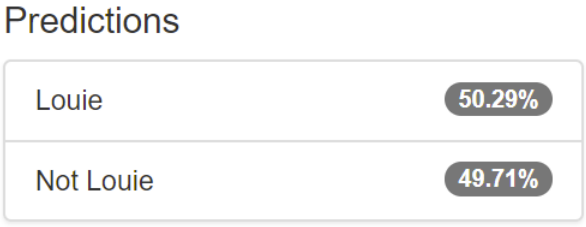
GPU TASK 1:
Across various industries.
Alexa (which converts speech to language). We can interact more we can make more predictions. Computer vision ability to search by itself in self driving cars. Using result of vision to do something. Computer vision also helps in diagnosis like MRI.
How do we teach where do we teach? NLP, Computer Vision, Drug discovery are some of various applications of Deep Learning.
First, we have to find what sort of problems are deep learning?
Second, we have to think about How do we solve the problem.

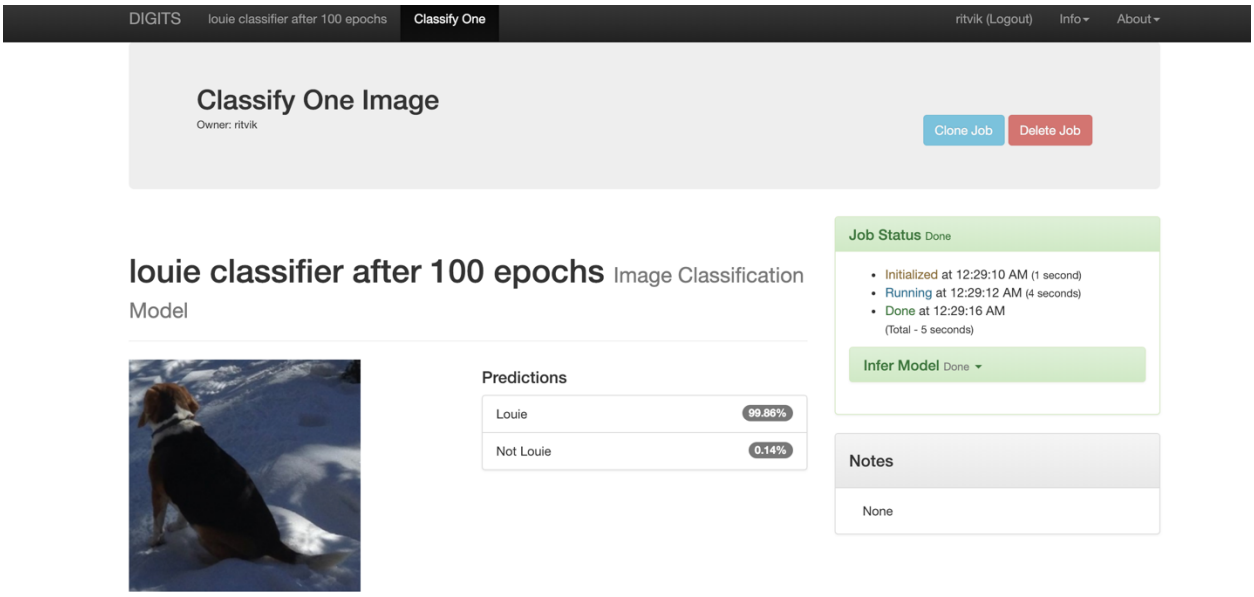
So how/why is deep learning being used?
Achieving complex goals
Connecting dots and forming new ideas
Example of teaching a machine to identify the dogs
Driving successfully in various environments (day, night etc.)

In Task 1 first we created a model by choosing a dataset, a neural network and indicating number of epochs to run.



Ran for 1 epochs and it showed 50% as louie and 49% not a louie.
our model was predicting no better than chance: 50/50

Second we are traning the same network for 100 epochs.



After 100 epochs the accuracy increased, and classifier started doing really well.

Neural Network principle:

1) Assumes random weights the inputs are passed through hidden layers. Each hidden layer has an activation function. Passed through activation function and enters the enters the next layers.
2) After prediction loss function is calculated. Then gradient of that loss function is calculated such that it is minimized.
3) Using back propagation Weights are updated using that gradients.
4) Prediction is done again using the new weights.

GPU TASK 2:

In GPU task 1 we were training very small models. But in the era of big data we create ~2.5 quintillion bytes of data per day. labeled facial recognition datasets are huge in size. Now we will use Kaggle dataset

First, we load our dataset:
Start Digits
Go to dataset -> new classification dataset.
We load the dataset by pointing digits to the folder location.

The data was structured in such a way that there was a folder with dogs and another with cats. We've put both the folders together called "dogscats". Such there was 2 classes. We made validation data around 25%.

Next what is digits doing?

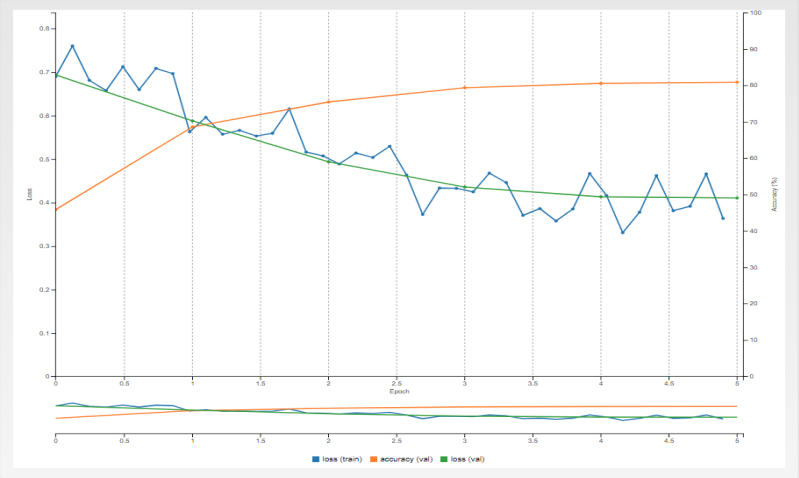
First its standardizing all the inputs to a size of 255*255.

Then splitting the data set into 75% training and 25% testing.

The training dataset will be used in the way we saw when we trained our Louie classifier; forward propagate an image through the network, generate an output, assess the loss, backward propagate the loss back through the network to update weights.
The validation dataset will be used to assess performance on new data using a technique that human learning cannot. Validation data is fed through the network to generate an output, but the network does not learn anything from the data. Loss is reported but the model itself is unchanged! We can use the same validation dataset to assess our performance on new data over and over again while continuing to treat it like new data

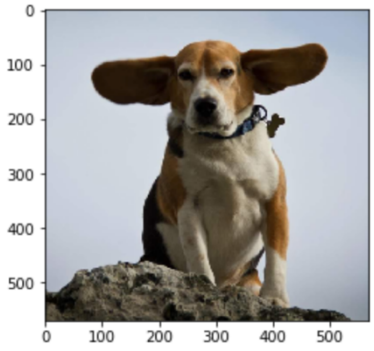
Next we are training alexnet .
Start Digits -> Create new classification model. Train the model for 5 epochs.

We can use this graph to evaluate model performance over the epochs.



Next we send a image to find out the result.

```
Input image:
```




```
Output label:n02088364 beagle
```

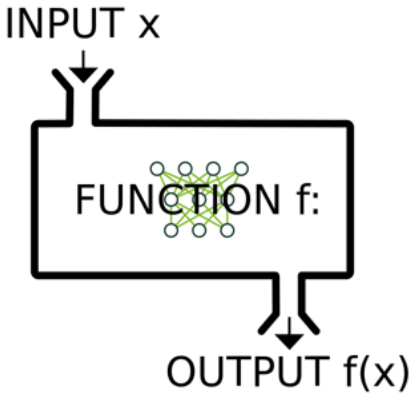
GPU TASK3:

In this task we learn how to deploy the models.

Our problem statement is defined as Louie wants to be able to enter and exit this house since his roommate Nala hunts. Nala needs to be checked at the door.

In this task we will learn :

1. How to deploy a trained model into an application.
2. What role a trained model plays within an application.
3. To identify and use the pieces of a trained model



Deployment is generally providing a model with some input and let the end user read some output.

First in MODEL_JOB_DIR we give the directory of our dataset/model.

Our model generally consists of 2 files
1) Architecture
2) Weights

Next, we need to make sure that the program that we're building can both read and process those files. For this basic type of deployment, we'll need to install (or include) the framework that they were written in to be able to interpret them. We'll learn to deploy to environments that don't require installing the framework later in this course. We'll also need to use the GPU to take advantage of parallel processing. Again, our model consists of hundreds of thousands of operations that can be largely accelerated through parallelization.

Forward Propogation :
`prediction = net.predict([grid_square])`.

We use this function to predict the classes of output image.It pases an input and returns an output.

Generating an Useful output:

```
In [ ]: ##Create an input our network expects
input_image= caffe.io.load_image('/dli/data/fromnest.PNG')
input_image=cv2.resize(input_image, (256, 256), 0,0)
ready_image = input_image-mean_image
##Treat our network as a function that takes an input and generates an output
prediction = net.predict([ready_image])
print("Input Image:")
plt.imshow(input_image)
plt.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"
```

Using this code we've built a simulator which takes the input imaged checks whether dog or cat and then decides whether it's a cat or dog then welcomes it if it's a dog but restricts if it's a cat.

GPU TASK 4:

Untill this point we have learnt:
1) Prepeare the dataset
2) Selected a network to train
3) Trained the network
4). Tested the model

In this mostly I learned about:
To run more epochs on the and to search the hyperparameter space, analogous to human leaner responding differently to a different style.

The importance of learning rate:
Learning rate is the rate at which each "weight" changes during training. Each weight is moving in the direction that reduces loss at a value multiplied by the learning rate.
In this experiment we use two learning rates

1) 5 epoch 0.01(step down) 79% accuracy
2) 5 epoch 0.0001 (fix) 82% accuracy

We start from a pretrained model
Accuracy increases as increasing epochs
The four important aspects of your model performance are

1) Data
2) Hyperparameters
3) Training Time
4) Network Architecture

This section, you'll learn to deploy other people's networks so that you can get the performance gains of their research, compute time, and data curation.
We load the image do some data processing like create a data transformer. And then run the function and visualize the output

In [ ]:
```
#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) pixel values

# 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 dimension
transformer.set_mean('data', mu)            # subtract the dataset-mean value in each channel
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 can also change it later, e.g., for
net.blobs['data'].reshape(1,        # batch size
                          3,         # 3-channel (BGR) images
                          227, 227)  # image size is 227x227

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

Run the function and visualize the output.

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

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

output
```

GPU TASK :5

We use Alexnet to do object detection problem.

Our first method of solving an object detection problem will be to combine an image classification network with traditional programming to create the input/output pairing that we want.

We use the "sliding window" approach where we take split out image into small sections which we'll call grid squares. We're run each grid square through an image classifier. If that grid square contains an image of a dog, we'll have localized Louie in the image.

From there we use deep learning for more and more of your workflow, learning to adjust neural network architecture and finally, a completely new object detection specific workflow.

1) First we load the alexnet

2) Iterate over the given image to get squares of matching size of training image size and put them into the classifier.

3) Suqares with highest probability to bne certain object class are location for the target option.

4) Then we customize the alexnet and feedforward the image of any size to new classifier

5) We have build a complex deep learning network for object detection. We try it by passing an image.

In assessment we have to build train and deploy a model to identify a picture whetehjr it is a whale or not.

1) Train a model
2) New Data as a goal
3) Deployment

Suggestions:

- Use empty code blocks to find out any informantion necessary to solve this problem: eg: `!ls [directorypath] prints the files in a given directory`
- Executing the first two cells below will run your python script with test images, the first should return "whale" and the second should return "not whale"

Start in DIGITS.

In [8]:
```
!python submission.py '/dli/data/whale/data/train/face/w_1.jpg'  #This should return "whale" at the very bottom
```

```
I0214 23:03:17.489045   271 net.cpp:1137] Copying source layer relu2 Type:ReLU #blobs=0
I0214 23:03:17.489058   271 net.cpp:1137] Copying source layer norm2 Type:LRN #blobs=0
I0214 23:03:17.489065   271 net.cpp:1137] Copying source layer pool2 Type:Pooling #blobs=0
I0214 23:03:17.489070   271 net.cpp:1137] Copying source layer conv3 Type:Convolution #blobs=2
I0214 23:03:17.489650   271 net.cpp:1137] Copying source layer relu3 Type:ReLU #blobs=0
I0214 23:03:17.489681   271 net.cpp:1137] Copying source layer conv4 Type:Convolution #blobs=2
I0214 23:03:17.490137   271 net.cpp:1137] Copying source layer relu4 Type:ReLU #blobs=0
I0214 23:03:17.490152   271 net.cpp:1137] Copying source layer conv5 Type:Convolution #blobs=2
I0214 23:03:17.490432   271 net.cpp:1137] Copying source layer relu5 Type:ReLU #blobs=0
I0214 23:03:17.490447   271 net.cpp:1137] Copying source layer pool5 Type:Pooling #blobs=0
I0214 23:03:17.490453   271 net.cpp:1137] Copying source layer fc6 Type:InnerProduct #blobs=2
I0214 23:03:17.513213   271 net.cpp:1137] Copying source layer relu6 Type:ReLU #blobs=0
I0214 23:03:17.513245   271 net.cpp:1137] Copying source layer drop6 Type:Dropout #blobs=0
I0214 23:03:17.513252   271 net.cpp:1137] Copying source layer fc7 Type:InnerProduct #blobs=2
I0214 23:03:17.523456   271 net.cpp:1137] Copying source layer relu7 Type:ReLU #blobs=0
I0214 23:03:17.523491   271 net.cpp:1137] Copying source layer drop7 Type:Dropout #blobs=0
I0214 23:03:17.523497   271 net.cpp:1137] Copying source layer fc8 Type:InnerProduct #blobs=2
I0214 23:03:17.523526   271 net.cpp:1129] Ignoring source layer loss
whale
```

In [9]:
```
!python submission.py '/dli/data/whale/data/train/face/w_1.jpg'  #This should return "not whale" at the very bottom
```

```
I0214 23:03:58.970263   286 net.cpp:1137] Copying source layer relu2 Type:ReLU #blobs=0
I0214 23:03:58.970278   286 net.cpp:1137] Copying source layer norm2 Type:LRN #blobs=0
I0214 23:03:58.970288   286 net.cpp:1137] Copying source layer pool2 Type:Pooling #blobs=0
I0214 23:03:58.970300   286 net.cpp:1137] Copying source layer conv3 Type:Convolution #blobs=2
I0214 23:03:58.970885   286 net.cpp:1137] Copying source layer relu3 Type:ReLU #blobs=0
I0214 23:03:58.970901   286 net.cpp:1137] Copying source layer conv4 Type:Convolution #blobs=2
I0214 23:03:58.971334   286 net.cpp:1137] Copying source layer relu4 Type:ReLU #blobs=0
I0214 23:03:58.971349   286 net.cpp:1137] Copying source layer conv5 Type:Convolution #blobs=2
I0214 23:03:58.971657   286 net.cpp:1137] Copying source layer relu5 Type:ReLU #blobs=0
I0214 23:03:58.971673   286 net.cpp:1137] Copying source layer pool5 Type:Pooling #blobs=0
I0214 23:03:58.971683   286 net.cpp:1137] Copying source layer fc6 Type:InnerProduct #blobs=2
I0214 23:03:58.994503   286 net.cpp:1137] Copying source layer relu6 Type:ReLU #blobs=0
I0214 23:03:58.994539   286 net.cpp:1137] Copying source layer drop6 Type:Dropout #blobs=0
I0214 23:03:58.994544   286 net.cpp:1137] Copying source layer fc7 Type:InnerProduct #blobs=2
I0214 23:03:59.004709   286 net.cpp:1137] Copying source layer relu7 Type:ReLU #blobs=0
I0214 23:03:59.004739   286 net.cpp:1137] Copying source layer drop7 Type:Dropout #blobs=0
I0214 23:03:59.004748   286 net.cpp:1137] Copying source layer fc8 Type:InnerProduct #blobs=2
I0214 23:03:59.004787   286 net.cpp:1129] Ignoring source layer loss
not whale
```

In [ ]: