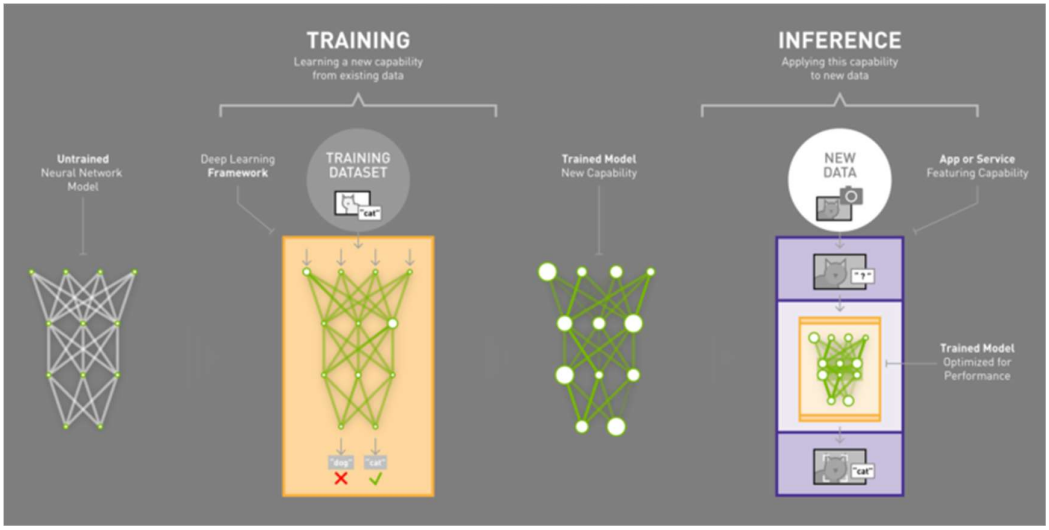
Fundamentals of Deep Learning for Computer Vision- Course Report

1. Training Deep Neural Networks
   a) Deep Neural Networks: GPU Task 1

   Deep Neural Networks are flexible algorithms inspired by the human brain that allow practitioners to use training strategies inspired by human learning. The input of an image generated an output of the network's confidence that the image belonged to one of two classes. Something clearly changed between the first epoch and the 100th. It indicates algorithms would learn the experience from the huge dataset. A neural network changes when exposed to data to create an accurate map between inputs and outputs.

   b) Deep Neural Networks: GPU Task 2

   In this task, I mainly learnt the workflow of training the model with data. The following notes are the workflow from course.



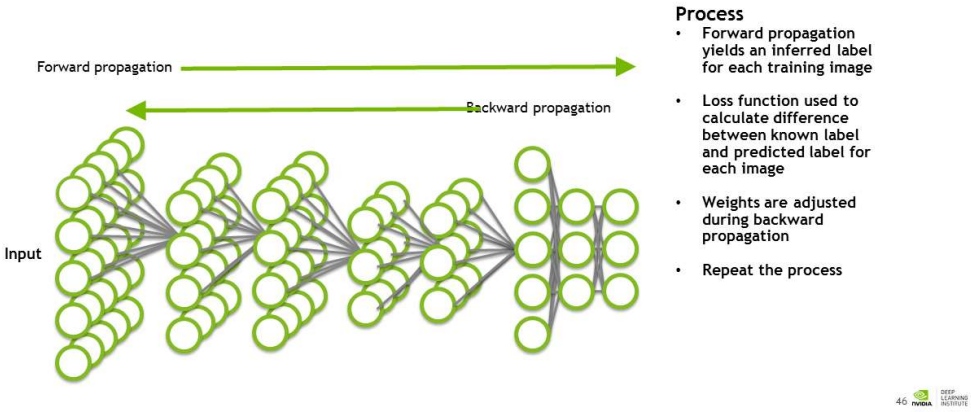
   Data processing:

   1) Standardize them to the same size to match what the network you are training expects. We'll be training AlexNet again which was designed to take an input of 256X256 color images. Expect more on this in the next task.

   2) Split them into two datasets, where 75% of each class is used for training and 25% is set aside for validation.

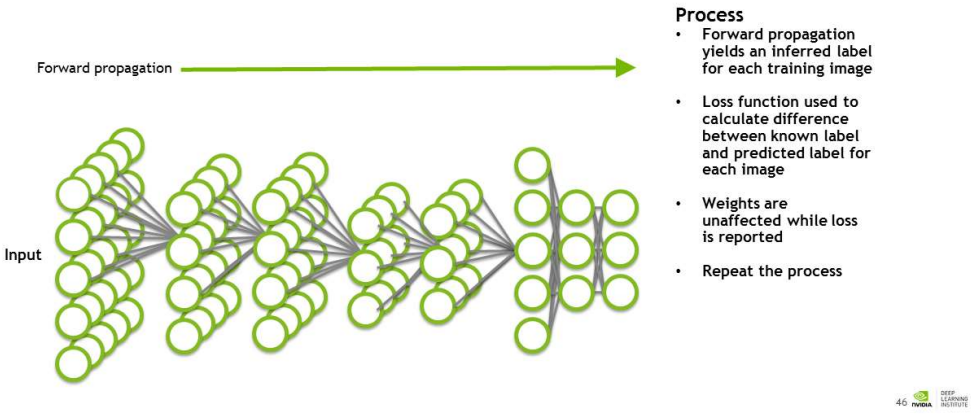
   We could use training accuracy, validation loss, training loss to measures the performance of model.

DEEP LEARNING APPROACH - TRAINING

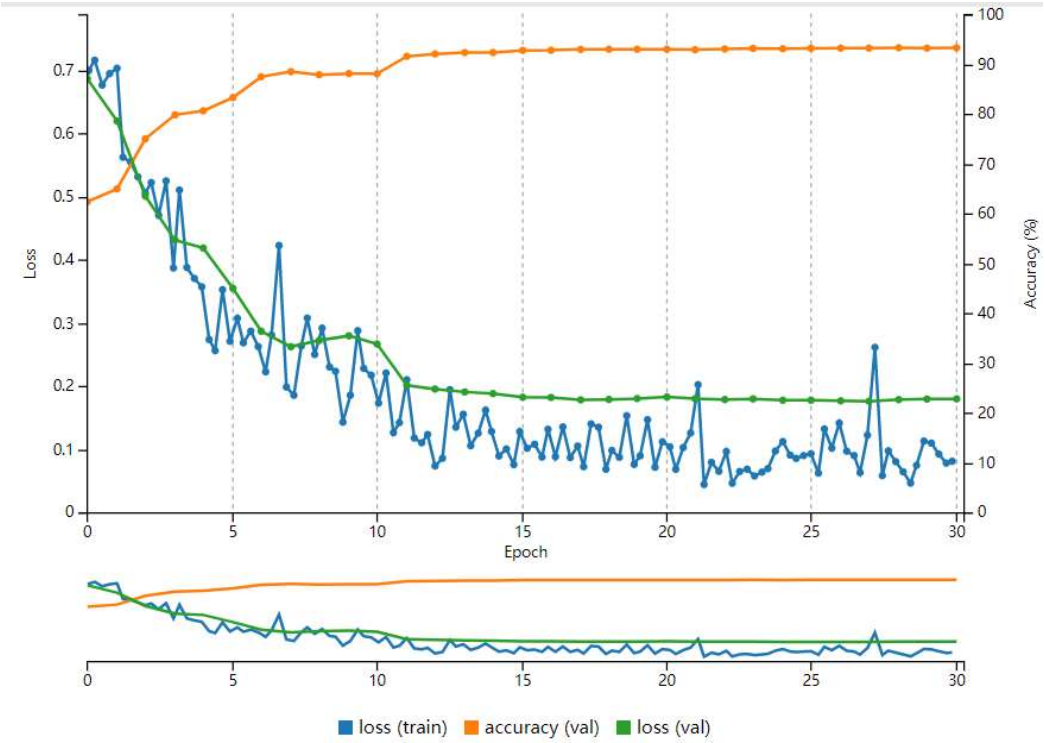


The validation dataset will be used to assess performance on *new data* using a technique that human learning can not. Validation data is fed through the network to generate an output, but the network *does not learn anything* from the data.

DEEP LEARNING APPROACH - VALIDATION



And I also did the experiment on cats and dogs dataset. Here is the result of the experiment.
I used alexnet as my model, here is the performance graph of alexnet.



Its validation accuracy is 93%, training loss is 8% and validation loss is 18%. Its performance is very good.

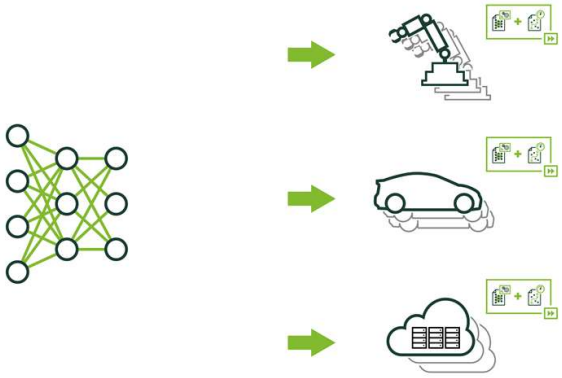
2. Deploying Trained Neural Networks
   a) Deploying our Model: GPU Task 3

   In this task, I mainly learnt knowledge about how to deploy trained networks into applications to solve problems in the real world. The following are my notes.

   Deployment is the work of taking a trained model and putting it to work as a part of an application. You can deploy to edge devices such as robots or autonomous vehicles. You can also deploy to a server in order to play a role in any piece of software.

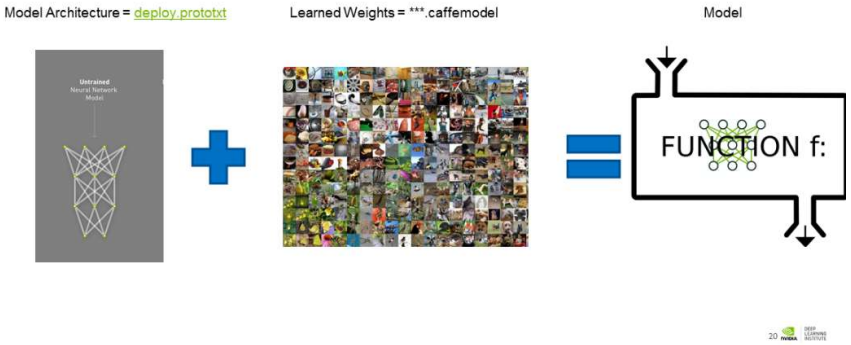
Deployment

How do I use a trained neural network as part of a solution?



To successfully *deploy* a trained model, we have two initial jobs.
1) Our first job is to provide our model an input that it expects.
2) Our second job is to provide our end user an output that is useful.

Components of a Model



Also, I deployed the model according to the requirement.
First, we should preprocessing the data, change the format for the expected input.

```
# Create an input that the network expects.
input_image= caffe.io.load_image(img_path)
test_image = cv2.resize(input_image,  (256,256))
mean_image = caffe.io.load_image(DATASET_JOB_DIR + '/mean.jpg')
test_image = test_image-mean_image
```

And then loading the model

```
# Initialize the Caffe model using the model trained in DIGITS.
net = caffe.Classifier(ARCHITECTURE, WEIGHTS,
                                channel_swap=(2,1,0),
                                raw_scale=255,
                                image_dims=(256, 256))
```

Finally, make a prediction

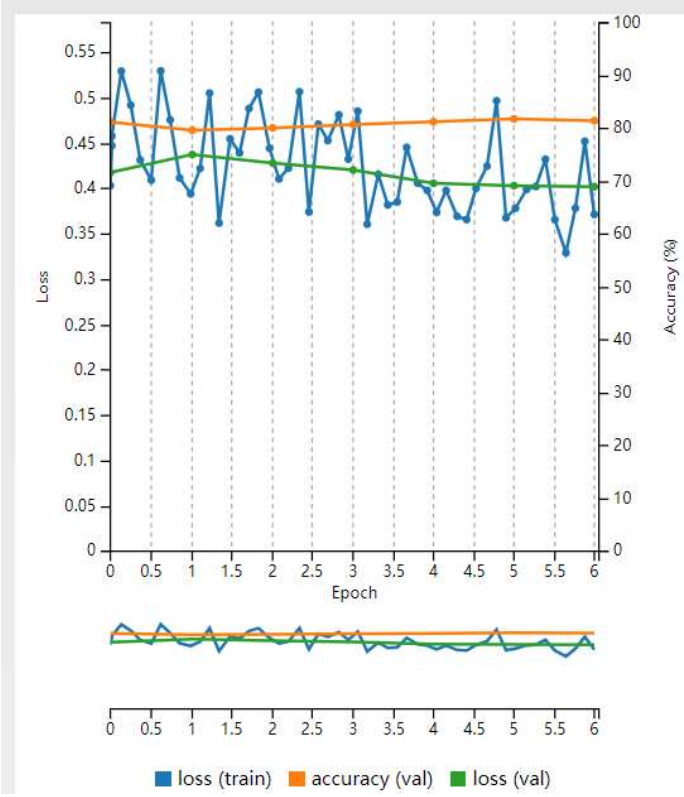
```
prediction = net.predict([test_image])

#print("Input Image:")
#plt.imshow(sys.argv[1])
#plt.show()
#Image.open(input_image).show()
print(prediction)
##Create a useful output
print("Output:")
if prediction.argmax()==0:
      print "Sorry cat:( https://media.giphy.com/media/jb8aFEQk3tADS/giphy.gif"
else:
      print "Welcome dog! https://www.flickr.com/photos/aidras/5379402670"
```

3. Measuring and Improving Performance
   a) Performance

   GPU Task 4

   In this task, the most important thing that I learnt is the reason of why learning rate decrease. The learning rate decreases throughout the training session because the network is getting closer to its ideal solution.

   According to requirement, I select cat vs dog model as pretrained model and train it again.



   This is the graph shows that the performance of the new model.

   After that, I followed the instruction and downloaded the award winning models. The winning model is trained on ImageNet. I used the wget command to download the pretrained model and dataset. Then, I loaded the

```
#Load the image
image= caffe.io.load_image(TEST_IMAGE)
plt.imshow(image)
plt.show()

#Load the mean image
mean_image = np.load(MEAN_IMAGE)
mu = mean_image.mean(1).mean(1)  # average over pixels to obtain the mean (BGR) pix

# create transformer for the input called 'data'
transformer = caffe.io.Transformer({'data': net.blobs['data'].data.shape})
transformer.set_transpose('data', (2,0,1))  # move image channels to outermost dime
transformer.set_mean('data', mu)            # subtract the dataset-mean value in ea
transformer.set_raw_scale('data', 255)      # rescale from [0, 1] to [0, 255]
transformer.set_channel_swap('data', (2,1,0))  # swap channels from RGB to BGR
# set the size of the input (we can skip this if we're happy with the default; we c
net.blobs['data'].reshape(1,        # batch size
                          3,        # 3-channel (BGR) images
                          227, 227)  # image size is 227x227

transformed_image = transformer.preprocess('data', image)
```

dataset.

And then I made prediction with the following code.

```
# copy the image data into the memory allocated for the net
net.blobs['data'].data[...] = tranAsformed_image

### perform classification
output = net.forward()

output
```

At last, I made output useful for users.

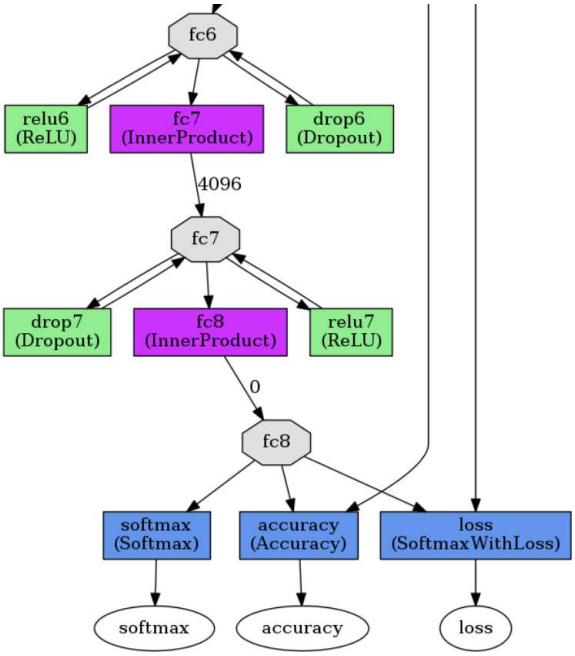
```
!wget https://raw.githubusercontent.com/HoldenCaulfieldRye/caffe/master/data/ilsvrc
labels_file = 'synset_words.txt'
labels = np.loadtxt(labels_file, str, delimiter='\t')

print 'output label:', labels[output_prob.argmax()]
```

GPU Task 5: Object Detection

Here are my notes from the course.

Workflow	Input	Output
Image Classification	Raw Pixel Values	A vector where each index corresponds with the likelihood or the image of belonging to each class
Object Detection	Raw Pixel Values	A vector with (X,Y) pairings for the top-left and bottom-right corner of each object present in the image
Image Segmentation	Raw Pixel Values	A overlay of the image for each class being segmented, where each value is the likelihood of that pixel belonging to each class
Text Generation	A unique vector for each 'token' (word, letter, etc.)	A vector representing the most likely next 'token'
Image Rendering	Raw Pixel Values of a grainy Image	Raw pixel values of a clean image

In this task, our goal is to detect and localize objects within images. One of methods is to slide window with non-overlapping grid squares. The other approach is to rebuild from an existing neural network. First, we should know the architecture of AlexNet. According to jupyter notebook, we visualize the architecture of AlexNet.



The picture above is part of architecture. And this method is to covert AlexNet to fully convolutional new work because convolutional network could be fed by various size of images without first splitting them into grid squares.

The third method is to use object detection network designed to detect and localize dogs, which name is DetectNet. And by the task, I learnt that the measurement of object detection is difference from classification. Object detection's measure methods are loss_bobox(train), loss_bbox(val), loss_coverage(val), mAp(val), precision(val), recall(val), loss_coverage(train). The mean Average Precision (mAP) is a combined measure of how well the network can detect the dogs and how accurate its bounding box (localization) estimates were for the validation dataset.

4. Assessment

   I followed the instruction to make dataset, train the model and deploy the model. Here is the deploy model code.

```
MODEL_JOB_DIR = '/dli/data/digits/20190226-202036-51b2'
ARCHITECTURE = MODEL_JOB_DIR + '/' + 'deploy.prototxt'
WEIGHTS = MODEL_JOB_DIR + '/' + 'snapshot_iter_735.caffemodel'
print ("Filepath to Architecture = " + ARCHITECTURE)
print("Filepath to weights = "+ WEIGHTS)
DATA_JOB_DIR = '/dli/data/digits/20190226-201730-49f1'


caffe.set_mode_gpu()

# Initialize the Caffe model using the model trained in DIGITS. Which two files constitute your trained model?
net = caffe.Classifier(ARCHITECTURE, WEIGHTS,
                                    channel_swap=(2,1,0),
                                    raw_scale=255,
                                    image_dims=(256, 256))

# Create an input that the network expects. This is different for each project, so don't worry about the exact steps, but find the dataset job
tory to show you know that whatever preprocessing is done during training must also be done during deployment.
input_image= caffe.io.load_image(img_path)
input_image = cv2.resize(input_image, (256,256))
mean_image = caffe.io.load_image(DATA_JOB_DIR+'/mean.jpg')
input_image = input_image-mean_image

# Make prediction. What is the function and the input to the function needed to make a prediction?
prediction = net.predict([input_image])##REPLACE WITH THE FUNCTION THAT RETURNS THE OUTPUT OF THE NETWORK##([##REPLACE WITH THE INPUT TO THE
ION##])

# Create an output that is useful to a user. What is the condition that should return "whale" vs. "not whale"?
if prediction.argmax()==1: ##REPLACE WITH THE CONDITION THAT WOULD MAKE THE FUNCTION RETURN WHALE##:
      return "whale"
else:
      return "not whale"
```

I pass the assessement successfully.