

# DEEP LEARNING FUNDAMENTALS FOR COMPUTER VISION

## Deep Learning:

It is a subset of machine learning in which we have algorithms that are inspired by the human brain which allows us to use learning strategies that are inspired by human learning. The strides happening in deep learning often come from reflection of human acquisition. Deep Learning breakthroughs drive through AI boom. Examples of Deep Learning are: Object Classification and Detection of images. This task requires the classification of objects within a photograph as one of a set of previously known objects. State-of-the-art results have been achieved on benchmark examples of this problem using very large convolutional neural networks using Alexnet. A more complex variation of this task called object detection involves specifically identifying one or more objects within the scene of the photograph and drawing a box around them.

Solving problems with Deep Learning requires identifying some pattern in the world, finding examples that highlight both sides of the pattern (the input and the output), and then letting a "neural network" learn the map between the two. This opens the types of problems where computers can help us to those where we: 1) Have identified a pattern within a problem 2) Have enough data that exemplifies the pattern

## Deep Neural Networks: GPU Task 1:

1. To choose a dataset that is designed for image classification. The dataset **Images of Beagles** contains 8 labeled images of Louie(dog) and 8 labeled images of other dogs.
2. In step two, we choose a neural network that was designed for image classification.
3. We use **AlexNet** which is architected after the human visual cortex and has won the largest image recognition challenge in the world, ImageNet. Access to high-performing networks like AlexNet remove the requirement that every deep learning practitioner understands the structure of the brain and instead can focus on training.

### GPU Task 1 Insights of Model 1:

- In GPU task 1, we performed image recognition to predict images of Louie(dog).
- In the first model, we trained a model with 5 epochs, which gave an accuracy of 50.29% of images predicted that the images were of Louie.
- 49.71% of images predicted that the images were of not the dog Louie.

### GPU Task 1 Insights of Model 2:

- In GPU task 2, we performed image recognition to predict images of Louie(dog).
- In the second model, we trained a model with 100 epochs, which gave an accuracy of 100% of images being predicted that the images were of Louie.
- Along with that accuracy, the learning rate of the model was changed for training the model to get better predictions

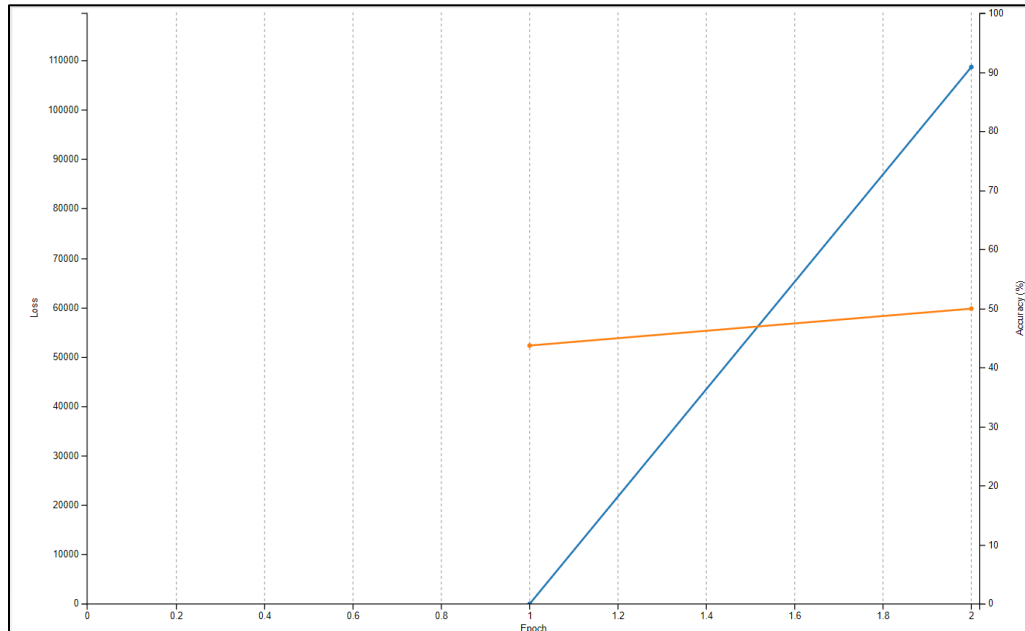
# First Task: Image Classification

Model\_1:

Epoch: 5

Learning Rate: 0.0001

Standard Network: AlexNet



Predictions	
Louie	50.29%
Not Louie	49.71%

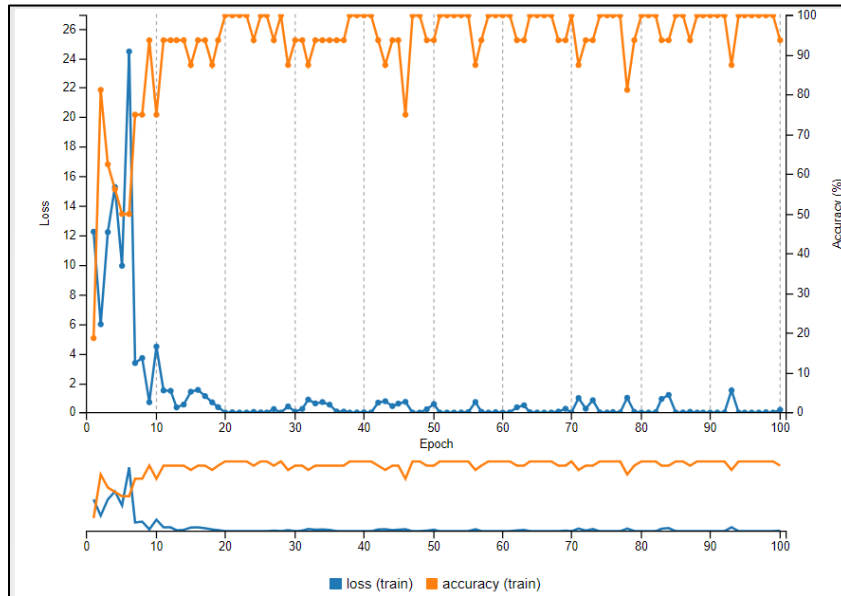
**Comments:** By keeping above hyper-parameters, we can get only 50.29% accuracy which is very low. As mentioned in the certification program, Now I am increasing the number of epochs to 100 and see the results.

Model\_2:

Epoch: 100

Learning Rate: 0.0001

Standard Network: AlexNet



Predictions	
Louie	100.0%
Not Louie	0.0%

**Comments:** By increasing the number of epochs to 100, We got 100% accuracy. Which means the model can predict exactly which is Louie image.

## Deep Neural Networks: GPU Task 2:

A neural network changes when exposed to data to create an accurate map between inputs and outputs. In the Louie example, our model was effective on data from our dataset, but not on any new data, rendering it almost useless in the real world. In this task, we use a new dataset named Images of Dogs and Cats. This dataset contains images of both cats and dogs unlike the first task which had images of only dogs. We make use of Alexnet which we used in the first dataset. Let's train the model to test the accuracy.

### GPU Task 2 Insights of Model 1:

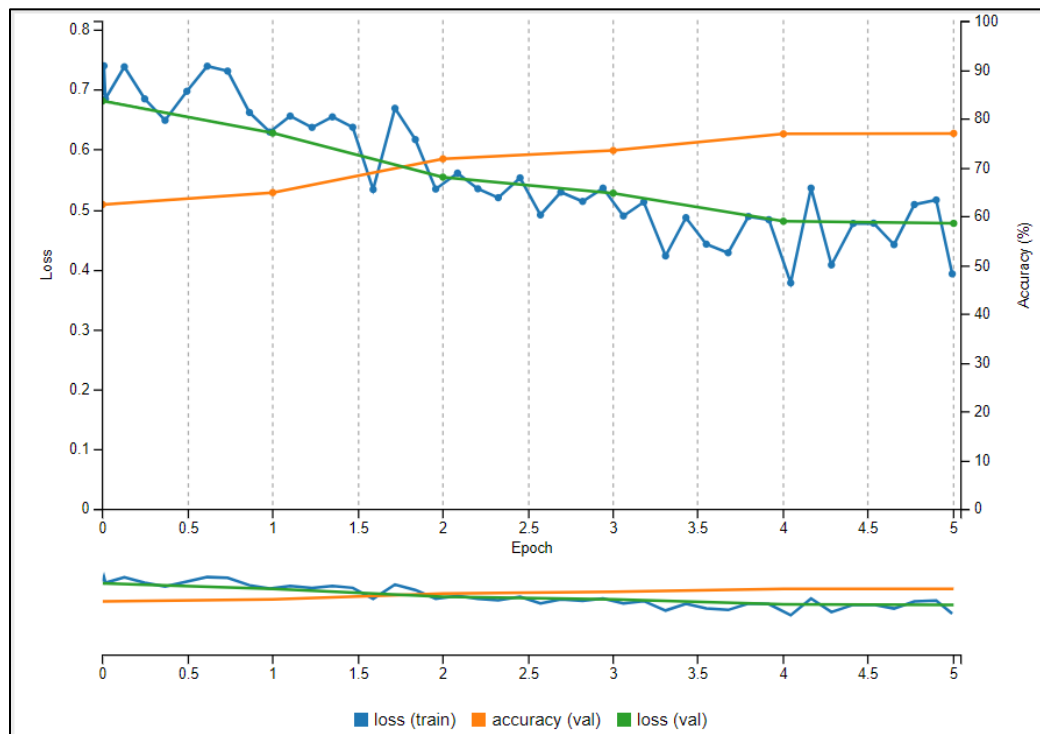
- In GPU task 1, we performed image recognition to predict images of Louie(dog).
- In the first model, we trained a model with 5 epochs, which gave an accuracy of 50.29% of images predicted that the images were of Louie.
- 81.01% of images predicted that the images were of not the dog Louie.
- This tells us that the model is somewhat overfitting.

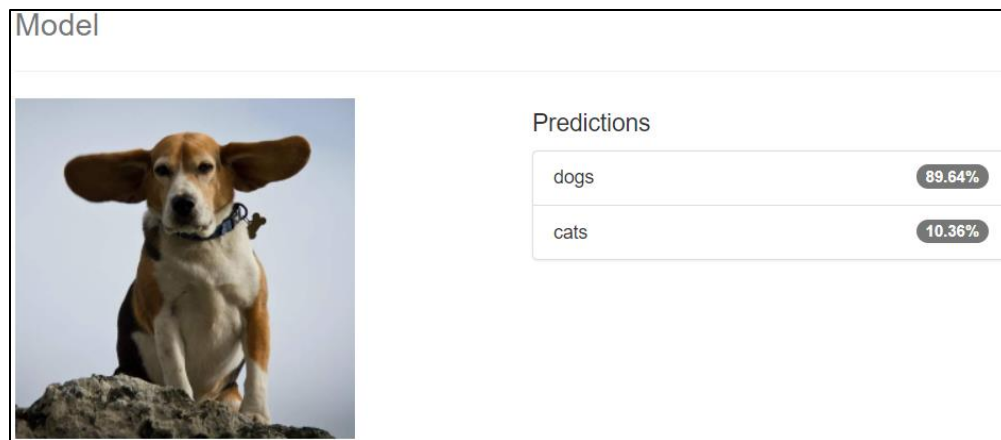
### GPU Task 2 Insights of Model 2:

- In GPU task 2, we performed image recognition to predict images of Louie(dog).
- In the second model, we trained a model with 100 epochs, which gave an accuracy of 100% of images being predicted that the images were of Louie.
- Along with that accuracy, the learning rate of the model was changed for training the model to get better predictions

## Second Task: Image Classification

### Model 1:





For model 1 the following parameters were used:

Epoch: 5

Learning Rate: 0.01

Standard Network: AlexNet

### Model 2:

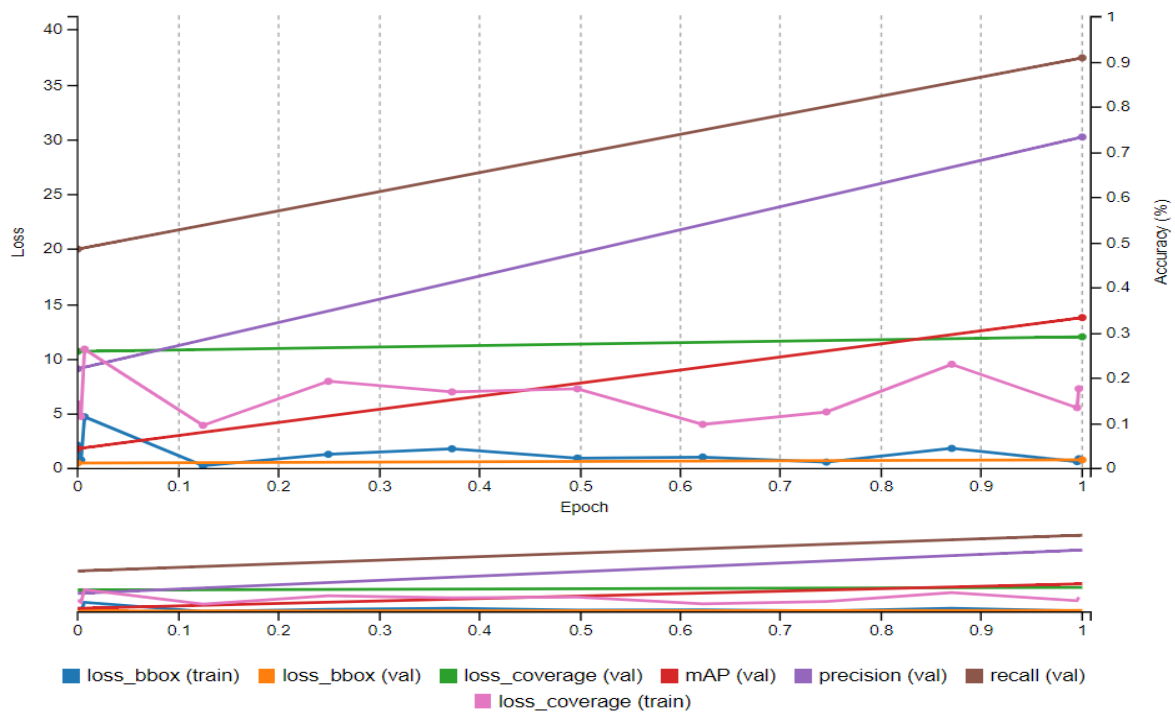
Epoch: 5

Learning Rate: 0.001

Standard Network: AlexNet

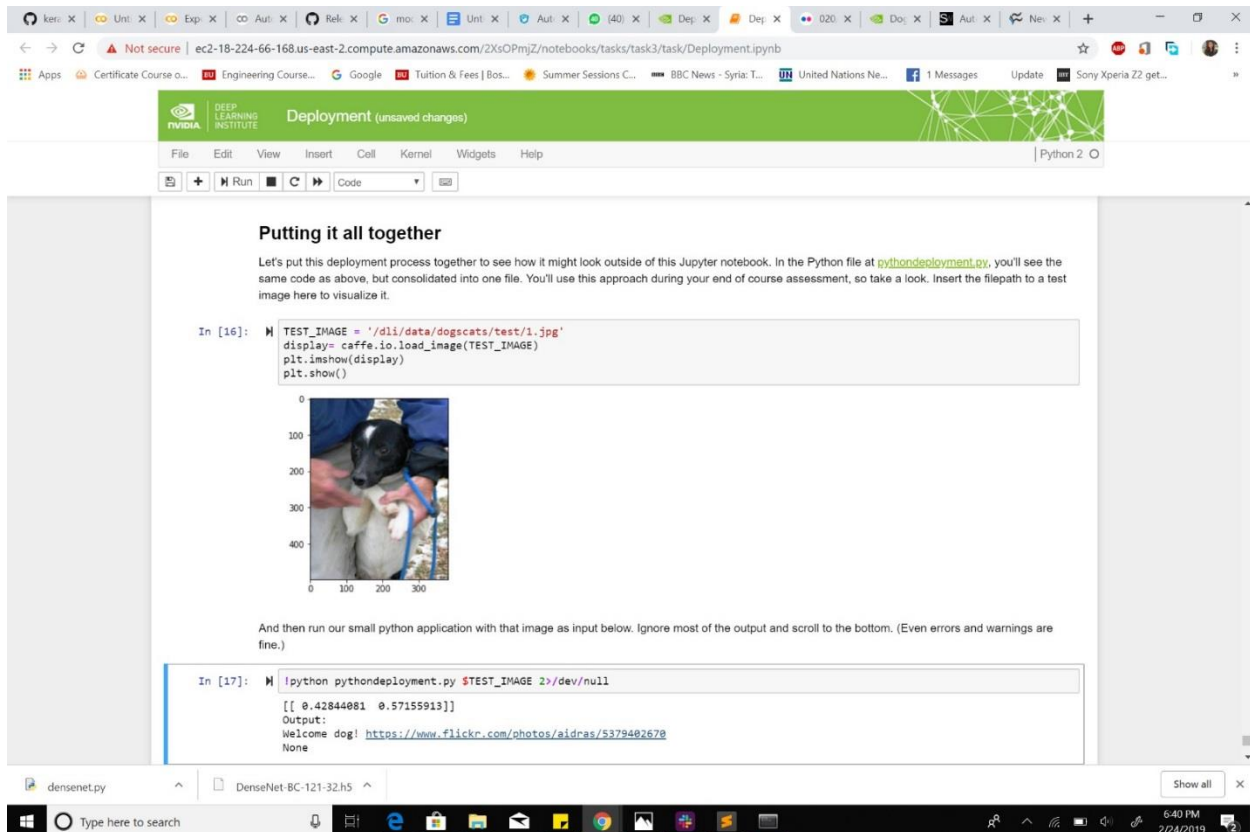
Validation size: 6250

Training size: 18750



## Task 3: Deploying pretrained model

- In this task, we make use of the pretrained model which consists of the images of dogs and cats.
- This dataset has a training size of 18750 and a validation size of 6250. We are using the pretrained model which is deployed by making use of model architecture and weights.
- For the previous tasks we have trained a classification model on a smaller dataset and another model on larger dataset and with more number of epochs.
- The objective of this task is to deploy the model we trained previously. The model was used for deployment had weights and architecture of Dog vs cats model.
- We created a simulation to detect if a dog is present at the dog door or not.



The screenshot shows a Jupyter Notebook interface with the title "Deployment (unsaved changes)". The notebook is running on a Python 2 kernel. The first code cell, labeled "Putting it all together", contains the following code:

```
In [16]: TEST_IMAGE = '/dli/data/dogscats/test/1.jpg'
display= caffe.io.load_image(TEST_IMAGE)
plt.imshow(display)
plt.show()
```

The output of this code cell is a plot showing a black and white image of a dog's head. The plot has x and y axes ranging from 0 to 300.

Below the plot, the text reads: "And then run our small python application with that image as input below. Ignore most of the output and scroll to the bottom. (Even errors and warnings are fine.)"

The second code cell, labeled "In [17]:", contains the following code:

```
In [17]: !python pythondeployment.py $TEST_IMAGE 2>/dev/null
```

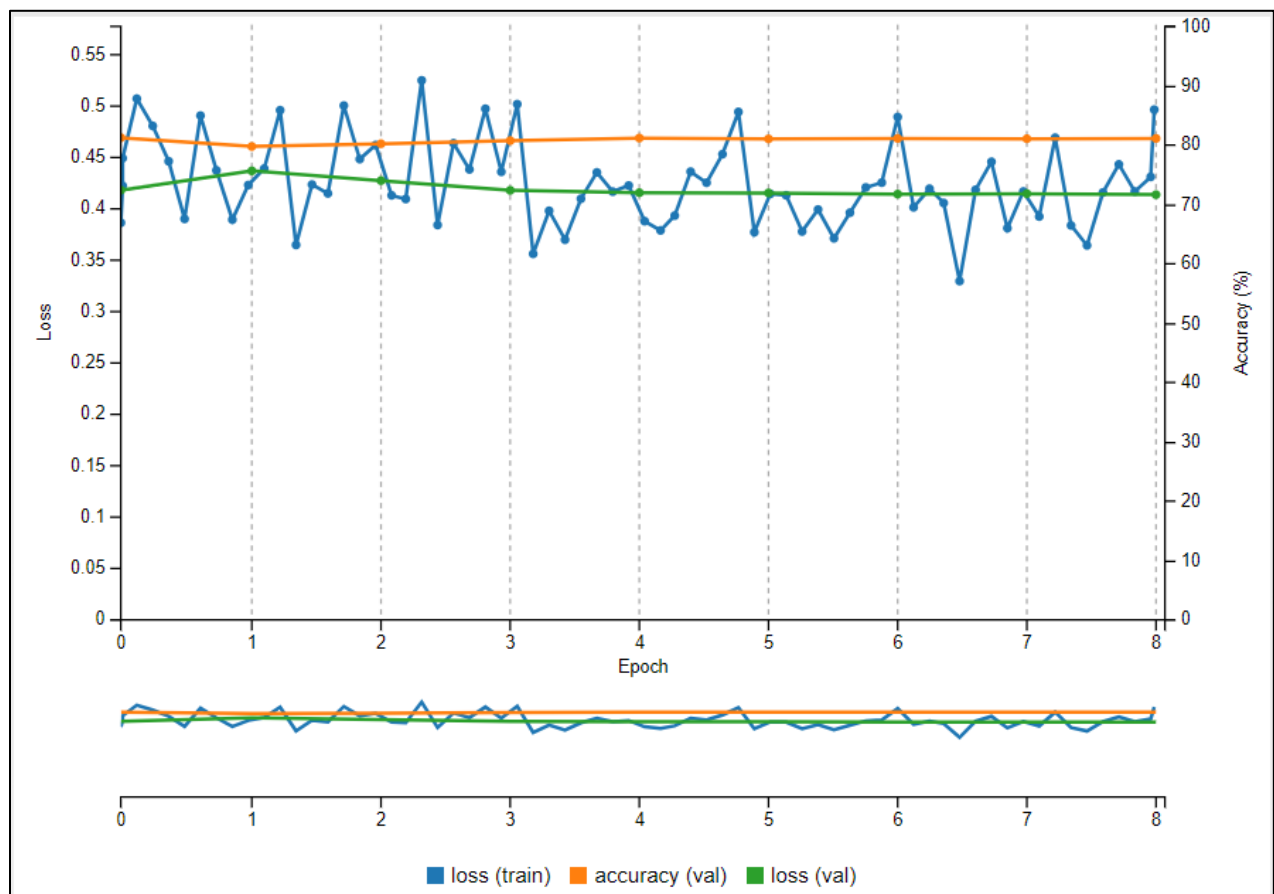
The output of this code cell is:

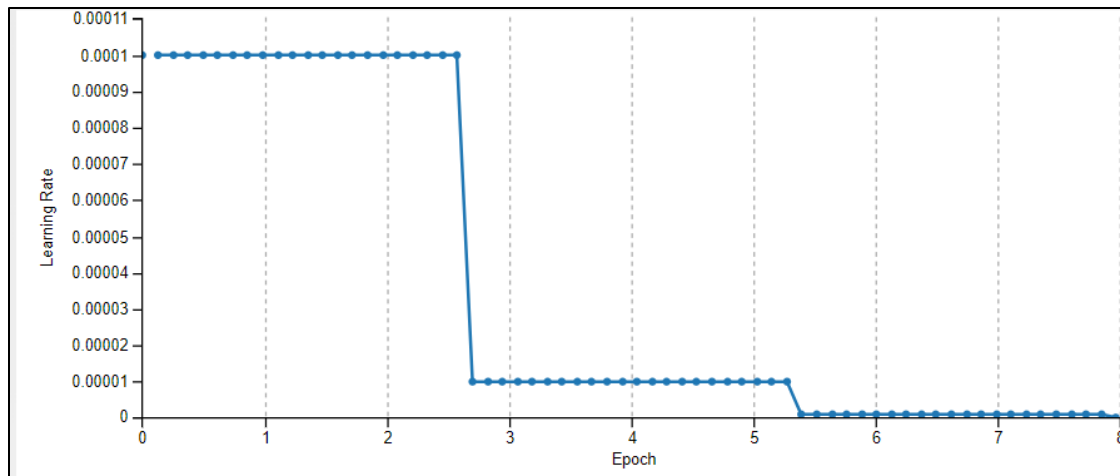
```
[[ 0.42844081  0.57155913]]
Output:
Welcome dog! https://www.flickr.com/photos/aidras/5379402670
None
```

## Task 4: Performance during GPU task

For model 1 we are using pretrained model for cats Vs dogs:

- Prepared a dataset for training
- Selected a network to train
- Trained the network
- Tested the trained model in a training environment
- Deployed the trained model into an application
- Epochs: 8
- Learning Rate= 0.0001





## Task 5: Object Detection

- Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos.
- That fact allowed us to build a simple application that harnessed deep learning. In the assessment, we apply the same technique to build something more complex.
- We implement deep learning through the task of image classification. In this section, we'll augment the dataset to expand your definition of deep learning to guide you to see what types of problems can be solved with deep learning and what types can't.
- With image classification, our network took an *image* and generated a *classification*. More specifically, our input was a 256X256X3 *tensor* of pixel values and our output was a 2-unit vector (since we had 2 classes) of probabilities.

## Object Detection Assessment

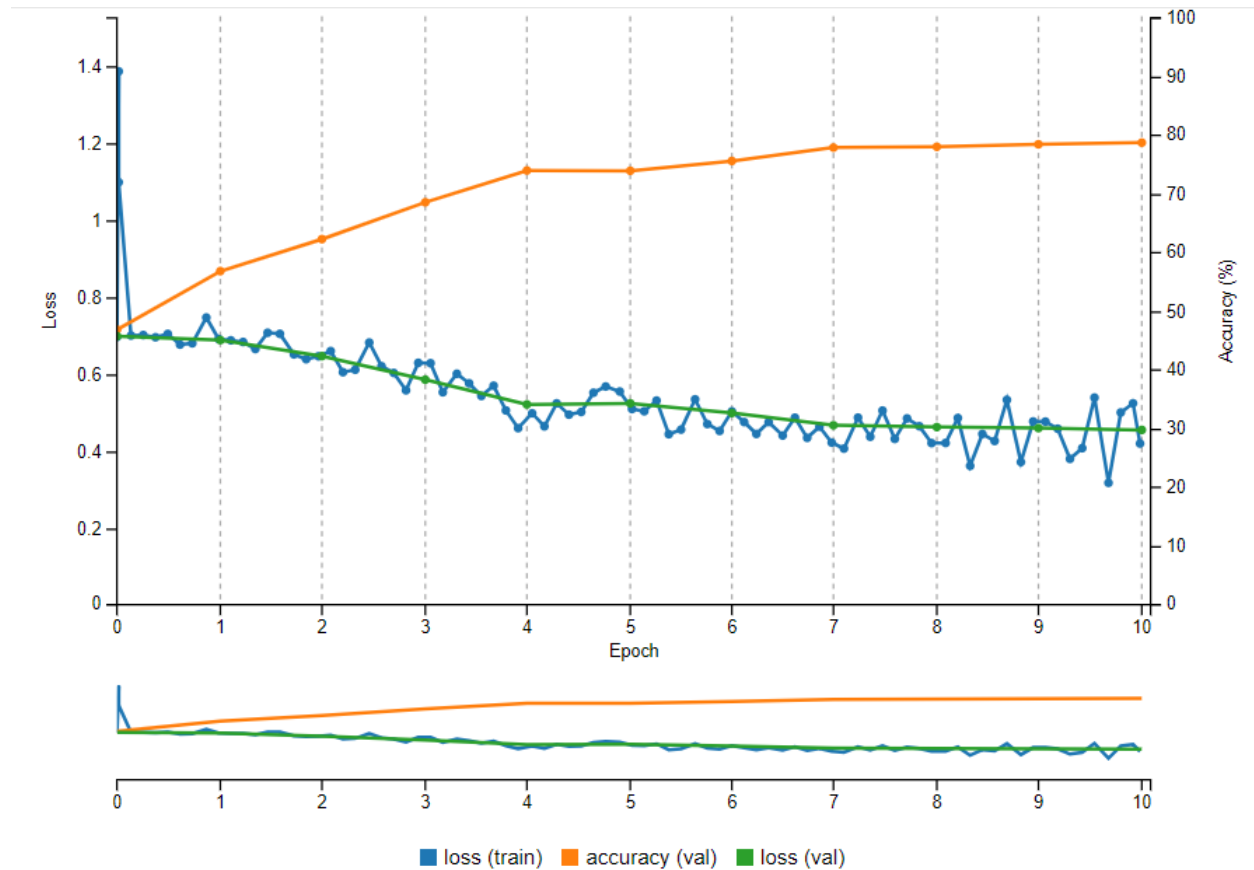
- In this task, we used a 'sliding window' approach, where we took the small pieces of images which they called grid squares.
- Each grid square ran through an image classifier. The objective was to find Louie in the image. We combined deep learning with traditional computer vision and modified the internal of neural networks.
- We use a total of 3 approaches in this task to perform image detection. In the first approach, we use the traditional network.
- The second approach is rebuilding the existing neural network. The new network was built on the previous network.
- The new network was formed by changing the layer structure from previous AlexNet architecture. DetectNets are used in the third approach. When approach two and three are



visualized we can say that the DetectNet is a Fully convolutional network which is configured to produce precise data representations.

- The bulk layers in DetectNet is identical to the GooGLeNet Network. The model was pretrained for 16 hours on NVIDIA Tesla K80.
- Since the model was well trained the DetectNets successfully detected the dog and drew a bounding box around it.

### Model\_1



### Object detection Predictions:

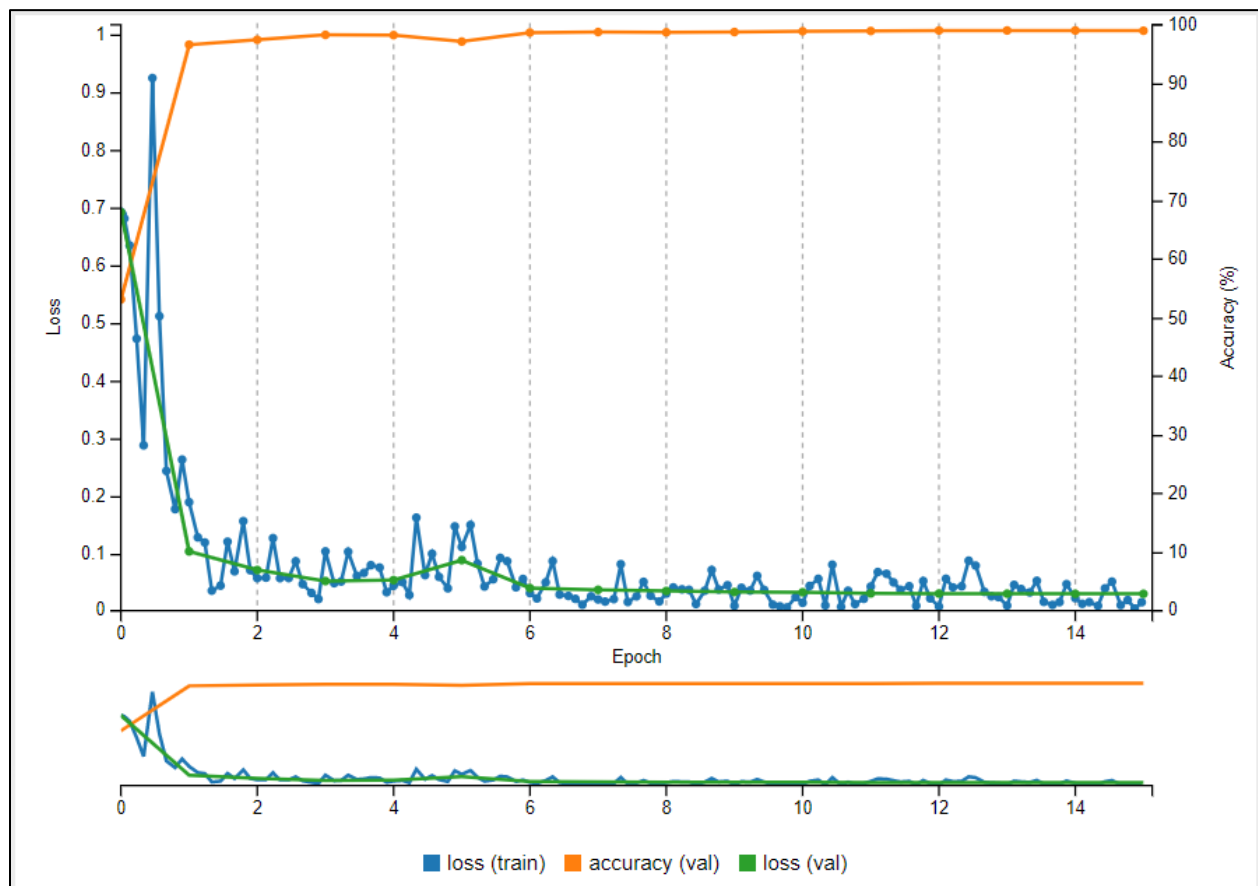


## GPU Task 6: Assessment Task:

In the final task of this certification, we are supposed to build a neural network for the given dataset. That data set consists of 2 class of images. We need to build a model whose accuracy should be greater than 80%.

Epoch= 15

Learning Rate= 0.001



When we built a neural network for the classification problem in the assessment, the accuracy predicted was almost 100%. which is very good accuracy and it predicts accurately.

The screen shot of classifier which yields the result for the input image:

```
I0224 21:17:08.781922 216 net.cpp:1137] Copying source layer pool2 Type:Pooling #blobs=0
I0224 21:17:08.781927 216 net.cpp:1137] Copying source layer conv3 Type:Convolution #blobs=2
I0224 21:17:08.782362 216 net.cpp:1137] Copying source layer relu3 Type:ReLU #blobs=0
I0224 21:17:08.782377 216 net.cpp:1137] Copying source layer conv4 Type:Convolution #blobs=2
I0224 21:17:08.782708 216 net.cpp:1137] Copying source layer relu4 Type:ReLU #blobs=0
I0224 21:17:08.782723 216 net.cpp:1137] Copying source layer conv5 Type:Convolution #blobs=2
I0224 21:17:08.782976 216 net.cpp:1137] Copying source layer relu5 Type:ReLU #blobs=0
I0224 21:17:08.782990 216 net.cpp:1137] Copying source layer pool5 Type:Pooling #blobs=0
I0224 21:17:08.782995 216 net.cpp:1137] Copying source layer fc6 Type:InnerProduct #blobs=2
I0224 21:17:08.801208 216 net.cpp:1137] Copying source layer relu6 Type:ReLU #blobs=0
I0224 21:17:08.801246 216 net.cpp:1137] Copying source layer drop6 Type:Dropout #blobs=0
I0224 21:17:08.801259 216 net.cpp:1137] Copying source layer fc7 Type:InnerProduct #blobs=2
I0224 21:17:08.809172 216 net.cpp:1137] Copying source layer relu7 Type:ReLU #blobs=0
I0224 21:17:08.809201 216 net.cpp:1137] Copying source layer drop7 Type:Dropout #blobs=0
I0224 21:17:08.809206 216 net.cpp:1137] Copying source layer fc8 Type:InnerProduct #blobs=2
I0224 21:17:08.809231 216 net.cpp:1129] Ignoring source layer loss
whale

In [7]: !python submission.py '/dli/data/whale/data/train/not_face/w_1.jpg' #This should return "not whale" at the very bottom
I0224 21:17:18.346844 231 net.cpp:1137] Copying source layer relu2 Type:ReLU #blobs=0
I0224 21:17:18.346858 231 net.cpp:1137] Copying source layer norm2 Type:LRN #blobs=0
I0224 21:17:18.346864 231 net.cpp:1137] Copying source layer pool2 Type:Pooling #blobs=0
I0224 21:17:18.346869 231 net.cpp:1137] Copying source layer conv3 Type:Convolution #blobs=2
I0224 21:17:18.347314 231 net.cpp:1137] Copying source layer relu3 Type:ReLU #blobs=0
I0224 21:17:18.347329 231 net.cpp:1137] Copying source layer conv4 Type:Convolution #blobs=2
I0224 21:17:18.347668 231 net.cpp:1137] Copying source layer relu4 Type:ReLU #blobs=0
I0224 21:17:18.347682 231 net.cpp:1137] Copying source layer conv5 Type:Convolution #blobs=2
I0224 21:17:18.347918 231 net.cpp:1137] Copying source layer relu5 Type:ReLU #blobs=0
I0224 21:17:18.347932 231 net.cpp:1137] Copying source layer pool5 Type:Pooling #blobs=0
I0224 21:17:18.347939 231 net.cpp:1137] Copying source layer fc6 Type:InnerProduct #blobs=2
I0224 21:17:18.365532 231 net.cpp:1137] Copying source layer relu6 Type:ReLU #blobs=0
I0224 21:17:18.365564 231 net.cpp:1137] Copying source layer drop6 Type:Dropout #blobs=0
I0224 21:17:18.365571 231 net.cpp:1137] Copying source layer fc7 Type:InnerProduct #blobs=2
I0224 21:17:18.373425 231 net.cpp:1137] Copying source layer relu7 Type:ReLU #blobs=0
I0224 21:17:18.373458 231 net.cpp:1137] Copying source layer drop7 Type:Dropout #blobs=0
I0224 21:17:18.373463 231 net.cpp:1137] Copying source layer fc8 Type:InnerProduct #blobs=2
I0224 21:17:18.373495 231 net.cpp:1129] Ignoring source layer loss
not whale
```

Hence, the model was trained and deployed successfully.