GPU Task 1 Concept

Train an image classification network from pre-loaded small data set - to teach a neural network to understand the difference between Louie and other dogs

Performance:

1. Select dataset - 8 labeled images of Louie and 8 labeled images of other dogs

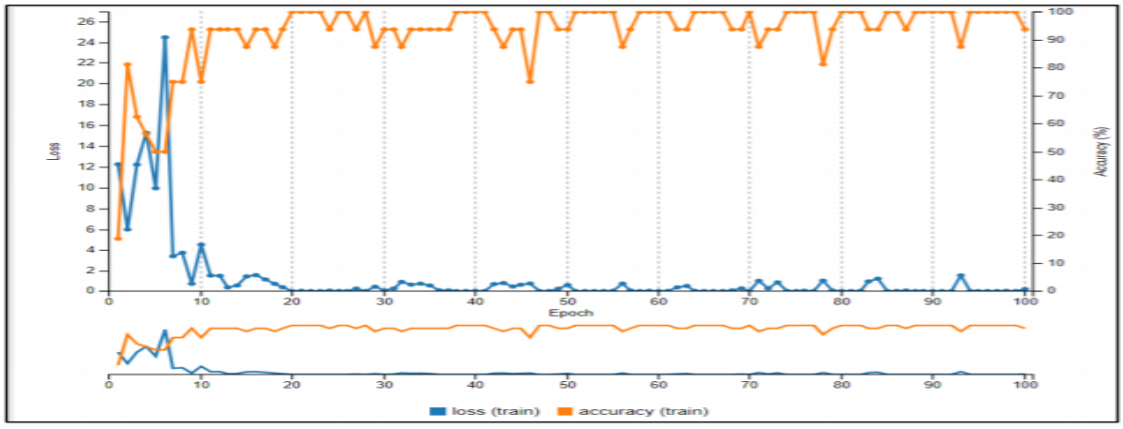
2. Set number of epochs for training 2/100

3. Set network configuration • AlexNet (AlexNet is intended to be used with 256X256 (color) images)

4. Train

5. Test Louie's image from training dataset to check if network could identify Louie

A



Observation and Conclusion:

1. After 100 epoch training, Louie classifier are more accurate to classify Louie vs. non-Louie.

2. Training time is very important to train a neural network.

GPU Task 2

The purpose of this task was to learn about how validation data can be used to monitor a model's performance with new data while it trains. In this task I trained a neural network to detect dogs versus cats. First the data has to be preprocessed to a size compatible with the AlexNet, and organized into folders where the folder name is the category label, and with 25% of the data set aside as a validation set. Then I trained AlexNet for 5 epochs. The validation data had above 80% accuracy after the 5 epochs as shown in Figure 2, and it categorized a picture of a dog correctly with greater than 90% probability.

Configuration	AlexNet	GoogLenet
Number of Epochs	5 Epoch	5 Epoch
Base Learning Rate	0.01	0.01
Training Time	3 minutes, 49 seconds	16 minutes, 7 seconds
Accuracy (validation)	~ 80%	~ 76%
Loss (train)		
Loss (validation)		

Observation and Conclusion:

1. Only 5 epoch with AlexNet configuration can reach an accuracy of 80% for dog/cat classification

2. Validation dataset allow us to asses the loss/accuracy of current trained network with new data

3. Inference is to making decisions based on what was learned - classify unlabeled image

GPU Task 3

Concept: Deploy a trained model into an application - door sensor which only allow dogs.

Performance:

- Prepare the trained model and environment
- Get the filePath of model architecture file (deploy.prototxt) and weights (.caffemodel)
- Use GPU: caffe.set_mode_gpu()
- Build caffe classifier with trained model files

Preprocessing

- Resize the images to 256 X 256. '
- Normalize the images by subtracting the mean image from each image

3. Forward Propagation - use the built classifier and preprocessed data to predict

Post-processing - output the desired output.

Observation and Conclusion:

1. For pre-processing: Whatever was done prior to training must be done prior to inference

2. Preprocessing could be uniform the size according to trained data + normalize by subtracting the mean

GPU Task 4

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

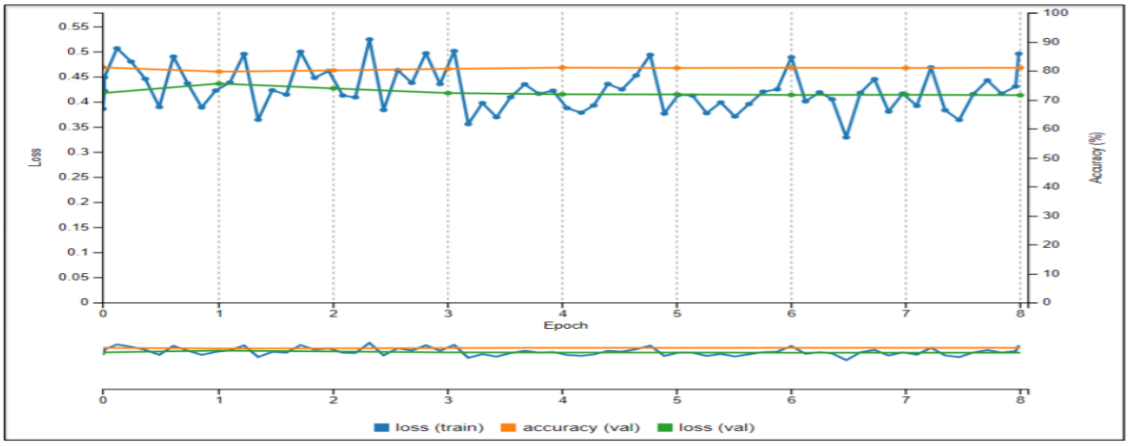
• Prepared a dataset for training

• Selected a network to train

• Trained the network

• Tested the trained model in a training environment

• Deployed the trained model into an application



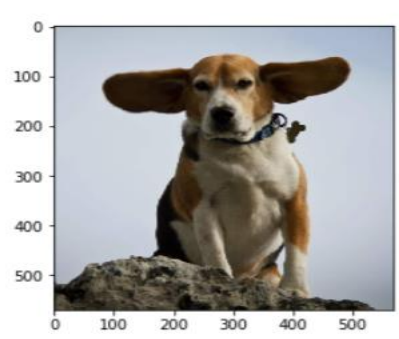


Figure 5: Pretrained AlexNet application output for an image of a beagle

GPU Task 5

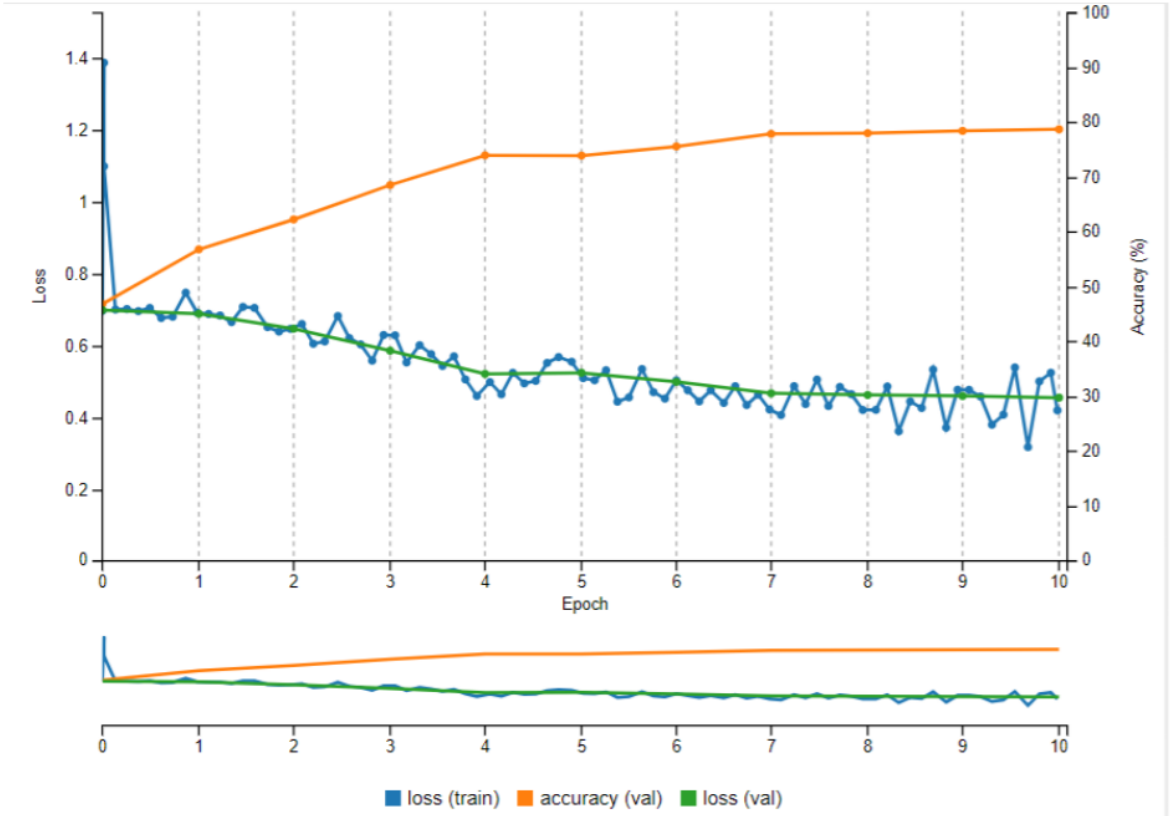
Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos.

• 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.

Model_1



Assessment Results

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

```
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 21:17:18.346858   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
```