segmentation of images displaying a brain with a tumour.
- **DATASET**: BRATS challenge
  - 484 Training Patients & 266 Testing Patients
  - Information stored in a single patient: Four 3D images + One 3D image (ground truth)
  - Segmentation: Voxel-level classification in the set {0, 1, 2, 3}.
- **ARCHITECTURE:** U-Net



- **INPUT**
  - Slice by slice comparison: blocks with 4 channels
  - Example: 4th Patient → slices 77th



- **INPUT**
  - Slice by slice comparison: blocks with 4 channels
  - Example: 4th Patient → slices 77th



- **OUTPUT**:
  - Four classes: {0, 1, 2, 3}
  - Binarizing the labels: non-tumour / tumour

Preprocessing

- Trainig: 484 Volumes * 155 slices * (4 Images + label) = 375.100 Images
- Test: 266 Volumes * 155 slices * 4 images = 164.920 Images
- > 500.000 Images
- Solution: TFRecords
  - Is a simple format for storing a sequence of binary records.
  - You need specify the structure of your data before you write it to the file
  - Advantages:
    - Save memory
    - Faster Reading
    - TF Native
    - Multi-platform

Preprocessing

- TFRecordWriter to write it to disk
  - writer = tf.python_io.TFRecordWriter(outputFile)

- Example is a data structure for representing a record
  - example = tf.train.Example(features = tf.train.Features(feature = {

```
'image': _bytes_feature(frames.tostring()),
'label': _bytes_feature(label.tostring()),
'PatientID': _int64_feature(int(self.get_patient_id(inputFileVolume))),
'Slide': _int64_feature(i),
'height': _int64_feature(self.HEIGHT),
'width': _int64_feature(self.WIDTH),
'depth': _int64_feature(self.DEPTH),
 }))
writer.write(example.SerializeToString())
```

Preprocessing

- Preprocessing considerations:
  - Images are joined creating a 240x240x4 vector
  - 20% of volumes for validation
  - Exclusion of empty images
  - 50% Images with tumor and 50% Images without tumor
  - Added aditional information (PatientID, Slide No., height, width, depth)

Implementation

- Dataset
  - **TFRecordDataset**
    - dataset = tf.data.TFRecordDataset(filenames=filenames)
  - **map_func** determine the structure of each element in the returned dataset.
    - dataset = dataset.map(_parse_function)
  - Data augmentation and options
    - Central crop
    - Optional Binarize
    - Optional One Hot
  - Other parameters
    - Perform shuffle
    - Batch size

Implementation

Train.py Trains the model

Main.py

Test.py Inference purposes

Implementation

Train.py Trains the model

Main.py

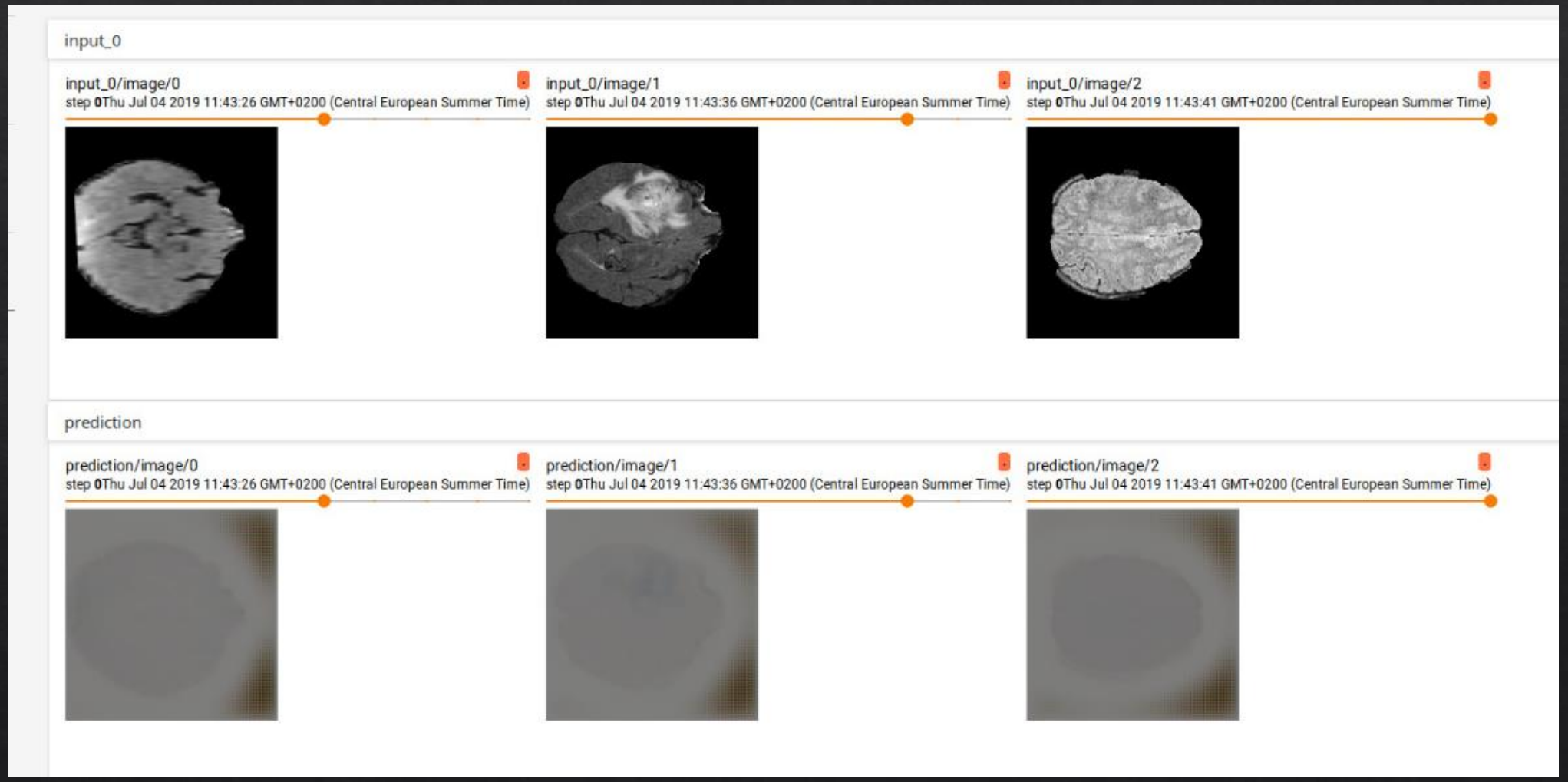
Test.py Inference purposes

```
print("Are we testing or training? True for training, False for testing")
training = input('Enter your input:')
print(type(training))
if training == "True" :
    train.main(args.trainingdir, args.model, args.num_epochs, args.size_batch_train, args.size_batch_valid,
    args.step_metrics, args.steps_saver, args.learning_rate, args.logdir,
    args.restore_weights,args.perform_one_hot,args.binarize_labels)
else:
    test.main(args.trainingdir, args.model, args.num_epochs, args.size_batch_test, args.logdir, args.logdir_w,
    args.perform_one_hot, args.binarize_labels)
```

Train.py

```
import tensorflow as tf
def unet(inputs, training=False,label_output_size=1):
    “All layers”
    conv10 = tf.keras.layers.Conv2D(label_output_size, 1)(conv9)
    if label_output_size==1:
        conv10_soft = tf.keras.activations.sigmoid(conv10)
    else:
        conv10_soft = tf.keras.activations.softmax(conv10)
    return conv10, conv10_soft
```

Train.py


input_0
input_0/image/0
step 0Thu Jul 04 2019 11:43:26 GMT+0200 (Central European Summer Time)
input_0/image/1
step 0Thu Jul 04 2019 11:43:36 GMT+0200 (Central European Summer Time)
input_0/image/2
step 0Thu Jul 04 2019 11:43:41 GMT+0200 (Central European Summer Time)
prediction
prediction/image/0
step 0Thu Jul 04 2019 11:43:26 GMT+0200 (Central European Summer Time)
prediction/image/1
step 0Thu Jul 04 2019 11:43:36 GMT+0200 (Central European Summer Time)
prediction/image/2
step 0Thu Jul 04 2019 11:43:41 GMT+0200 (Central European Summer Time)

Train.py

One Hot	BinaryLabels	Loss	Layer Input Size	Layer Output Size
True	True	softmax_cross_entropy	2	2
True	False	softmax_cross_entropy	4	4
False	True	softmax_cross_entropy	1	1
False	False	sparse_softmax_cross_entropy	1	4

Train.py

```
handle = tf.placeholder(tf.string, shape = [])

iterator = tf.data.Iterator.from_string_handle(handle, train_dataset.output_types,
train_dataset.output_shapes)
```

Train.py

```
handle = tf.placeholder(tf.string, shape = [])

iterator = tf.data.Iterator.from_string_handle(handle, train_dataset.output_types,
train_dataset.output_shapes)
```

```
train_handle = sess.run(train_iterator.string_handle())

validation_handle = sess.run(validation_iterator.string_handle())
```

Train.py

```
handle = tf.placeholder(tf.string, shape = [])

iterator = tf.data.Iterator.from_string_handle(handle, train_dataset.output_types,
train_dataset.output_shapes)
```

```
train_handle = sess.run(train_iterator.string_handle())

validation_handle = sess.run(validation_iterator.string_handle())
```

```
_,cost,summary_val,step_gl,logits_val,_ = sess.run([train_op,loss_op,summary_op,global_step,logits,logits_soft],
feed_dict={handle: train_handle,training_placeholder: True})

cost_valid, _ = sess.run([loss_op, IoU_metrics_update], feed_dict={handle: validation_handle,training_placeholder:
False})
```

Test.py

```
test_dataset = create_dataset(filenames=test_list,mode="testing", num_epochs=1,
batch_size=size_batch_test,perform_one_hot=perform_one_hot,binarize_labels=binarize_labels)

test_iterator = test_dataset.make_initializable_iterator()
```

Test.py

```
test_dataset = create_dataset(filenames=test_list,mode="testing", num_epochs=1,
batch_size=size_batch_test,perform_one_hot=perform_one_hot,binarize_labels=binarize_labels)

test_iterator = test_dataset.make_initializable_iterator()
```

```
handle = tf.placeholder(tf.string, shape = [])

iterator = tf.data.Iterator.from_string_handle(handle, test_dataset.output_types,
test_dataset.output_shapes)

    x, _ = iterator.get_next()
```

Test.py

```
test_dataset = create_dataset(filenames=test_list,mode="testing", num_epochs=1,
batch_size=size_batch_test,perform_one_hot=perform_one_hot,binarize_labels=binarize_labels)

test_iterator = test_dataset.make_initializable_iterator()
```

```
handle = tf.placeholder(tf.string, shape = [])

iterator = tf.data.Iterator.from_string_handle(handle, test_dataset.output_types,
test_dataset.output_shapes)

    x, _ = iterator.get_next()
```

```
saver.restore(sess, tf.train.latest_checkpoint(logdir))

test_handle = sess.run(test_iterator.string_handle())

sess.run(test_iterator.initializer)
```

Test.py

```
try:
    while True:
        summary_val,logits_test =
        sess.run([summary_test,logits_soft],feed_dict={handle: test_handle,
        training_placeholder:False})


        writer.add_summary(summary_val)


except tf.errors.OutOfRangeError:
pass
```

Experiments

- Diferent ways of computing the loss based on different approaches:
  - Shape of X **(Input)** = [25, 192, 192, 4] → [Batch, Height, Width, Channels]
  - Shape of Y **(Label) =** [25, 192, 192, 1] → 25 images of 192 *192 with pix.values {0,1,2,3}
  - Shape of A **(Pred)** = [25,192,192,4] →

0.1	0.2	…. n x m (192 * 192 x 25)	→
0.2	0.6	……	→
0.0	0.1	……	→
0.7	0.1	……	→
1	1		

All the pixels from the same 25 images flatten out with the probability of belonging to each class

PROBLEM?

FORGOT TO APPLY SOFTMAX, PREDICTIONS ARE BAD

Softmax for 4 clases at pixel level

Experiments

- Diferent ways of computing the loss based on different approaches:
  - Shape of X **(Input)** = [25, 192, 192, 4] → [Batch, Height, Width, Channels]
  - Shape of Y **(Label) =** [25, 192, 192, 1] → 25 images of 192 *192 with pix.values {0,1,2,3}
  - Shape of A **(Pred)** = [25,192,192,1] → 0.8 0.2 0.1 …. n x m (192 * 192 x 25)

25 images flatten out in a neurone with the probability of belonging to the first class and 1-P of belonging to the second class

Sigmoid for 2 classes at pixel level

SOLUTION

SIMPLIFY PROBLEM BY BINARIZING THE PREDICTIONS: TUMOUR/NON-TUMOUR

Experiments

- Diferent ways of computing the loss based on different approaches:
  - Shape of Y **(Label) =** [25, 192, 192, 1] → **1 channel represents 4 classes**

    - 1 3 2 0 2 …. n x m (192 * 192 x 25)

PROBLEM?

CAN'T USE IoU METRICS, ONLY
LEARNING CURVES TO DIAGNOSE

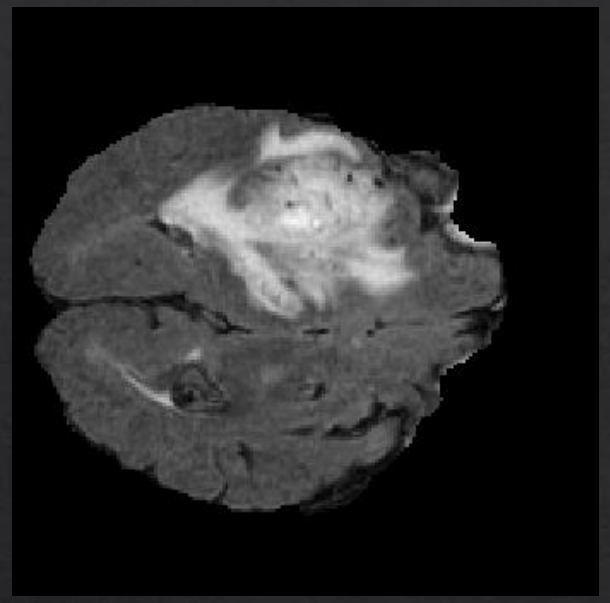
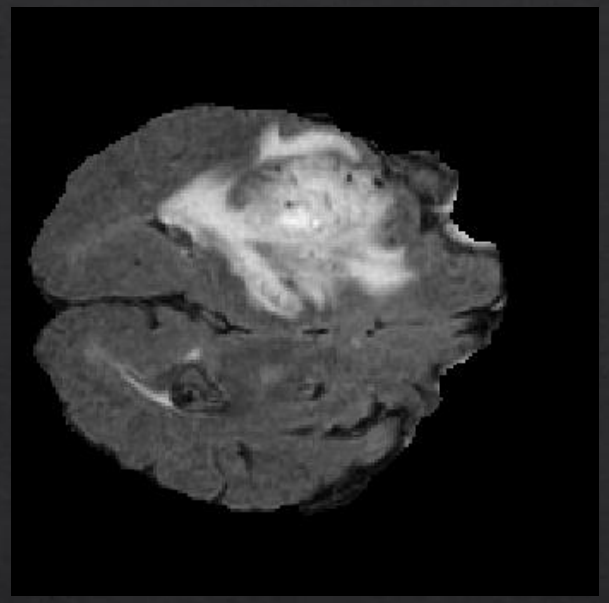
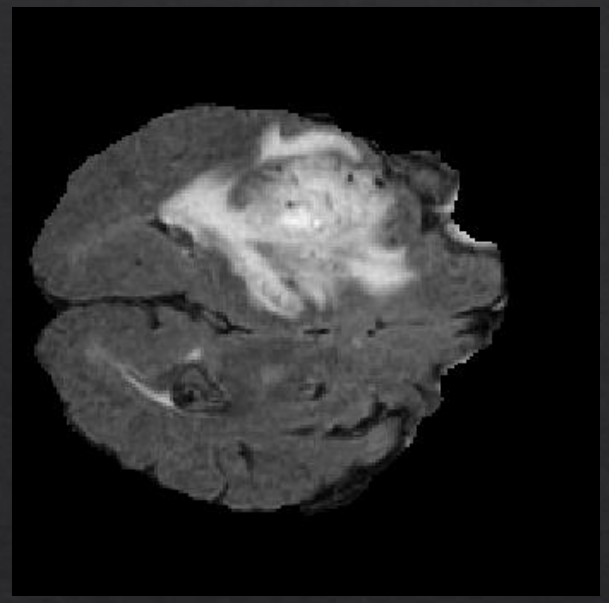
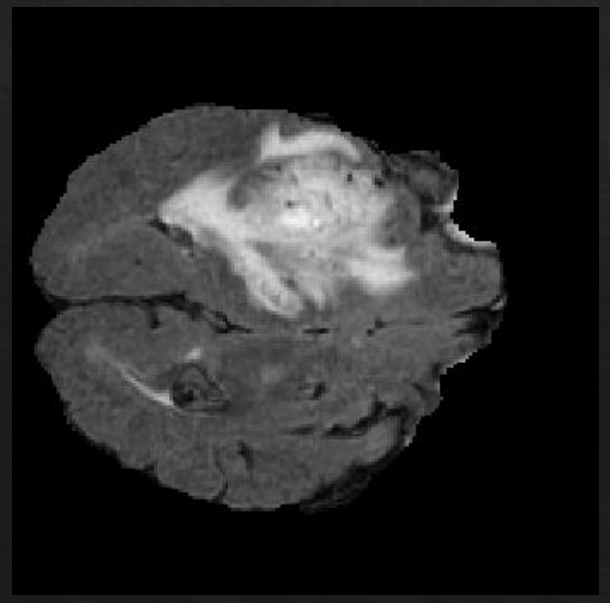
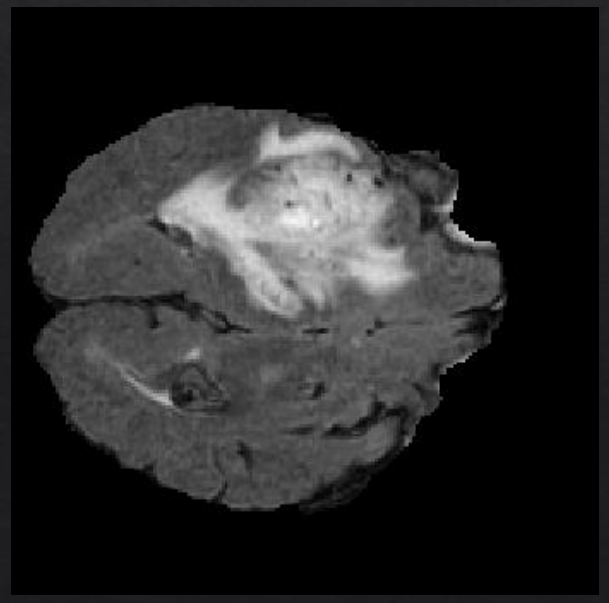
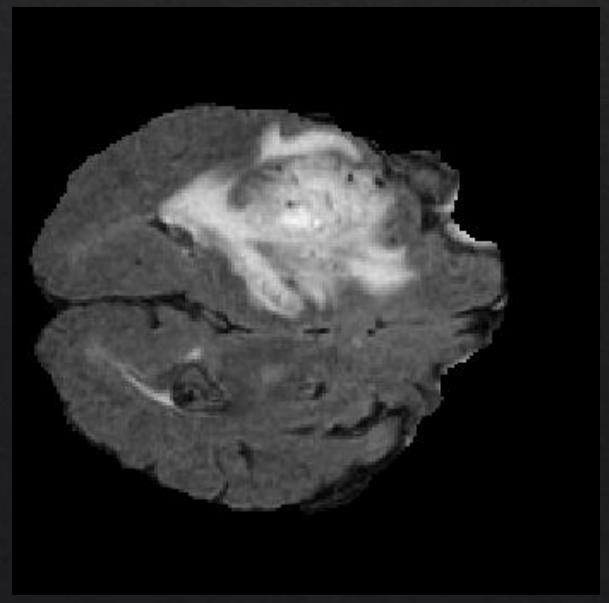
Experiments

- Diferent ways of computing the loss based on different approaches:
  - Shape of Y **(Label) =** [25, 192, 192, 4] → one-hot matrix

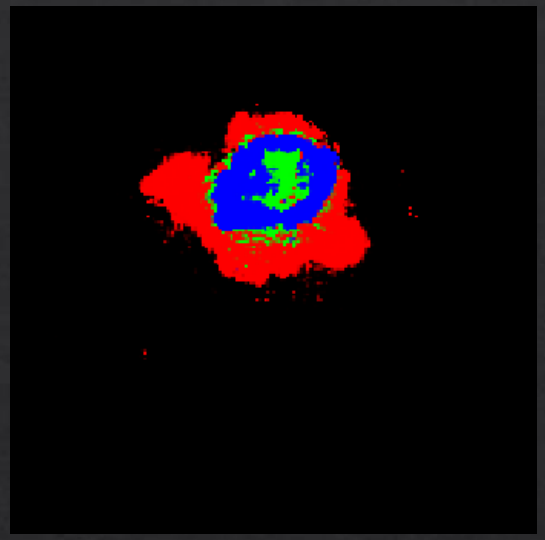
0	0	0	1	0 …. n x m (192 * 192 x 25)
1	0	0	0	0 …. n x m (192 * 192 x 25)
0	0	1	0	1 …. n x m (192 * 192 x 25)
0	1	0	0	0 …. n x m (192 * 192 x 25)

SOLUTION

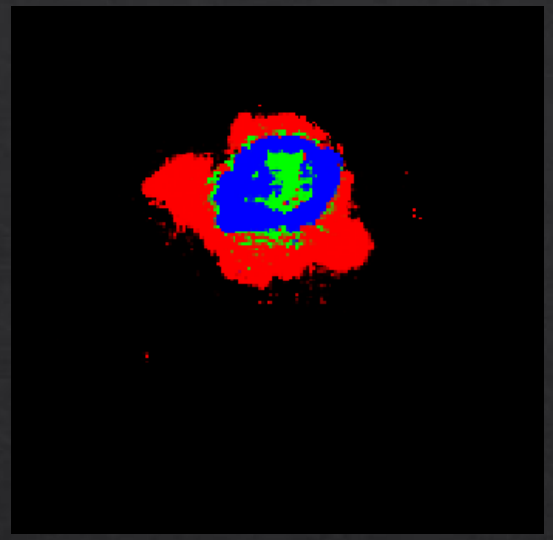
CONVERT TO ONE-HOT ENCODING

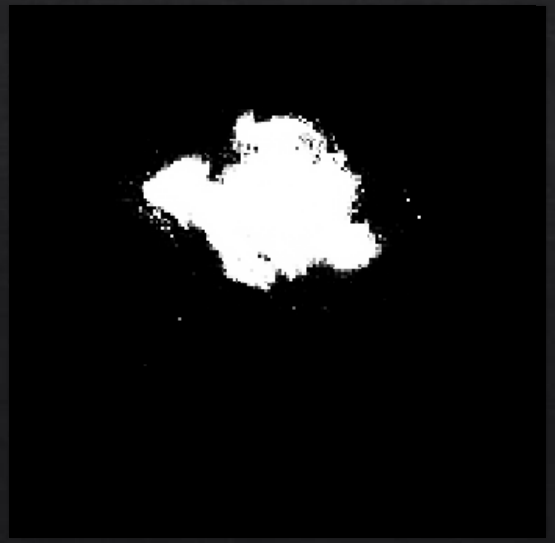
Keras no-one-hot + 2 classes



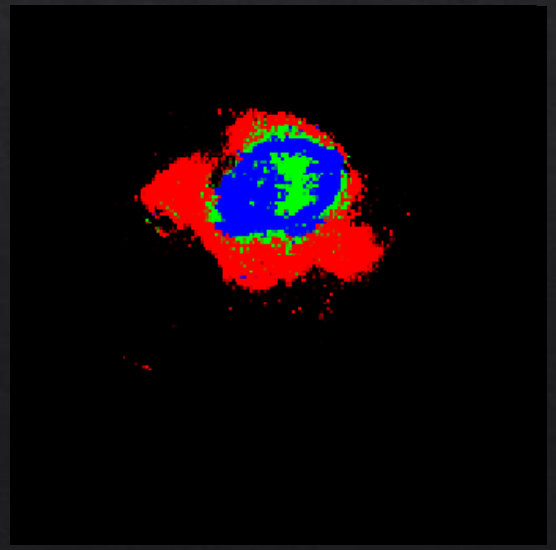
Keras no-one-hot + 4 classes



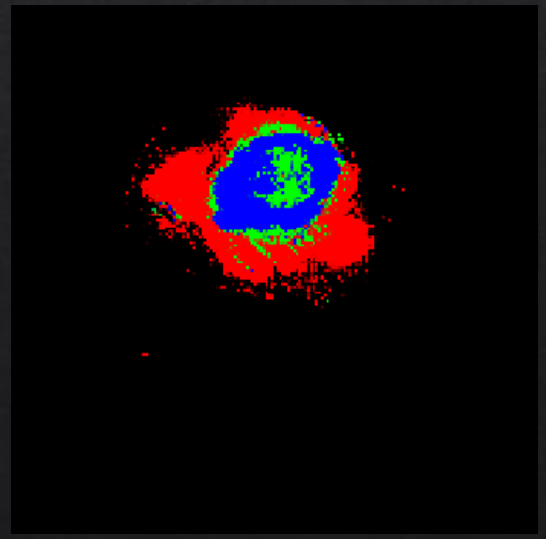
Keras one-hot + 4 classes



Tensorflow no-one-hot + 2 classes

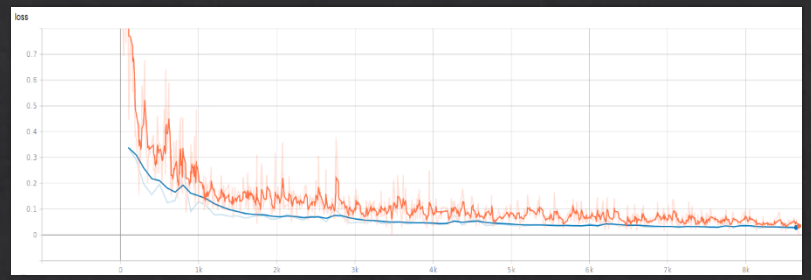


Tensorflow no-one-hot + 4 classes

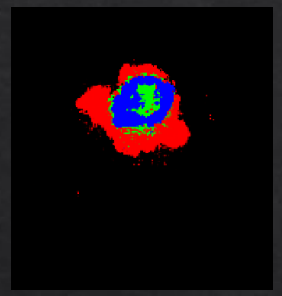


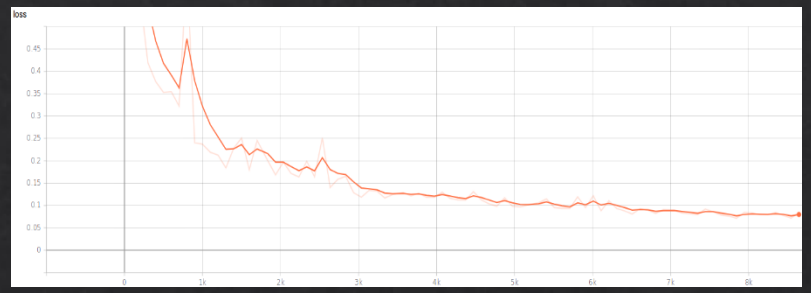
Tensorflow one-hot + 4 classes

Keras

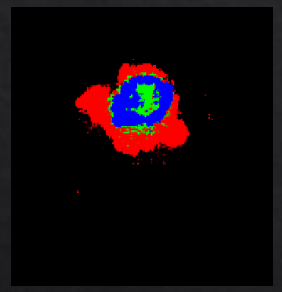


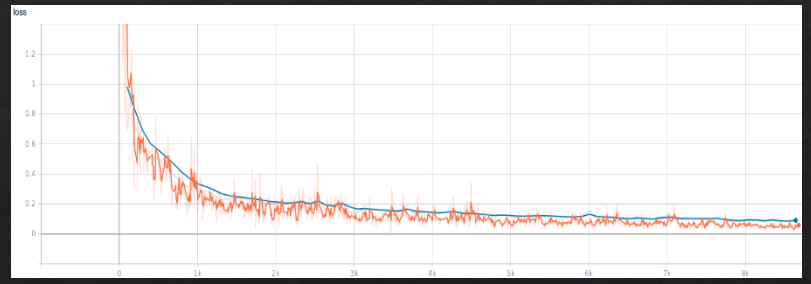
No-one-hot 2 class





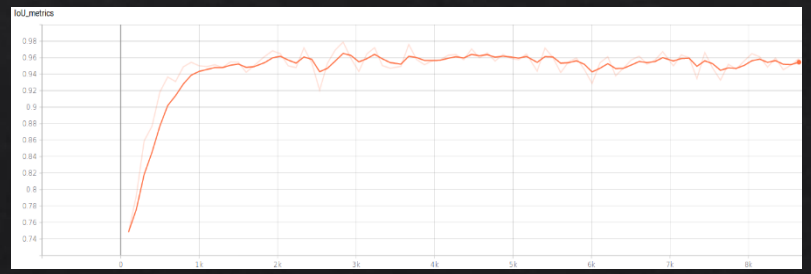
No-one-hot 4 class



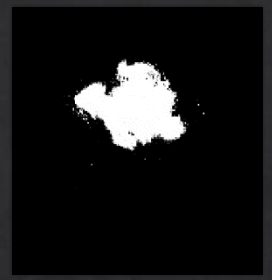


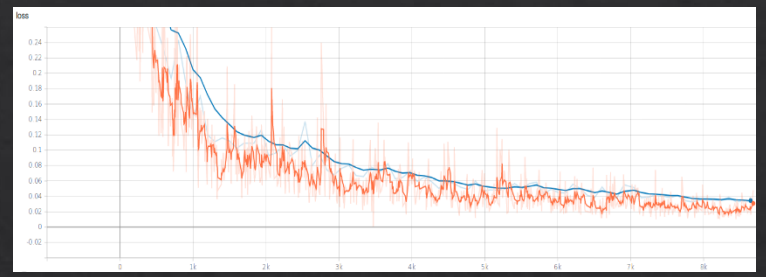
One-hot 4 class

IoU metric

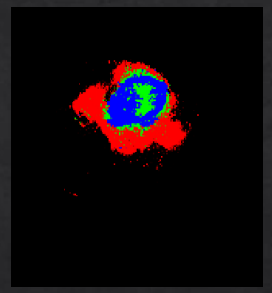


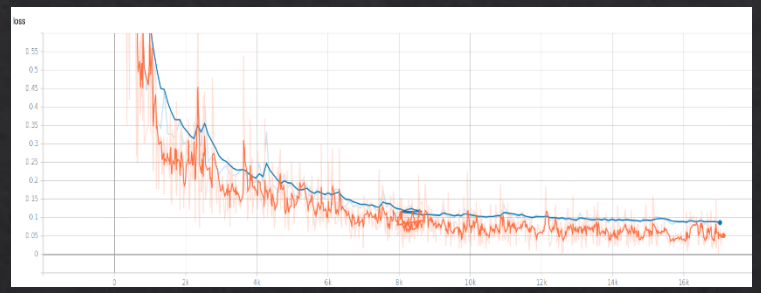
Tensorflow



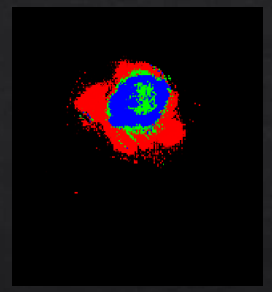


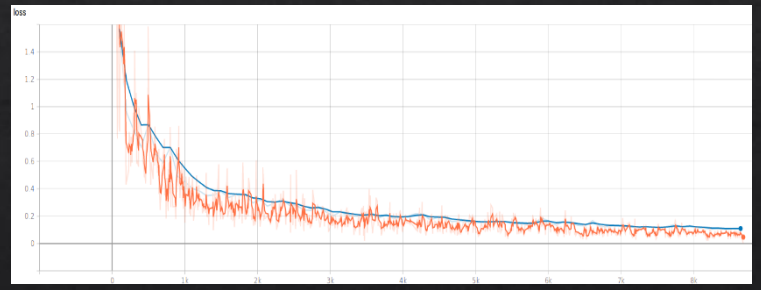
No-one-hot 2 class





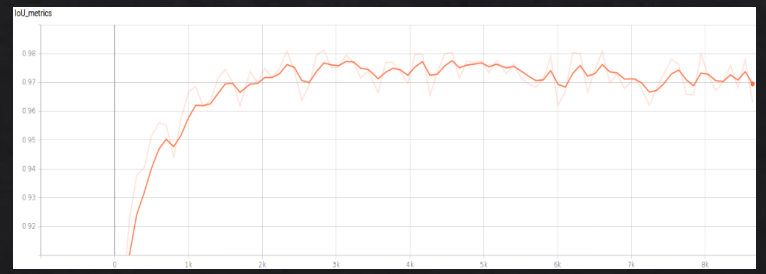
No-one-hot 4 class





One-hot 4 class

IoU metric



Q&A

¿?