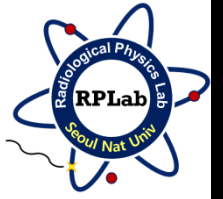
TF 스터디 발표



ALEXNET REVIEW

2016. 07. 18.

JIMIN LEE

CONTENTS

1. AlexNet

2. TensorFlow Code

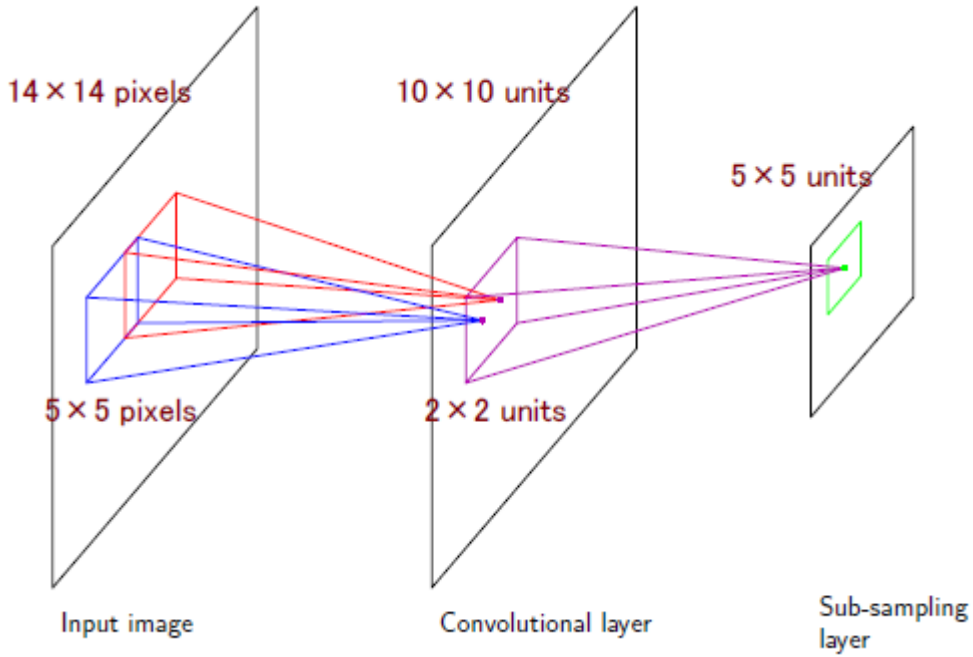
3. Function Code

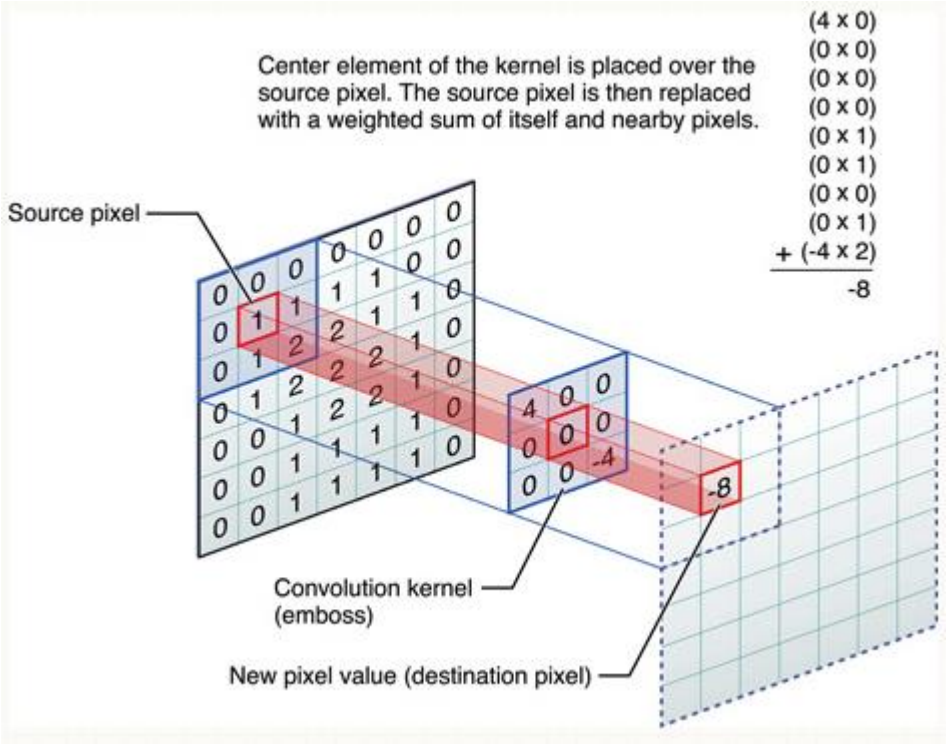
4. Related Paper Review

CNN

- **Convolutional Neural Networks (CNN)**
  - A very powerful algorithm for image recognition
  - Convolutional layer : Convolved feature made by convolutional kernel (filter)

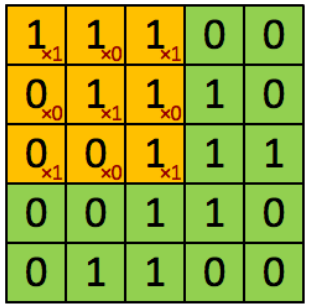
➡ **Hidden layer in NN (Neural Network)**



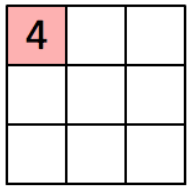


CNN

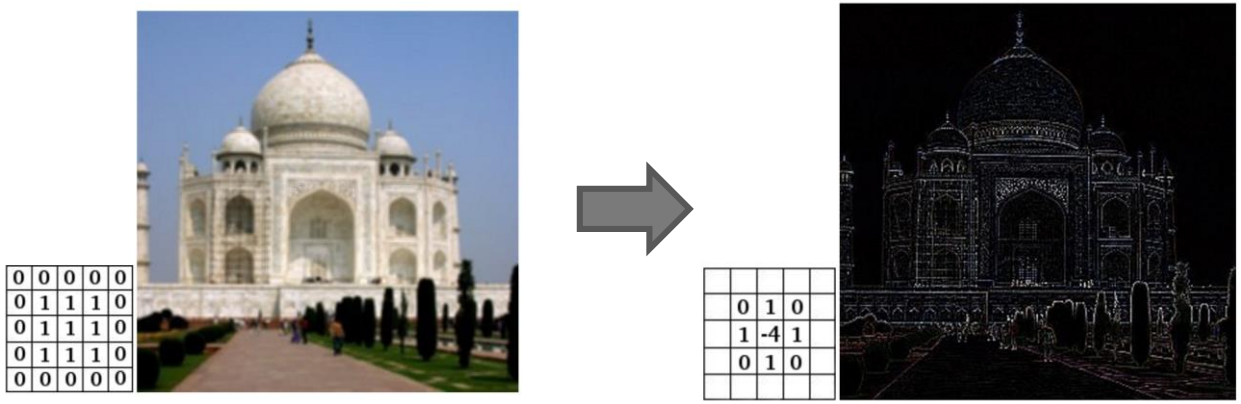
- **Convolutional Neural Networks (CNN)**

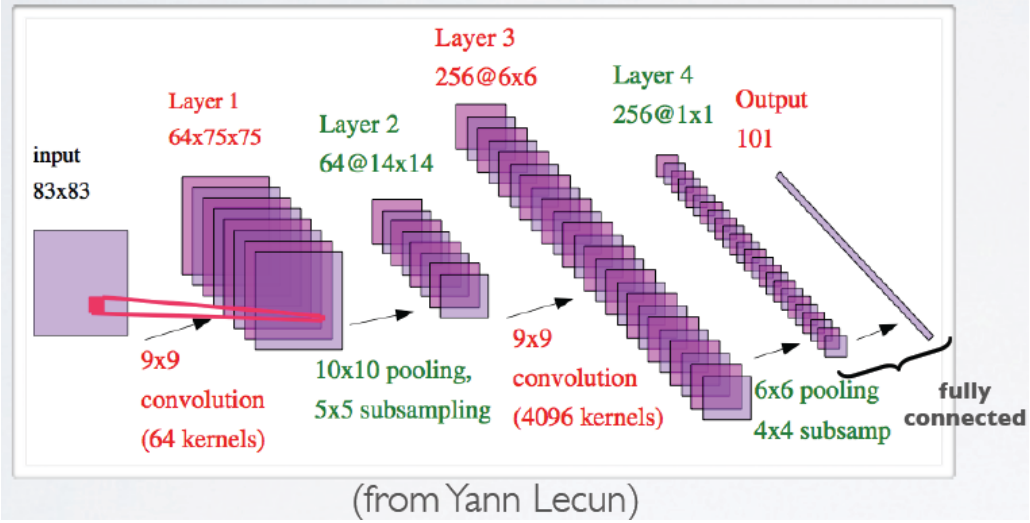


Image



Convolved Feature





(from Yann Lecun)

http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

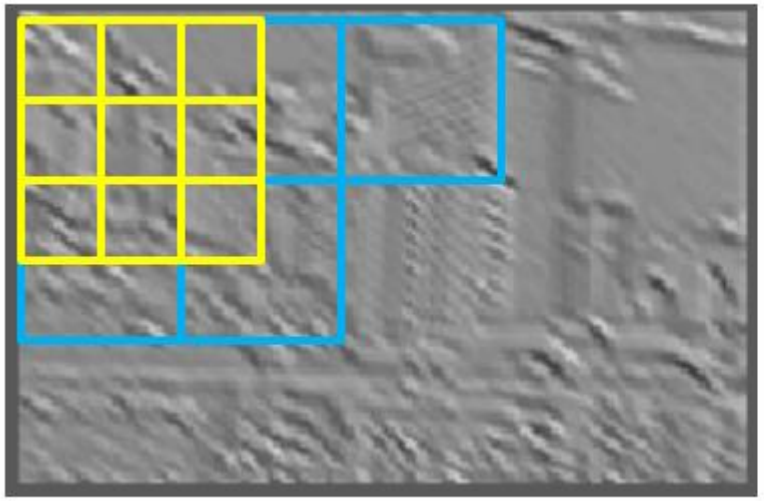
CNN

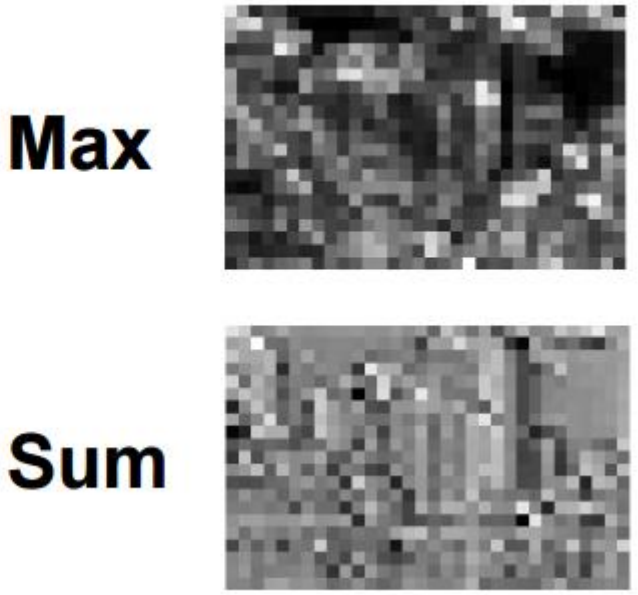
- **Pooling**

  - By pooling (e.g., max or average) filter responses at different locations, we gain robustness to the exact spatial location of features.

  - Invariance to small transformations

  - Reduction of the effect of noises and shift or distortion





ALEXNET

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

ALEXNET

- **Convolutional Neural Networks (CNN)**

  - Comprised of one or more convolutional layers (often with a subsampling step) and then followed by one or more fully connected layers as in a standard multilayer neural network.

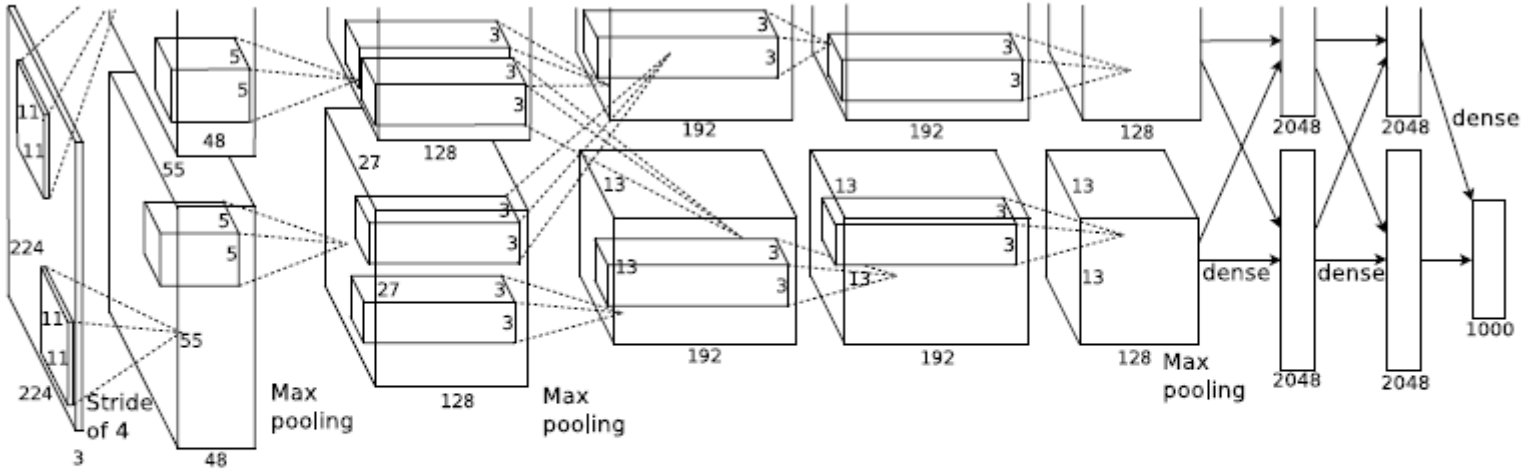
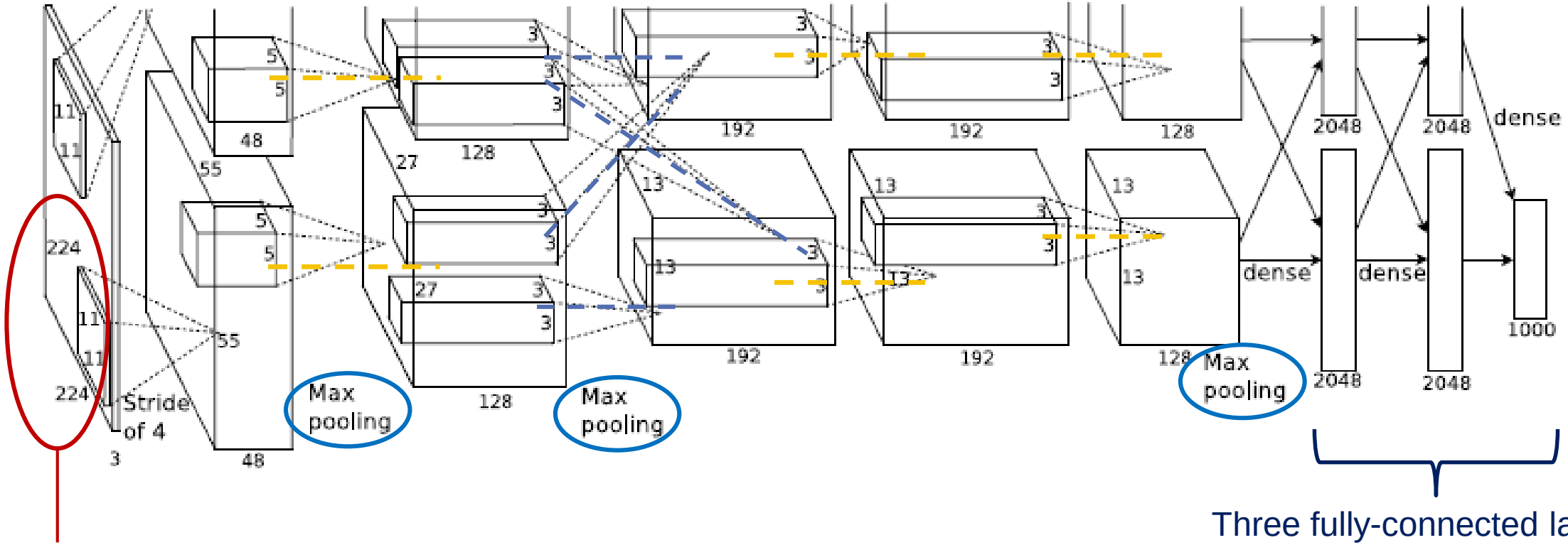


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

ALEXNET



Three fully-connected layers

Extraction of random 224 x 224 patches from the 256 x 256 images
(Increase of the size of the training set by a factor of 2048)

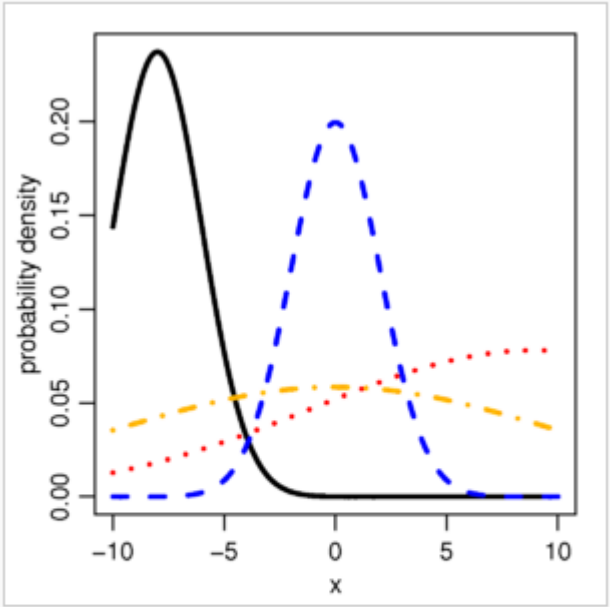
Last layer is fed to a
1000-way softmax.

TENSORFLOW CODE

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

from datetime import datetime
import math
import time

from six.moves import xrange  # pylint: disable=redefined-builtin
import tensorflow as tf


FLAGS = tf.app.flags.FLAGS

tf.app.flags.DEFINE_integer('batch_size', 128,
                            """Batch size.""")
tf.app.flags.DEFINE_integer('num_batches', 100,
                            """Number of batches to run.""")


def print_activations(t):
  print(t.op.name, ' ', t.get_shape().as_list())
```

** Truncated Normal Distribution

Suppose $X \sim N(\mu, \sigma^2)$ has a normal distribution and lies within the interval $X \in (a, b)$, $-\infty \leq a < b \leq \infty$. Then X conditional on $a < X < b$ has a truncated normal distribution.



Probability density function for the truncated normal distribution for different sets of parameters. In all cases, $a = -10$ and $b = 10$. For the black: $\mu = -8$, $\sigma = 2$; blue: $\mu = 0$, $\sigma = 2$; red: $\mu = 9$, $\sigma = 10$; orange: $\mu = 0$, $\sigma = 10$.

https://github.com/tensorflow/tensorflow/blob/master/tensorflow/models/image/alexnet/alexnet_benchmark.py

TENSORFLOW CODE

```
def inference(images):
  """Build the AlexNet model.

  Args:
    images: Images Tensor

  Returns:
    pool5: the last Tensor in the convolutional component of AlexNet.
    parameters: a list of Tensors corresponding to the weights and biases of the
        AlexNet model.
  """
  parameters = []
  # conv1
  with tf.name_scope('conv1') as scope:
    kernel = tf.Variable(tf.truncated_normal([11, 11, 3, 64], dtype=tf.float32,
                                             stddev=1e-1), name='weights')
    conv = tf.nn.conv2d(images, kernel, [1, 4, 4, 1], padding='SAME')
    biases = tf.Variable(tf.constant(0.0, shape=[64], dtype=tf.float32),
                         trainable=True, name='biases')
    bias = tf.nn.bias_add(conv, biases)
    conv1 = tf.nn.relu(bias, name=scope)
    print_activations(conv1)
    parameters += [kernel, biases]
```

```
  # lrn1
  # TODO(shlens, jiayq): Add a GPU version of local response normalization.

  # pool1
  pool1 = tf.nn.max_pool(conv1,
                         ksize=[1, 3, 3, 1],
                         strides=[1, 2, 2, 1],
                         padding='VALID',
                         name='pool1')
  print_activations(pool1)
```

```
  # conv2
  with tf.name_scope('conv2') as scope:
    kernel = tf.Variable(tf.truncated_normal([5, 5, 64, 192], dtype=tf.float32,
                                             stddev=1e-1), name='weights')
    conv = tf.nn.conv2d(pool1, kernel, [1, 1, 1, 1], padding='SAME')
    biases = tf.Variable(tf.constant(0.0, shape=[192], dtype=tf.float32),
                         trainable=True, name='biases')
    bias = tf.nn.bias_add(conv, biases)
    conv2 = tf.nn.relu(bias, name=scope)
    parameters += [kernel, biases]
  print_activations(conv2)
```

- padding = ‘SAME’ : Output has the same dimensions as input.
- padding = ‘VALID’ : No padding

https://github.com/tensorflow/tensorflow/blob/master/tensorflow/models/image/alexnet/alexnet_benchmark.py

TENSORFLOW CODE

```
# pool2
pool2 = tf.nn.max_pool(conv2,
                       ksize=[1, 3, 3, 1],
                       strides=[1, 2, 2, 1],
                       padding='VALID',
                       name='pool2')
print_activations(pool2)
```

```
# conv3
with tf.name_scope('conv3') as scope:
  kernel = tf.Variable(tf.truncated_normal([3, 3, 192, 384],
                                           dtype=tf.float32,
                                           stddev=1e-1), name='weights')
  conv = tf.nn.conv2d(pool2, kernel, [1, 1, 1, 1], padding='SAME')
  biases = tf.Variable(tf.constant(0.0, shape=[384], dtype=tf.float32),
                       trainable=True, name='biases')
  bias = tf.nn.bias_add(conv, biases)
  conv3 = tf.nn.relu(bias, name=scope)
  parameters += [kernel, biases]
  print_activations(conv3)
```

```
# conv4
with tf.name_scope('conv4') as scope:
  kernel = tf.Variable(tf.truncated_normal([3, 3, 384, 256],
                                           dtype=tf.float32,
                                           stddev=1e-1), name='weights')
```

```
def time_tensorflow_run(session, target, info_string):
  """Run the computation to obtain the target tensor and print timing stats.

  Args:
    session: the TensorFlow session to run the computation under.
    target: the target Tensor that is passed to the session's run() function.
    info_string: a string summarizing this run, to be printed with the stats.

  Returns:
    None
  """
  num_steps_burn_in = 10
  total_duration = 0.0
  total_duration_squared = 0.0
  for i in xrange(FLAGS.num_batches + num_steps_burn_in):
    start_time = time.time()
    _ = session.run(target)
    duration = time.time() - start_time
    if i > num_steps_burn_in:
      if not i % 10:
        print ('%s: step %d, duration = %.3f' %
               (datetime.now(), i - num_steps_burn_in, duration))
      total_duration += duration
      total_duration_squared += duration * duration
  mn = total_duration / FLAGS.num_batches
  vr = total_duration_squared / FLAGS.num_batches - mn * mn
  sd = math.sqrt(vr)
  print ('%s: %s across %d steps, %.3f +/- %.3f sec / batch' %
         (datetime.now(), info_string, FLAGS.num_batches, mn, sd))
```

https://github.com/tensorflow/tensorflow/blob/master/tensorflow/models/image/alexnet/alexnet_benchmark.py

TENSORFLOW CODE

```python
def run_benchmark():
  """Run the benchmark on AlexNet."""
  with tf.Graph().as_default():
    # Generate some dummy images.
    image_size = 224
    # Note that our padding definition is slightly different the cuda-convnet.
    # In order to force the model to start with the same activations sizes,
    # we add 3 to the image_size and employ VALID padding above.
    images = tf.Variable(tf.random_normal([FLAGS.batch_size,
                                           image_size,
                                           image_size, 3],
                                          dtype=tf.float32,
                                          stddev=1e-1))

    # Build a Graph that computes the logits predictions from the
    # inference model.
    pool5, parameters = inference(images)

    # Build an initialization operation.
    init = tf.initialize_all_variables()

    # Start running operations on the Graph.
    config = tf.ConfigProto()
    config.gpu_options.allocator_type = 'BFC'
    sess = tf.Session(config=config)
    sess.run(init)

    # Run the forward benchmark.
    time_tensorflow_run(sess, pool5, "Forward")

    # Add a simple objective so we can calculate the backward pass.
    objective = tf.nn.l2_loss(pool5)
    # Compute the gradient with respect to all the parameters.
    grad = tf.gradients(objective, parameters)
    # Run the backward benchmark.
    time_tensorflow_run(sess, grad, "Forward-backward")


def main(_):
  run_benchmark()


if __name__ == '__main__':
  tf.app.run()
```

https://github.com/tensorflow/tensorflow/blob/master/tensorflow/models/image/alexnet/alexnet_benchmark.py

FUNCTION CODE

```
def max_pool(value, ksize, strides, padding, data_format="NHWC", name=None):
  """Performs the max pooling on the input.

  Args:
    value: A 4-D `Tensor` with shape `[batch, height, width, channels]` and
      type `tf.float32`.
    ksize: A list of ints that has length >= 4.  The size of the window for
      each dimension of the input tensor.
    strides: A list of ints that has length >= 4.  The stride of the sliding
      window for each dimension of the input tensor.
    padding: A string, either `'VALID'` or `'SAME'`. The padding algorithm.
      See the [comment here](https://www.tensorflow.org/api_docs/python/nn.html#convolution)
    data_format: A string. 'NHWC' and 'NCHW' are supported.
    name: Optional name for the operation.

  Returns:
    A `Tensor` with type `tf.float32`.  The max pooled output tensor.
  """
  with ops.op_scope([value], name, "MaxPool") as name:
    value = ops.convert_to_tensor(value, name="input")
    return gen_nn_ops._max_pool(value,
                                ksize=ksize,
                                strides=strides,
                                padding=padding,
                                data_format=data_format,
                                name=name)
```

```
function max_pool = maxpooling(patch,dimension)
[m,n] = size(patch);
%judge if the dimension can be divided exactly
max_pool = zeros(floor(m/dimension),floor(n/dimension));
i_limit = floor(m/dimension);
j_limit = floor(n/dimension);
for i=1:i_limit
    for j=1:j_limit
        max_pool(i,j) = max(max(patch((i-1)*dimension+1:i*dimension,(j-1)*dimension+1:j*dimension)));
    end
end
end
```

```
def relu6(features, name=None):
  """Computes Rectified Linear 6: `min(max(features, 0), 6)`.

  Args:
    features: A `Tensor` with type `float`, `double`, `int32`, `int64`, `uint8`,
      `int16`, or `int8`.
    name: A name for the operation (optional).

  Returns:
    A `Tensor` with the same type as `features`.
  """
  with ops.op_scope([features], name, "Relu6") as name:
    features = ops.convert_to_tensor(features, name="features")
    return gen_nn_ops._relu6(features, name=name)
```

RELATED PAPER REVIEW

Brain Tumor Segmentation Using Convolutional Neural Networks in MRI Images

Sérgio Pereira*, Adriano Pinto, Victor Alves, and Carlos A. Silva*

Manuscript received December 15, 2015; revised February 08, 2016; accepted February 26, 2016. Date of publication March 04, 2016; date of current version April 29, 2016. This work is supported by FCT with the reference project UID/EEA/04436/2013, by FEDER funds through the COMPETE 2020—Programa Operacional Competitividade e Internacionalização (POCI) with the reference project POCI-01-0145-FEDER-006941. The work of S. Pereira was supported by a scholarship from the Fundação para a Ciência e Tecnologia (FCT), Portugal (PD/BD/105803/2014). *Asterisk indicates corresponding author.*

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

*S. Pereira is with the CMEMS-UMinho Research Unit, University of Minho, Campus Azurém, 4800-058 Guimarães, Portugal, and also with the Centro Algoritmi, Universidade do Minho, 4710-057 Braga, Portugal (e-mail: id5692@alunos.uminho.pt).

A. Pinto is with the CMEMS-UMinho Research Unit, University of Minho, Campus Azurém, 4800-058 Guimarães, Portugal.

V. Alves is with the Centro Algoritmi, Universidade do Minho, 4710-057 Braga, Portugal (e-mail: valves@di.uminho.pt).

*C. A. Silva is with the CMEMS-UMinho Research Unit, University of Minho, Campus Azurém, 4800-058 Guimarães, Portugal (e-mail: csilva@dei.uminho.pt).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMI.2016.2538465

RELATED PAPER REVIEW

- **Introduction**

  - Gliomas (신경교종) are the brain tumors with the highest mortality rate and prevalence.

  - MRI is especially useful to assess gliomas in clinical practice.

  - The accurate segmentation of gliomas and its intra-tumoral structures is important not only for **treatment planning**, but also for **follow-up evaluations**.

  - However, manual segmentation is **time-consuming** and subjected to inter- and intra-rater errors difficult to characterize.

  - Accurate semi-automatic or automatic methods are required.

RELATED PAPER REVIEW

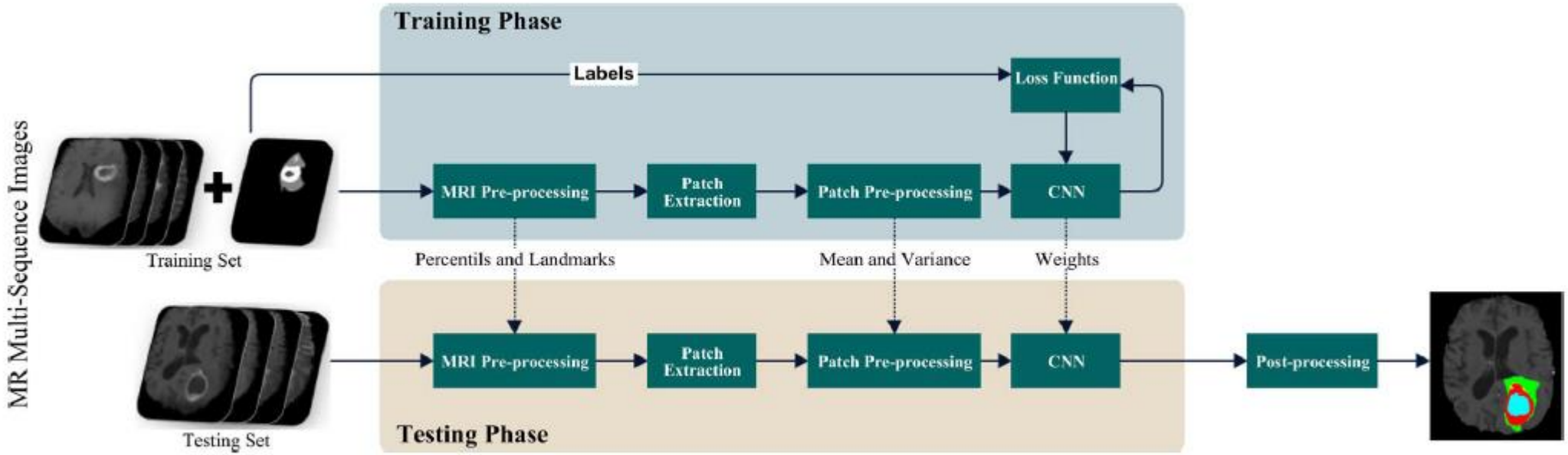


Fig. 1. Overview of the proposed method.

- **Pre-Processing**

- To make the contrast and intensity ranges more similar across patients and acquisitions, they applied the intensity normalization method proposed by Nyúl *et al.*

RELATED PAPER REVIEW

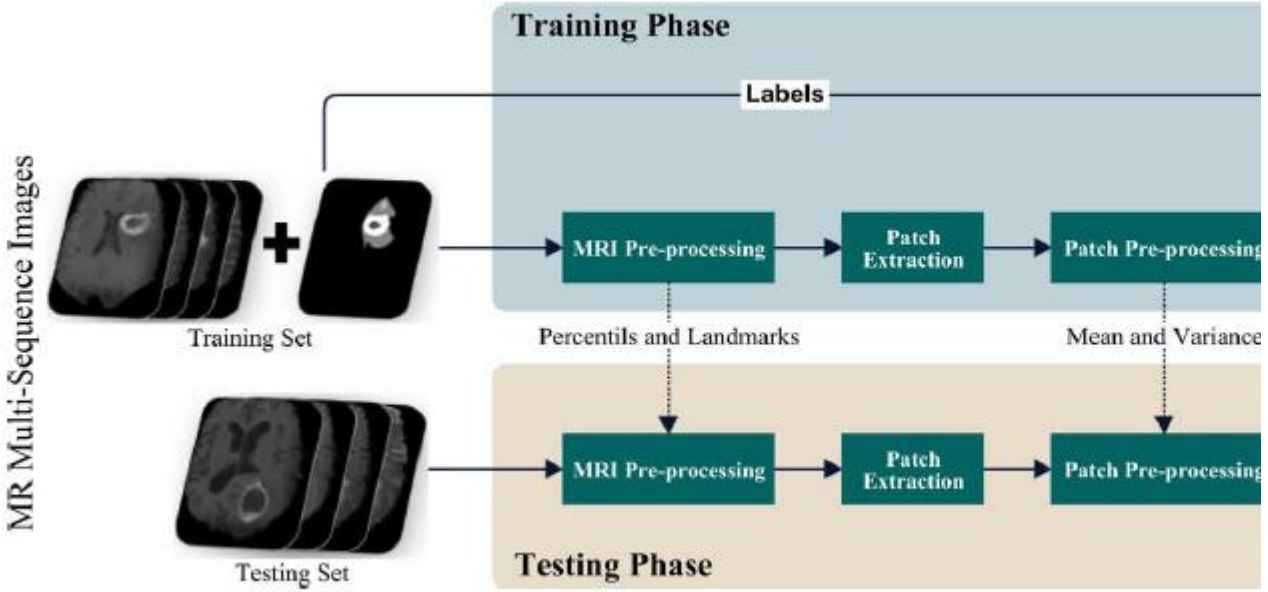


Fig. 1. Overview of the proposed method.

TABLE I
ARCHITECTURE OF THE HGG CNN. IN INPUTS, THE FIRST DIMENSION REFERS TO THE NUMBER OF CHANNELS AND THE NEXT TWO TO THE SIZE OF THE PATCH, OR FEATURE MAPS. CONV. REFERS TO CONVOLUTIONAL LAYERS AND MAX-POOL. TO MAX-POOLING.

	Type	Filter size	Stride	# filters	FC units	Input
			HGG			
Layer 1	Conv.	3× 3	1× 1	64	-	4× 33× 33
Layer 2	Conv.	3× 3	1× 1	64	-	64× 33× 33
Layer 3	Conv.	3× 3	1× 1	64	-	64× 33× 33
Layer 4	Max-pool.	3× 3	2× 2	-	-	64× 33× 33
Layer 5	Conv.	3× 3	1× 1	128	-	64× 16× 16
Layer 6	Conv.	3× 3	1× 1	128	-	128× 16× 16
Layer 7	Conv.	3× 3	1× 1	128	-	128× 16× 16
Layer 8	Max-pool.	3× 3	2× 2	-	-	128× 16× 16
Layer 9	FC	-	-	-	256	6272
Layer 10	FC	-	-	-	256	256
Layer 11	FC	-	-	-	5	256

TABLE II
ARCHITECTURE OF THE LGG CNN. IN INPUTS, THE FIRST DIMENSION REFERS TO THE NUMBER OF CHANNELS AND THE NEXT TWO TO THE SIZE OF THE PATCH, OR FEATURE MAPS. CONV. REFERS TO CONVOLUTIONAL LAYERS AND MAX-POOL. TO MAX-POOLING.

	Type	Filter size	Stride	# filters	FC units	Input
			LGG			
Layer 1	Conv.	3× 3	1× 1	64	-	4× 33× 33
Layer 2	Conv.	3× 3	1× 1	64	-	64× 33× 33
Layer 3	Max-pool.	3× 3	2× 2	-	-	64× 33× 33
Layer 4	Conv.	3× 3	1× 1	128	-	64× 16× 16
Layer 5	Conv.	3× 3	1× 1	128	-	128× 16× 16
Layer 6	Max-pool.	3× 3	2× 2	-	-	128× 16× 16
Layer 7	FC	-	-	-	256	6272
Layer 8	FC	-	-	-	256	256
Layer 9	FC	-	-	-	5	256

- **CNN (Convolutional Neural Network)**

  - The application of convolutional layers consists in convolving an maps.

RELATED PAPER REVIEW

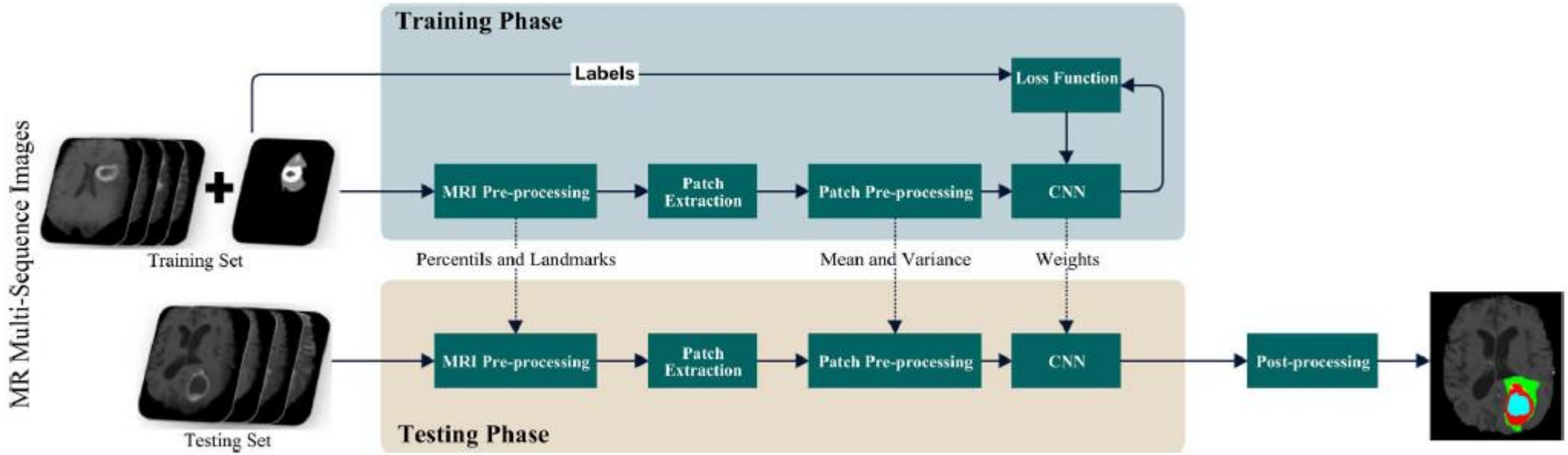


Fig. 1. Overview of the proposed method.

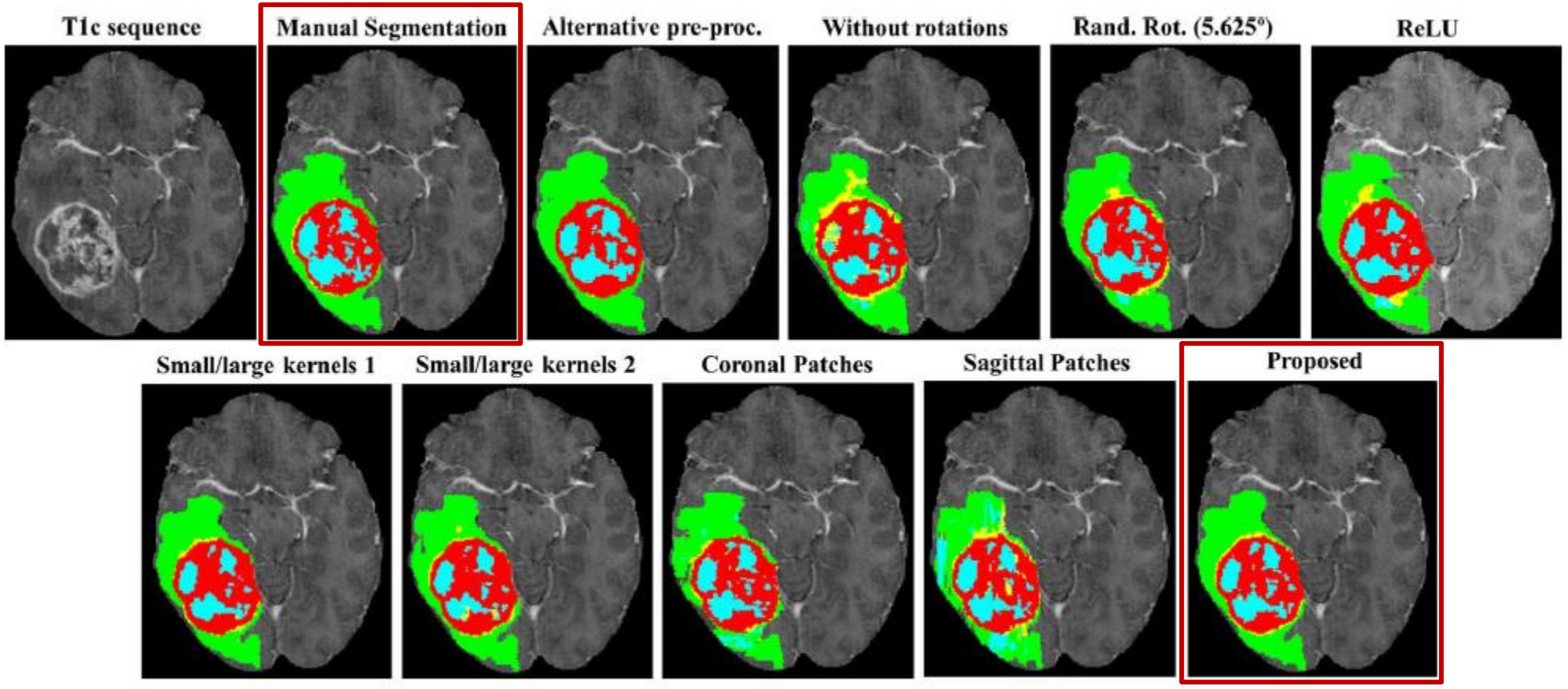
- **Post-Processing**

- Because some small clusters may be classified as tumor, they impose volumetric constrains by removing clusters in the segmentation obtained by the CNN that are smaller than a predefined threshold.

RELATED PAPER REVIEW

- **HGG (High Grade Gliomas) case**

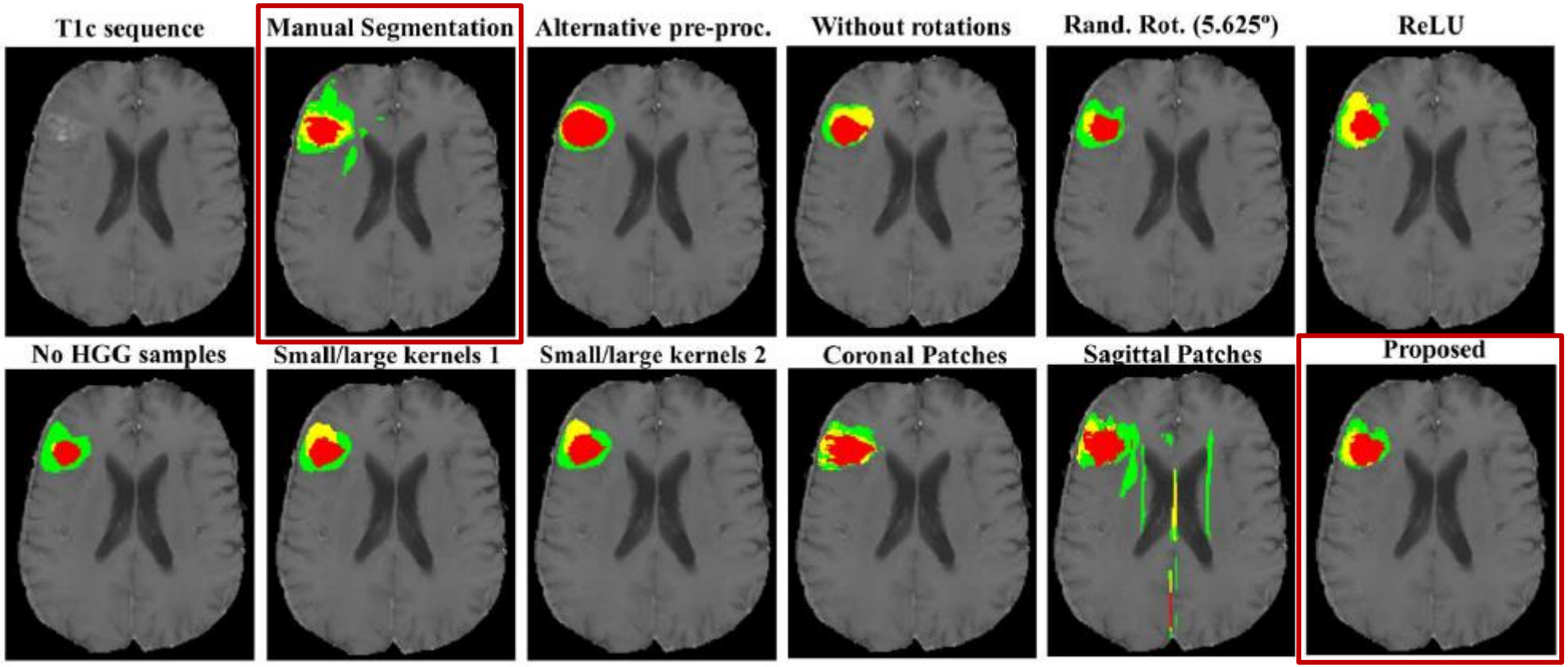
Green : Edema (부종)
Blue : Necrosis (괴사)
Yellow : Non-enhancing tumor
Red : Enhancing tumor



RELATED PAPER REVIEW

- **LGG (Low Grade Gliomas) case**

Green : Edema (부종)
Blue : Necrosis (괴사)
Yellow : Non-enhancing tumor
Red : Enhancing tumor



(b)

RELATED PAPER REVIEW

- **NCI-MICCAI 2013 Grand Challenges in Image Segmentation**
  - Multimodal Brain Tumor Segmentation (BRAT 2013)

TABLE V
RESULTS IN THE LEADERBOARD AND CHALLENGE DATA SETS OF BRATS 2013. THE RELATIVE RANK REFERS TO THE COMBINATION OF THE RANKING IN EACH METRIC FOR THE REFERRED CLASS, WHILE THE POSITION IS THE GLOBAL RANKING, AS PROVIDED BY THE ONLINE EVALUATION PLATFORM [47].

Methods	DSC			PPV			Sensitivity			Relative Rank			Position
	Complete	Core	Enh.	Complete	Core	Enh.	Complete	Core	Enh.	Complete	Core	Enh.	

DSC (Dice Similarity Coefficient) $DSC = \frac{2TP}{FP + 2TP + FN}$

PPV (Positive Predictive Value) $PPV = \frac{TP}{TP + FP}$

Sensitivity $Sensitivity = \frac{TP}{TP + FN}$

	Disease or Condition	No Disease or Condition
Test Positive	A True Positive	B False Positive
Test Negative	C False Negative	D True Negative

RELATED PAPER REVIEW

- **NCI-MICCAI 2013 Grand Challenges in Image Segmentation**
  - Multimodal Brain Tumor Segmentation (BRAT 2013)

TABLE V

RESULTS IN THE LEADERBOARD AND CHALLENGE DATA SETS OF BRATS 2013. THE RELATIVE RANK REFERS TO THE COMBINATION OF THE RANKING IN EACH METRIC FOR THE REFERRED CLASS, WHILE THE POSITION IS THE GLOBAL RANKING, AS PROVIDED BY THE ONLINE EVALUATION PLATFORM [47].

	Methods	DSC			PPV			Sensitivity			Relative Rank			Position
		Complete	Core	Enh.	Complete	Core	Enh.	Complete	Core	Enh.	Complete	Core	Enh.	
Leaderboard	**Proposed**	0.84	0.72	0.62	0.85	0.82	0.60	0.86	0.76	0.68	3.67	3.33	1.67	1
	Kwon et al. [11]	0.86	0.79	0.59	0.88	0.84	0.60	0.86	0.81	0.63	3.33	1.67	5.00	2
	Zhao et al.[8] [5]	0.83	0.73	0.55	0.77	0.67	0.46	0.94	0.89	0.78	4.67	4.00	9.33	3
	agnm1[9]	0.83	0.71	0.54	0.85	0.73	0.59	0.84	0.82	0.58	6.00	4.33	10.33	4
	havam2[9]	0.82	0.69	0.56	0.83	0.77	0.62	0.83	0.69	0.58	7.67	7.00	8.00	5
	Urban et al.[9] [30]	0.70	0.57	0.54	0.65	0.55	0.52	0.87	0.67	0.60	14.00	18.67	12.33	17
	Havaei et al.[10] [32]	0.84	0.71	0.57	0.88	0.79	0.54	0.84	0.72	0.68	–	–	–	–
	Davy et al. [31]	0.72	0.63	0.56	0.69	0.64	0.50	0.82	0.68	0.68	–	–	–	–
Challenge	**Proposed**	0.88	0.83	0.77	0.88	0.87	0.74	0.89	0.83	0.81	7.00	3.33	5.33	1
	Kwon et al. [11], [52]	0.88	0.83	0.72	0.92	0.90	0.74	0.84	0.78	0.72	9.33	5.00	13.00	2
	Tustison et al. [19]	0.87	0.78	0.74	0.85	0.74	0.69	0.89	0.88	0.83	10.33	11.67	9.00	3
	havam2[9]	0.88	0.78	0.73	0.89	0.79	0.68	0.87	0.79	0.80	8.33	10.67	13.33	4
	al-ss1[9]	0.87	0.78	0.70	0.89	0.83	0.75	0.86	0.78	0.70	9.67	8.67	14.67	5
	Urban et al.[8] [30]	0.86	0.75	0.73	0.82	0.75	0.79	0.92	0.79	0.70	11.67	16.00	11.67	12
	Havaei et al. [32]	0.88	0.79	0.73	0.89	0.79	0.68	0.87	0.79	0.80	–	–	–	–
	Davy et al. [31]	0.85	0.74	0.68	0.85	0.74	0.62	0.85	0.78	0.77	–	–	–	–

[8] Results retrieved from [47] using the cited method.

[9] Results retrieved from [47], but the method or author are unknown.

[10] Results provided by the author using the cited method.

THANK YOU FOR YOUR ATTENTION