



DEEP  
LEARNING  
INSTITUTE

(<https://www.nvidia.com/dli/>)

## Medical Image Segmentation using DIGITS

Medical professionals commonly use cardiac MRIs to help diagnose various heart conditions such as hypertrophy (enlargement and thickening of the left ventricle walls), heart failure with myocardial infarction (heart attack) and heart failure without myocardial infarction. The process of reviewing MRIs is time consuming – time that either the patient or the doctor may not have. One of the more common deep learning computer vision tasks – image segmentation – can be applied to cardiac MRIs to assist doctors in quickly and automatically locating the area of the left ventricle for faster diagnosis.

In this lab, we will be using DIGITS to architect, train and validate a neural network that learns to locate the left ventricle in cardiac MRIs. We will be using the 2009 Cardiac MR Left Ventricle Segmentation Challenge dataset. The dataset consists of 16-bit MRI images in DICOM format and expert-drawn contours in text format (coordinates of contour polylines). This dataset was originally published with the following paper:

Radau P, Lu Y, Connelly K, Paul G, Dick AJ, Wright GA. “Evaluation Framework for Algorithms Segmenting Short Axis Cardiac MRI.” The MIDAS Journal – Cardiac MR Left Ventricle Segmentation Challenge, <http://hdl.handle.net/10380/3070> (<http://hdl.handle.net/10380/3070>)

## Process

The approach we take in this lab is as follows:

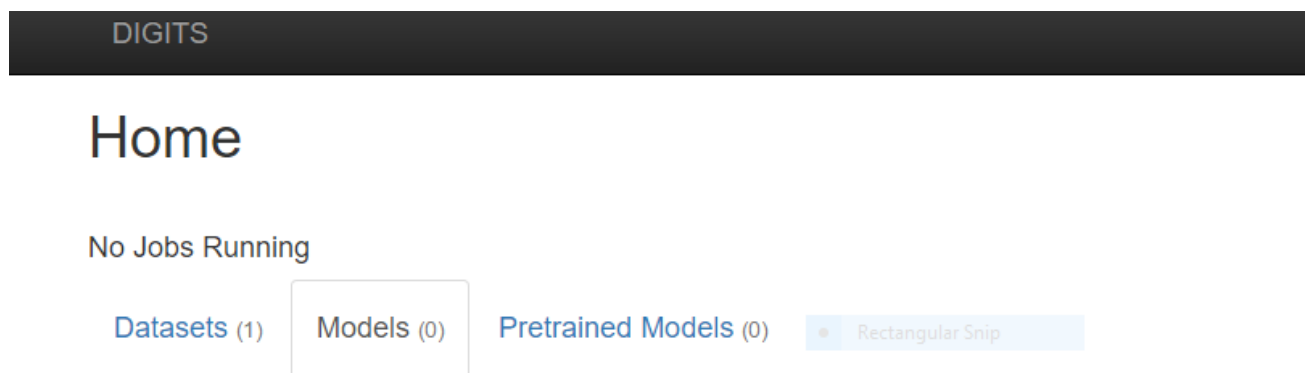
- Announce the Sunnybrook cardiac images dataset to DIGITS and have DIGITS create training and validation files automatically;
- Use a custom network based on AlexNet architecture to train a model;
- Validate the new model;

- Modify the existing model by introducing the Dice Metric;
- Utilize transfer learning to further improve results;
- Create an additional model that performs finer grain image segmentation.

## Creating the dataset in DIGITS

**[Click here \(/digits/\)](#) to open DIGITS.**

Any time you need to return to the DIGITS home page simply click on DIGITS found in the upper left corner as shown in the screenshot below:



We will be using a custom plug-in to load the data into DIGITS. DIGITS plug-ins are thin layers that users can implement to load data that are not in a format that DIGITS natively supports. Plug-ins define a list of options to present to the user as part of the data creation form. Plug-ins also take care of loading the data in Python into a format that DIGITS understands: the ubiquitous `numpy.array` object. See for example the code for the [Image Segmentation plug-in](https://github.com/NVIDIA/DIGITS/tree/278d24b108c5c195ca3d07da82490e187012a2d4/digits/e) (<https://github.com/NVIDIA/DIGITS/tree/278d24b108c5c195ca3d07da82490e187012a2d4/digits/e>)

The `sunnybrook` plug-in reads 16-bit images from DICOM files (those images are the input of the network we will train) and creates black-and-white images out of the `.txt` files that delineate the contours of the left ventricle (those black-and-white images will be our labels).

On the home page, click `New Dataset > Images > Sunnybrook LV Segmentation`.

No Jobs Running

Datasets (0)   Models (0)   Pretrained Models (0)   [Rectangular Snip](#)

Group Jobs: ☒[Delete](#) [Group](#)

name

No Datasets

refs

backend

status

elapsed

New Dataset

Images ▾

Classification

Object Detection

Other

Processing

Segmentation

Sunnybrook LV Segmentation

In the dataset creation form:

- set image folder to `/data/challenge_training`
- set contour folder to `/data/Sunnybrook Cardiac MR Database ContoursPart3`

DIGITS

New Dataset

## New Sunnybrook LV Segmentation Dataset

Image folder ?

[Rectangular Snip](#)  

Contour folder ?

Channel conversion ?

% for validation ?

Feature Encoding ?

- set feature encoding to none
- set encoder batch size to 1

- set dataset name to Sunnybrook
- click Create .

DIGITS

New Dataset

% for validation ?

10

Feature Encoding ?

Rectangular Snip

None

Label Encoding ?

None

Encoder batch size ?

1

Number of encoder threads ?

4

DB backend

LMDB

Enforce same shape ?

Yes

Group Name

Dataset Name

Sunnybrook

Create

When the dataset is created you can refresh the page to review the data shapes:

- features (MRI images) are 256x256 16-bit images
- labels (contour images) are 256x256 black-and-white images

DIGITS

Generic Dataset

Sunnybrook

Owner: ckillam

Rectangular Snip

Job Information

Job Directory

/jobs/20170307-211644-1b1d

Dataset size

100 MB

Create train\_db DB

Entry Count

234

Feature shape ?

(1, 256, 256)

Label shape ?

(1, 256, 256)

labels DB

/jobs/20170307-211644-1b1d/train\_db/labels

features DB

/jobs/20170307-211644-1b1d/train\_db/features

DB create log file

[create\\_train\\_db\\_db.log](#)

Mean file

[train\\_db/mean.binaryproto](#)

Create val\_db DB

Job Status Done

- Initialized
- Running
- Done at 0

(Total - 53 s)

Create train\_db

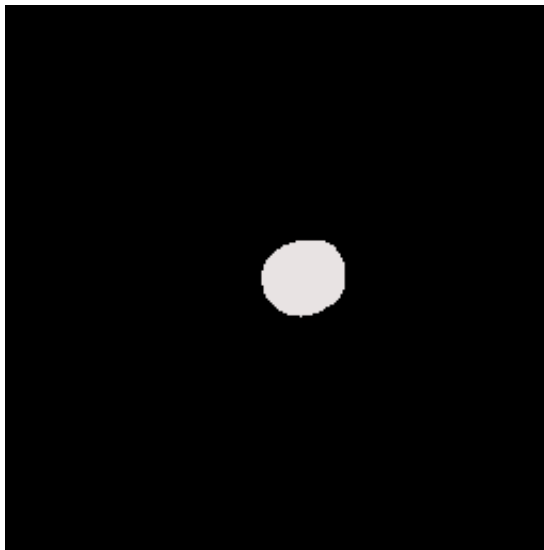
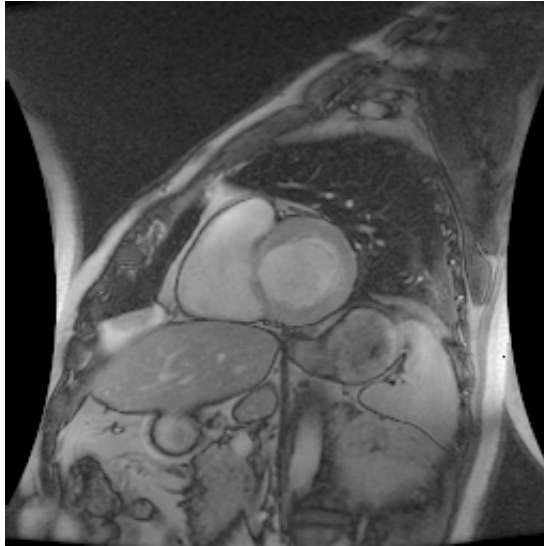
Create val\_db

Create test\_db

Notes

None

A sample feature/label pair may look like:



## Creating a model

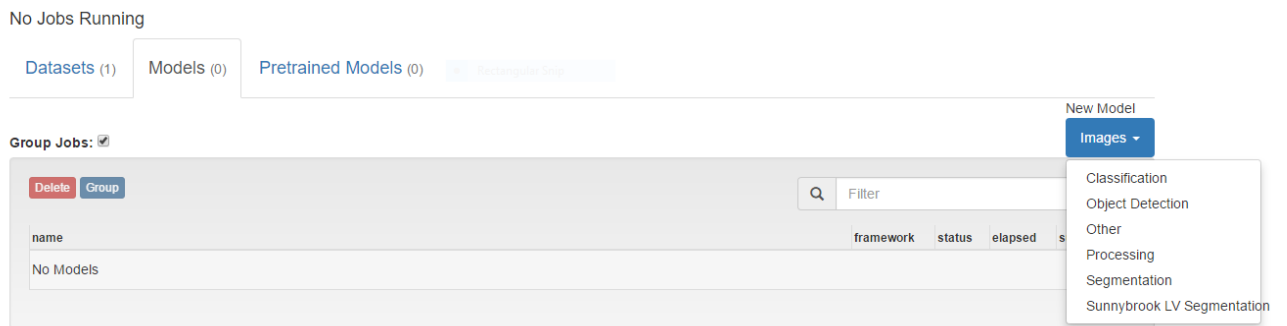
We will use a model from the FCN family. These models were first introduced in the following paper:

Fully Convolutional Models for Semantic Segmentation Jonathan Long, *Evan Shelhamer*, Trevor Darrell CVPR 2015 arXiv:1411.4038

The first model we will create is the simplest model from this family of models. It is derived from the popular AlexNet model.

NOTE: AlexNet is a classification network but with a few minor modifications it can be converted into a segmentation network. See the above paper for more details.

On the DIGITS home page, click `New Model > Images > Sunnybrook LV Segmentation` .



In the model creation form:

- set the number of epochs to 5
- set the snapshot interval to 5
- select the Sunnybrook dataset
- set the base learning rate to  $1e-4$

# New Sunnybrook LV Segmentation Model

**Select Dataset ?**

Sunnybrook

**Sunnybrook**

Done 09:17:37 PM

- DB backend: Imdb
- Create train\_db DB
  - Entry Count:** 234
  - Feature shape** (1, 256, 256)
  - Label shape** (1, 256, 256)
- Create val\_db DB
  - Entry Count:** 26
  - Feature shape** (1, 256, 256)
  - Label shape** (1, 256, 256)

**Python Layers ?**

**Server-side file ?**

☐ **Use client-side file**

**Solver Options** Snip

**Training epochs ?**

5

**Snapshot interval (in epochs) ?**

5

**Validation interval (in epochs) ?**

1

**Random seed ?**

[none]

**Batch size ?** multiples allowed

[network defaults]

**Batch Accumulation ?**

**Solver type ?**

Stochastic gradient descent (SGD) ▼

**Base Learning Rate ?** multiples allowed

1e-4

☐ Show advanced learning rate options

**Data**

**Subti**

Ima

**Crop**

non

- click the `custom network` tab then paste this [net description \(/8DNWSQGz/edit/fcn-alexnet.protobuf\)](/8DNWSQGz/edit/fcn-alexnet.protobuf)



☐ Use client-side file

**Base Learning Rate** ? multiples allowed  
  
☐ Show advanced learning rate options

Standard Networks

Previous Networks

Pretrained Networks

Custom Network

Caffe

Torch

Custom Network ?

Visualize

```
1 # data layers
2 layer {
3   name: "data"
4   type: "Data"
5   top: "data"
6   include {
7     phase: TRAIN
8   }
9   data_param {
10    batch_size: 4
11    backend: LMDB
12  }
13 }
14 layer {
15   name: "label"
16   type: "Data"
17   top: "label"
18   include {
19     phase: TRAIN
```

- name your model fcn\_alexnet-sunnybrook
- click Create

```
334
335 # Dice coefficient
336 #layer {
337 #   type: 'Python'
338 #   name: 'dice'
339 #   bottom: 'score'
340 #   bottom: 'label'
341 #   top: 'dice'
342 #   python_param {
343 #     module: "digits_python_layers"
344 #     layer: "Dice"
345 #   }
346 #   exclude { stage: "deploy" }
347 #}
```

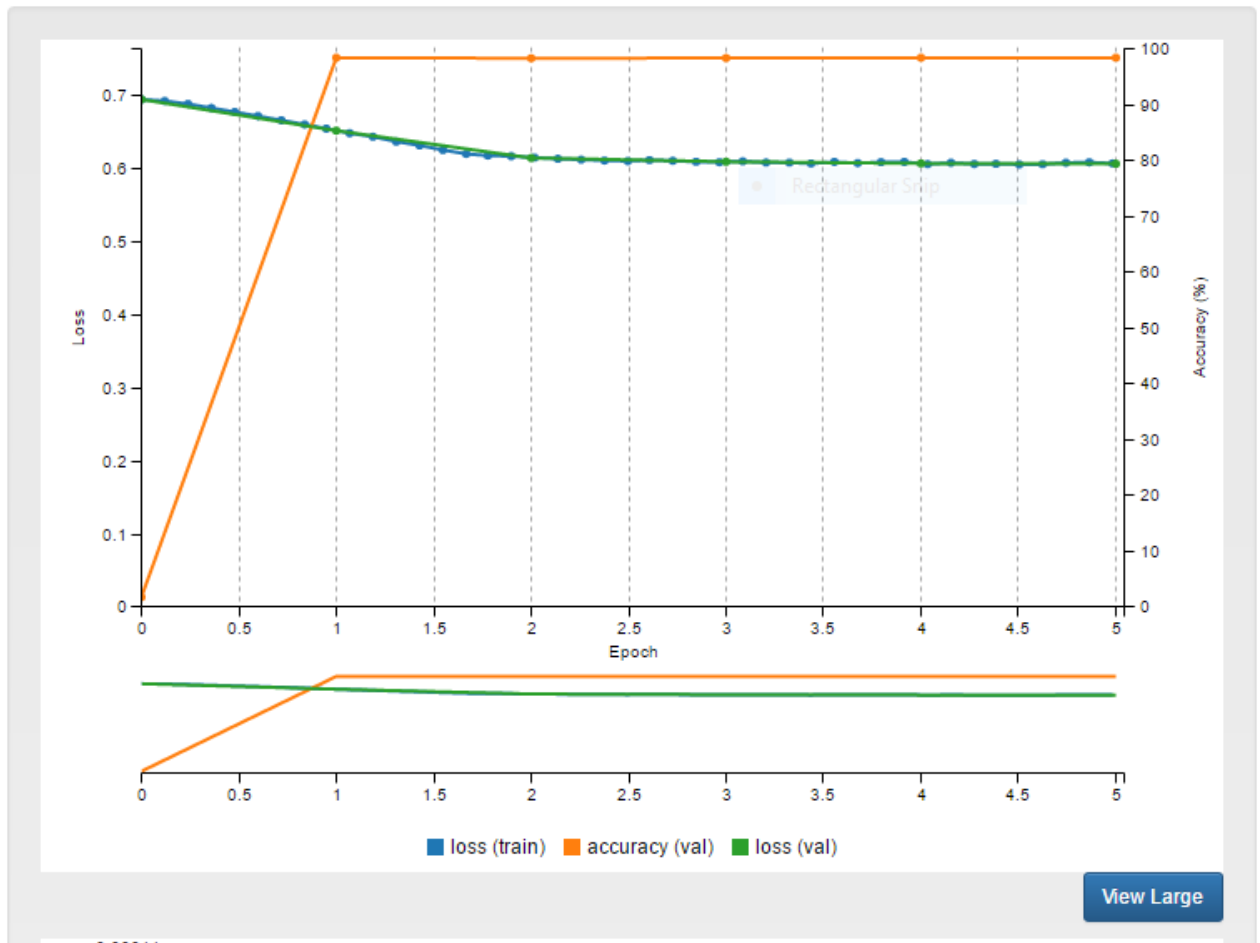
Pretrained model(s) ?

Group Name ?

Model Name ?

Create

You should see that after a few epochs the network has learnt to predict the location of the left ventricle with over 98% accuracy. This means that 98% of pixels are correctly predicted as belonging to either the left ventricle or the background. This is very good!



Now let us test the model on an image from the validation set.

- Scroll down to the model page and select Image Segmentation in Visualization Method .
- Under Select Inference Form ,choose Sunnybrook LV Segmentation
- In Test a record from validation set select SC-HF-NI-3 .
- Check Show visualizations and statistics .
- Click Test .

DIGITS


Generic Image Model

Select Visualization Method

Image Segmentation


Visualization Options

Display segmented image.

Colormap 

From dataset

Inference Options

☐ Do not resize input image(s) 

Select Inference form

Sunnybrook LV Segmentation

Test an image




Image file 

image file

☒ Show visualizations and statistics 

Test

Test a record from validation set

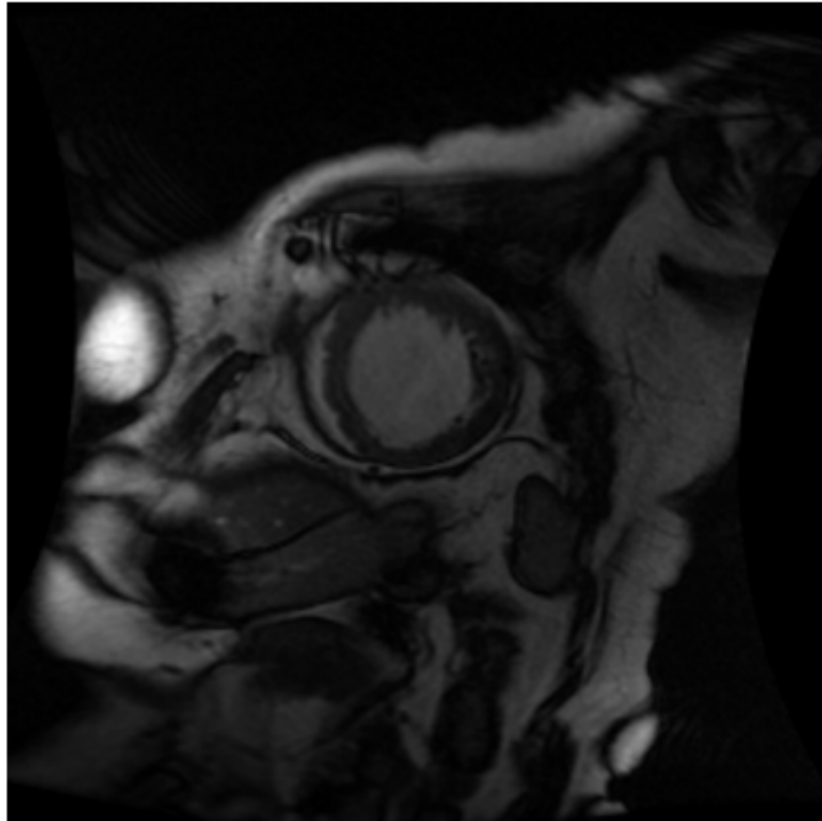
Record from validation set 

SC-HF-NI-3

A new page opens up. There you can see how the image was segmented. Can you spot the location of the left ventricle? Oh no, the segmented image is simply showing the original image! The network didn't find a single pixel of the left ventricle! How is that possible? It turns out that in this dataset, less than 2% of pixels represent the left ventricle. The network has just lazily learnt to classify everything as background. This is a typical "class imbalance" problem.

# Summary

## Output visualizations



### Dice metric

We need a better metric to evaluate the performance of the model. A popular metric for this kind of problems is the Dice coefficient

([https://en.wikipedia.org/wiki/S%C3%B8rensen%E2%80%93Dice\\_coefficient](https://en.wikipedia.org/wiki/S%C3%B8rensen%E2%80%93Dice_coefficient)).

Let us add this metric to our model. By default, Caffe, the underlying Deep Learning framework we are using, does not support this metric. In this case, we add a custom layer written in Python. This custom layer (`layers.py`) is saved as `"/notebooks/layers.py"`.

Select the `Clone` button on the model that you just trained to create a new model with the same settings. In the custom network description, scroll down to the end and un-comment the `dice` layer from the "layer" line to the last closing bracket `'}`' as shown below.

```
324
325 layer {
326     name: "accuracy"
327     type: "Accuracy"
328     bottom: "score"
329     bottom: "label"
330     top: "accuracy"
331     include { stage: "val" }
332 }
333
334
335 # Dice coefficient
336 layer {
337     type: 'Python'
338     name: 'dice'
339     bottom: 'score'
340     bottom: 'label'
341     top: 'dice'
342     python_param {
343         module: "digits_python_layers"
344         layer: "Dice"
345     }
346     exclude { stage: "deploy" }
347 }
```

**Pretrained model(s) ?**

In the 'Python layers' menu, enter the filepath for the custom layer:

`/notebooks/layers.py`

**Python Layers** ?  
**Server-side file** ?  
  
☐ **Use client-side file**

**Batch Accumulation** ?  
  
**Solver type** ?  
  
**Base Learning Rate** ? multiples allowed  
  
☐ Show advanced learning rate options

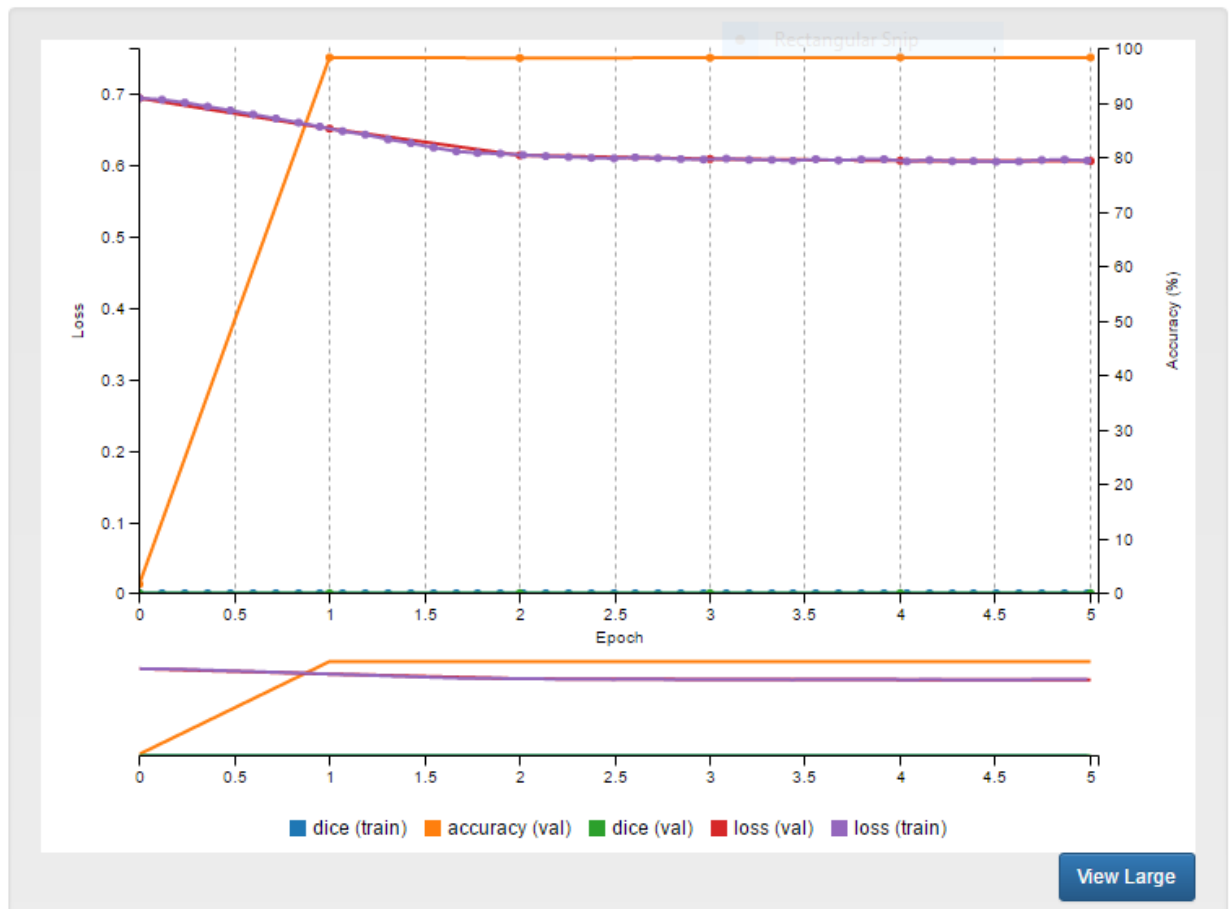
Name your model `fcn_alexnet-sunnybrook-with_dice` . Now click `Create` .

```
343     module: "digits_python_layers"
344     layer: "Dice"
345   }
346   exclude { stage: "deploy" }
347 }
```

**Pretrained model(s)** ?

**Group Name** ?  
  
**Model Name** ?

You will see that even though the accuracy is very high, the Dice coefficient remains at zero. Our model did not learn anything useful but at least we now have a metric to figure that out.



## Transfer learning

Part of the reason why we are unable to learn anything is that our dataset is too small for our model. We would need more samples. An alternative is to do "transfer learning" i.e. re-use knowledge that the model has acquired when training on another dataset and apply that knowledge to a different task. Here we will use a pre-trained Alexnet model that was trained on the 1.2M ImageNet database (ILSVRC2012).

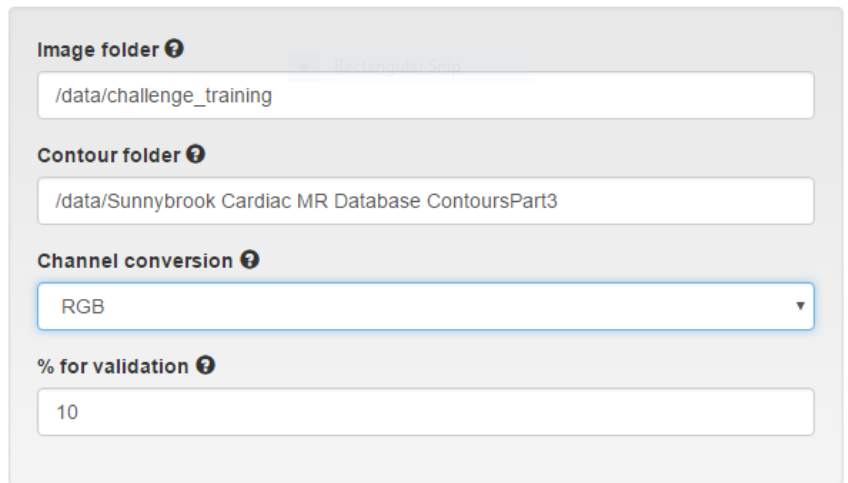
In order to be able to use the pre-trained Alexnet model, we need to make a modification to our dataset. The Sunnybrook dataset is currently in grayscale; however, the pre-trained model is expecting three channels - RGB. DIGITS provides an easy way to make this change. Go to your home page and clone the Sunnybrook dataset. Change Channel Conversion setting from Grayscale to RGB as shown below.



DIGITS

New Dataset

## New Sunnybrook LV Segmentation Dataset



The screenshot shows a web form for creating a new dataset in the DIGITS interface. The form is titled 'New Sunnybrook LV Segmentation Dataset'. It contains four main sections, each with a label, a help icon, and a text input field:


- Image folder**: The input field contains the path `/data/challenge_training`. A small blue icon with a question mark is to the right of the label.
- Contour folder**: The input field contains the path `/data/Sunnybrook Cardiac MR Database ContoursPart3`. A small blue icon with a question mark is to the right of the label.
- Channel conversion**: A dropdown menu is set to 'RGB'. A small blue icon with a question mark is to the right of the label.
- % for validation**: The input field contains the value '10'. A small blue icon with a question mark is to the right of the label.

Name the new copy Sunnybrook-RGB and click create.

Clone the last model you trained. Select the new Sunnybrook-RGB dataset and set the number of epochs to 20 .

Check the box `Show advanced learning rate options` and set the learning rate policy to `fixed` .

**Python Layers ?**  
**Server-side file ?**  
  
☐ Use client-side file

**Batch size ?** multiples allowed  
  
**Batch Accumulation ?**  
  
**Solver type ?**  
  
**Base Learning Rate ?** multiples allowed  
  
☒ Show advanced learning rate options  
**Policy**  
  
Visualize LR  


In pretrained model point DIGITS to this path: `/data/fcn_alexnet.caffemodel` .

103    stride: 1

**Pretrained model(s) ?**

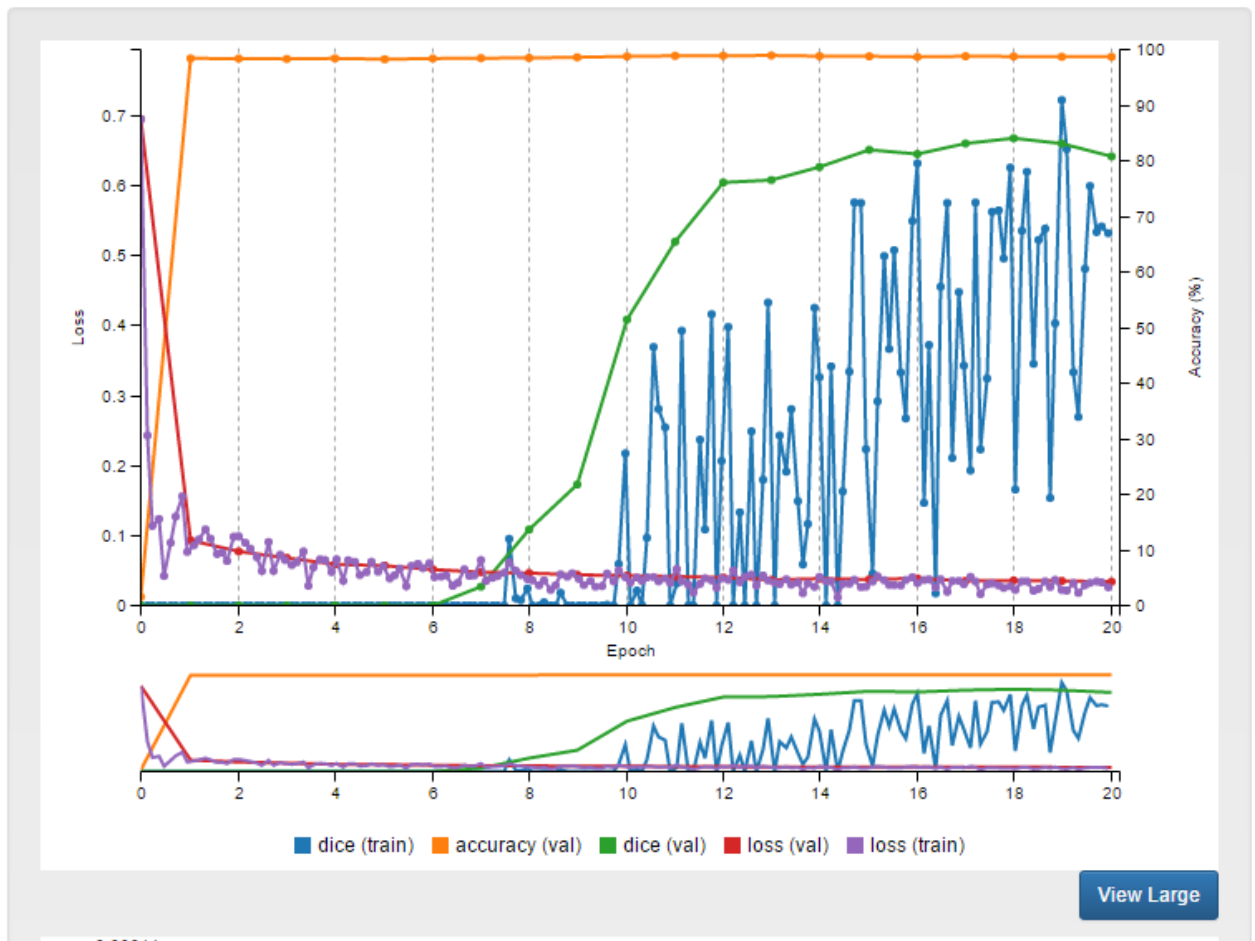
**Group Name ?**

**Model Name ?**  
  
Create

Name your model `fc_alexnet-sunnybrook-with_dice-pretrained` . Now click Create .

NOTE: This will likely take around 10 minutes to complete.

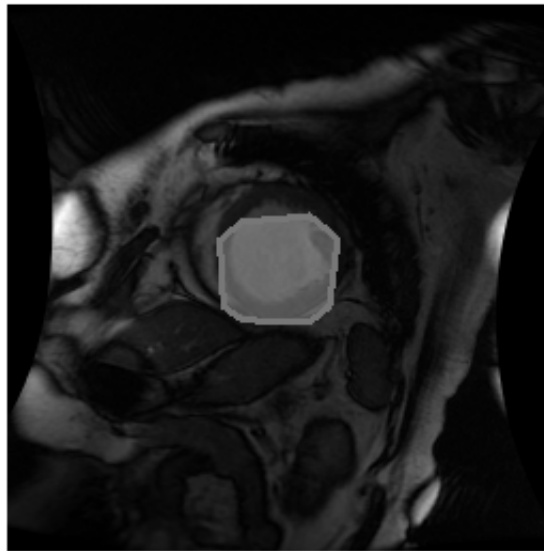
After training you will see that the Dice coefficient exceeds 65%.



Now test the same image again.

## Summary

### Output visualizations



□ left ventricle

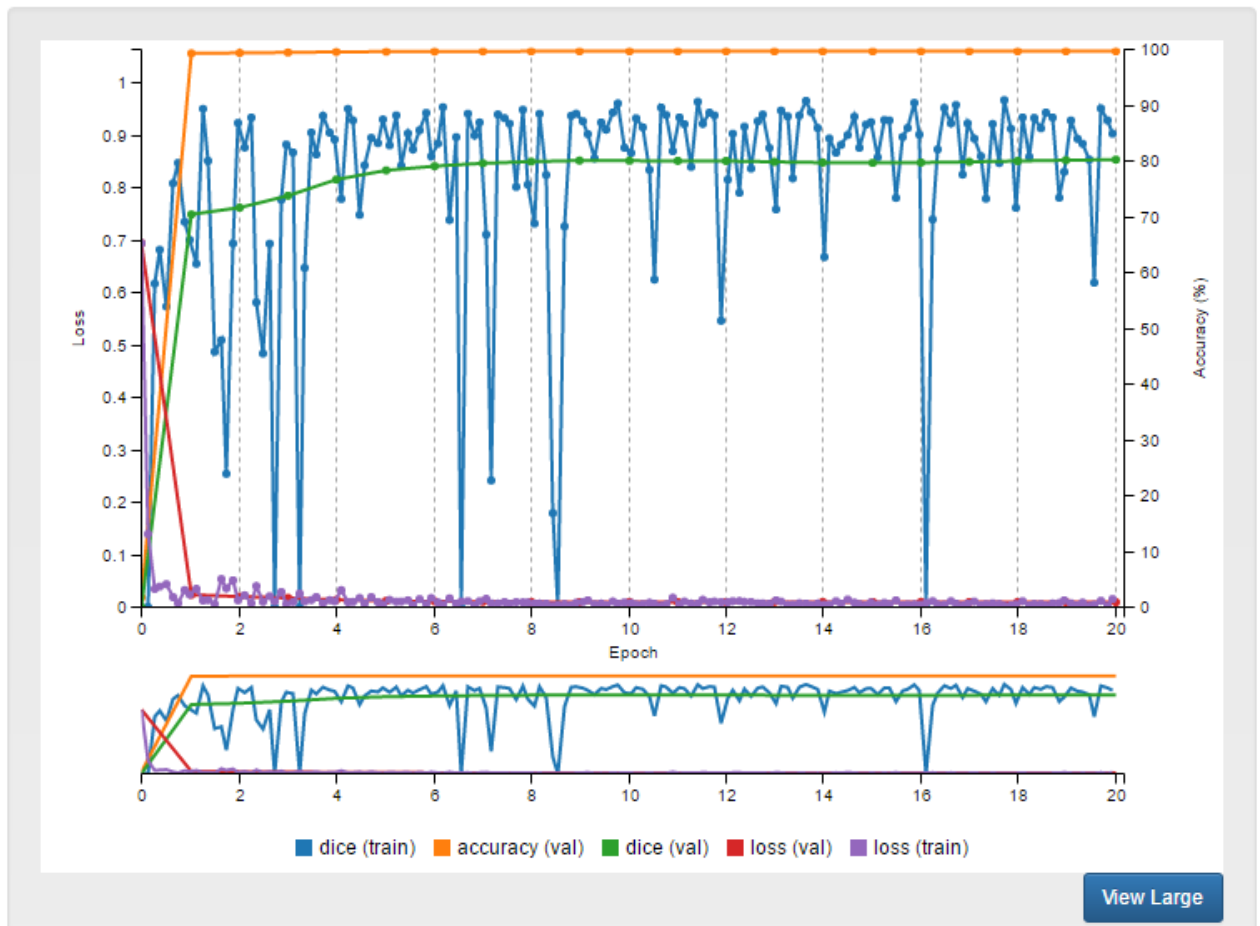
You will notice a greyish / white blob on the image which corresponds to the network's prediction of the location of the left ventricle. As you can see, the edges are pretty rough. This is because the predictions in FCN-Alexnet are made at a 32-pixel stride, meaning that only one prediction is made for each block of 32x32 pixels. This is too coarse.

## FCN-8s

We will now use a finer model called FCN-8s. This model is able to make predictions at a much finer grain - down to 8x8 blocks. Clone the model you just trained. Set the batch size to 1 to reduce the memory footprint during training and set the number of epochs to 3. NOTE: Time constraints warrant the use of 3 epochs; however, in practice you would likely specify a higher epoch value. The screen captures that follow show the results when the number of epochs is 20 (so that a comparison can be made with the AlexNet example from the previous step).

In the custom network tab paste this [`fcn-8s.protobuf`](#) `/(8DNWSQGz/edit/fcn-8s.protobuf)`. In pretrained model point DIGITS to this path: `/data/fcn8s-heavy-pascal.caffemodel`. Name your model `fcn_8s-sunnybrook-with_dice-pretrained`.

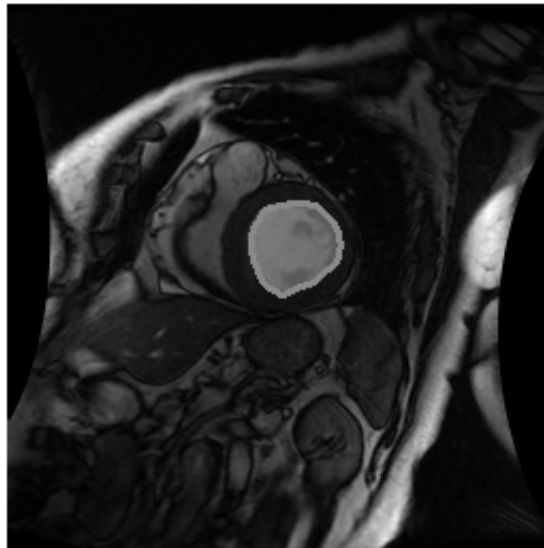
After training you should see that the model reaches a Dice coefficient of over 88%.



Now test the same image again, you should see that the contour of the prediction is much smoother.

## Summary

### Output visualizations



□ left ventricle

### REFERENCE: FCN-Alexnet and FCN-8s network descriptions

- [fcn-alexnet.protobuf](#) (/8DNWSQGz/edit/fcn-alexnet.protobuf)
- [fcn-8s.protobuf](#) (/8DNWSQGz/edit/fcn-8s.protobuf)



DEEP  
LEARNING  
INSTITUTE

(<https://www.nvidia.com/dli/>)

