Hacettepe University Department of Computer Engineering

Name & Surname	: Baturalp Furkan KARAMUS
Identity Number	: 21527137
Course	: BBM418 - Computer Vision Laboratory
Assignment	: Assignment3
Advisors	: Assoc. Prof. Dr. Nazlı İKİZLER CİNBİŞ, TA. Özge YALÇINKAYA
Due	: 31.05.19
E-mail	: b21527137@cs.hacettepe.edu.tr

1 Introduction

1.1 Problem

In this assignment we simply doing object trackking with regression networks. Train our model with pretrained features and bounding boxes the track the object.

This report contains all the steps involved in the process, as well as how the problems encountered in these stages are optimized and relevant.

2 Features

In this section i use Convolutional Neural Network algorithm for extract features from image. While doing this vgg16 pre trained model used. This is trained by someone else(ImageNet) previously by other projects. We use this model because this model prevent us build our CNN again.

Purpose is apply this model all images in test and train set. Then extract feature vector which is pool5. After that imply average pooling. These are 1024 dimensional vectors and include informations about object location.

.

2.1 Pre Trained VGG-16 Model

Simply put, a pre-trained model is a model created by some one else to solve a similar problem. Instead of building a model from scratch to solve a similar problem, you use the model trained on other problem as a starting point.

For example, if you want to build a self learning car. You can spend years to build a decent image recognition algorithm from scratch or you can take inception model (a pre-trained model) from Google which was built on ImageNet data to identify images in those pictures.

A pre-trained model may not be 100% accurate in your application, but it saves huge efforts required to re-invent the wheel

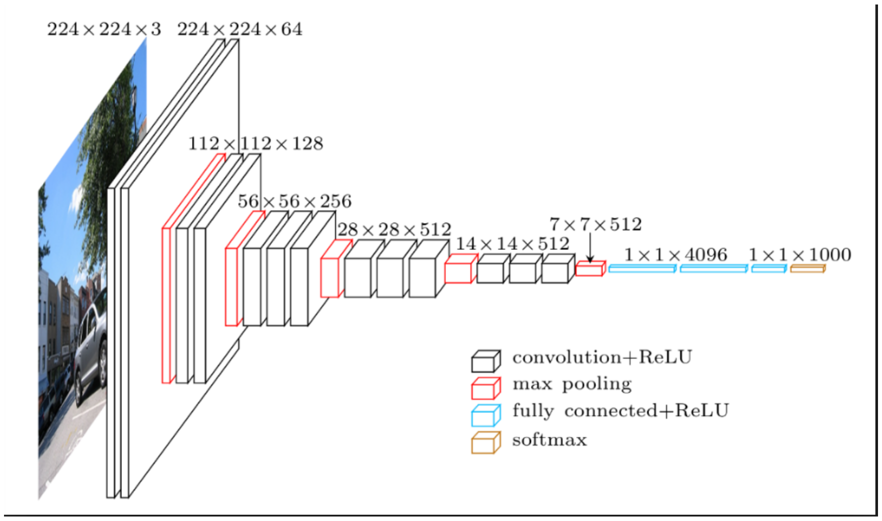


Figure 1: Basically vgg16 pre trained model.

Software Usage

I use basic vgg16 model for extract features. While doing this torch library was used. After that classify images with SVM classifier which implemented with scikit learn library.

- Create new dataset for read frames with pairs and initiliaze transform parameters.
- Create dataloader objects.
- Create vgg16 model which pre-tranined true.
- Take features of model and assign avg pool function.
- Put all images in vgg16 model in evaluation mode then extract features.
- Store all features in array
- Concatenate two features then get 1x1024 array.
- Store these too.
- Create new model which take 1024 dimensional array and return 1x4 coordinate vector.
- Put all these features in this model and get coordinates.
- Use this model for training model with all training and validation features.
- Calculate average loss for each epoch and draw graph
- While reading images crop each image according two 2 times enlarged version of bounding boxes.
- Then update bounding box values acoording to resized and cropped image.

Experiment Results
In this part model is trained with known bounding boxes and loss values good for this part.

When i increase batch size program run faster but loss increase.

Implementation Details

In this part there is some explanation about test steps:

1-Create Dataset:
In this step i create a new dataset. Then define transformations and get-item function. This dataset take an image pair and crop these images with first one's bounding box 2 time enlarged version.Then update coordinates.Then update coordinates again after transform(resize) operation. It return first frame,second frame and coordinates
2-Get features:
Create a vgg16 pretrained model and take it's features.Edit this model and assign average pool function. Put all images in this model and get 1x512 features.Concatenate these two model features and get 1x1024 features.Store these features into array.Do this operation for all dataset.

3 Training

In this part new model created and edit this layers. This model must take 1024 dimensional feature matrix and return 4 dimensional coordinate matrix. We use this model when training process. After edit model put all features in this model and get predicted coordinates. This will give us better results.If bounding box coordinates not update after resize and crop process this model can't train.But i did every calculations about that and these bounding boxes ready to train.

3.1 Convolutional Neural Networks

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery.

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually refer to fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks make them prone to overfitting data. Typical ways of regularization includes adding some form of magnitude measurement of weights to the loss function. However, CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.

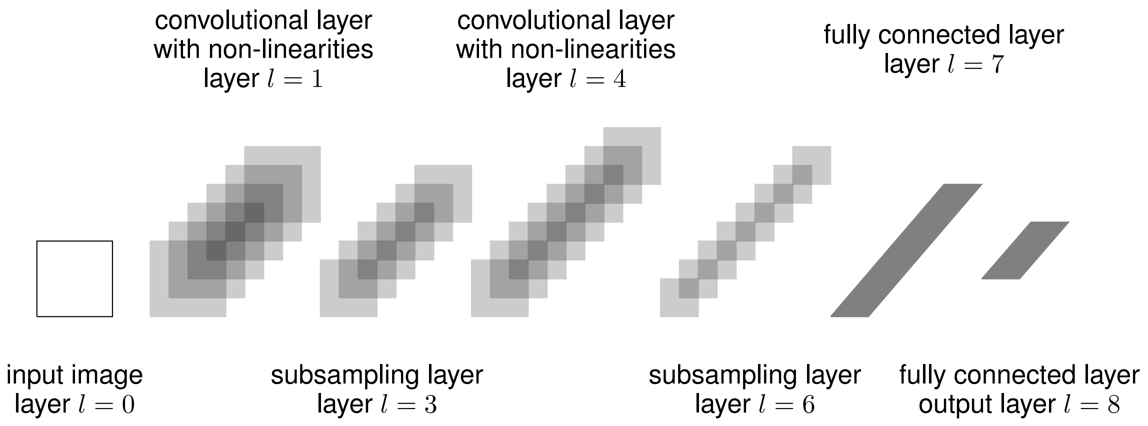
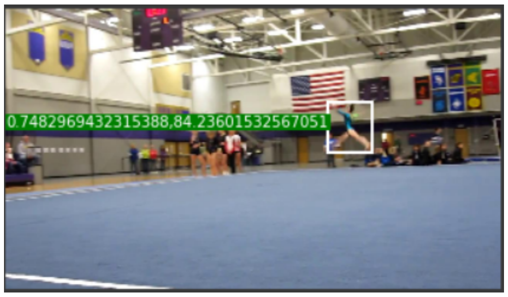

convolutional layer with non-linearities layer $l = 1$
convolutional layer with non-linearities layer $l = 4$
fully connected layer layer $l = 7$
input image layer $l = 0$
subsampling layer layer $l = 3$
subsampling layer layer $l = 6$
fully connected layer output layer $l = 8$

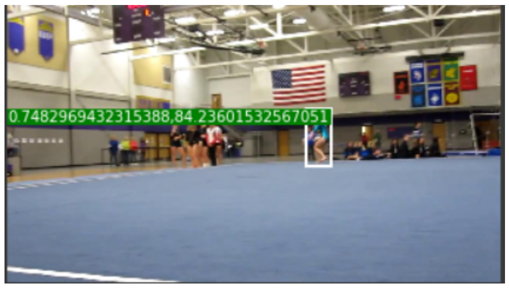
Figure 2: The architecture of the original convolutional neural network, as introduced by LeCun et al. (1989), alternates between convolutional layers including hyperbolic tangent non-linearities and subsampling layers. In this illustration, the convolutional layers already include non-linearities and, thus, a convolutional layer actually represents two layers. The feature maps of the final subsampling layer are then fed into the actual classifier consisting of an arbitrary number of fully connected layers. The output layer usually uses softmax activation functions.

SOFTWARE USAGE:

In this part i will use pre trained model and freeze all layers then start train network. Before starting train process we prepare network for training process. That means we have 10 classes and pre trained network has 1000 classes for last layer. We must change this fc8 layer like 4096x10 dimension. After do that initilaze optimizer. We must use ADAM optimizer in this example. For this optimizer i decide best learning rate value is 0.0001. This gives me best result.Assign criterion as Cross Entropy Loss and assign a scheduler. Then start training. After the training we change model to evaluation mode and evaluate test images response then calculate accuracy for best.

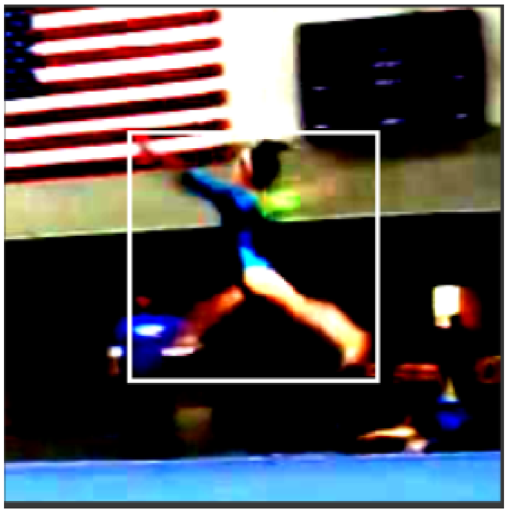


(a) 1a

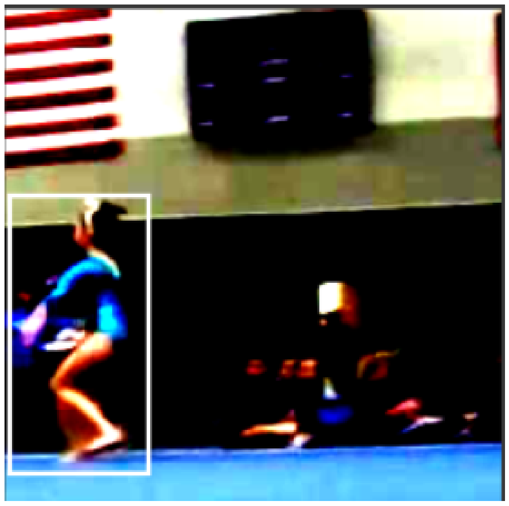


(b) 1b

Figure 3: Image with no resize and crop operation....

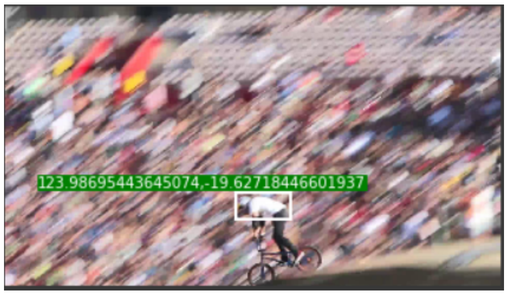


(a) 1a

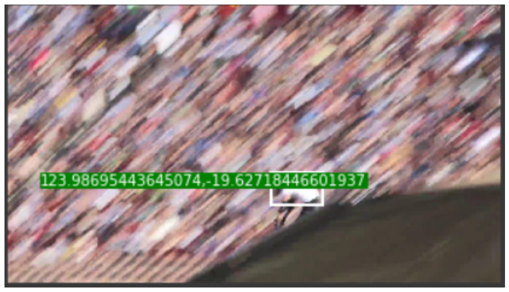


(b) 1b

Figure 4: Image after resize and crop with updated bounding box....

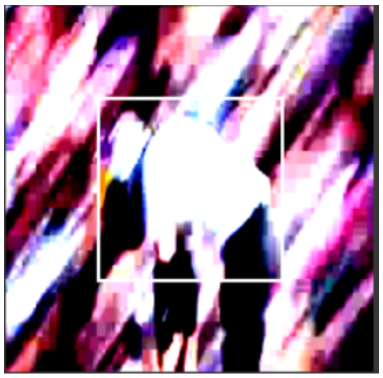


(a) 1a



(b) 1b

Figure 5: Image with no resize and crop operation....



(a) 1a

(b) 1b

Figure 6: Image after resize and crop with updated bounding box....

EXPERIMENT RESULTS:

In this part i will describe training process and test process seperately. In training process train/valid loss graph will be explained. Then comment about overfitting network.

Train Mode

Batch Size:1 Learning Rate:0.0001->

```
Epoch 3/20
----------
Epoch 10 result:
Avg loss (train): 0.0484
Avg loss (val): 0.0439

----------
```

This is the graph for training and validation loss.

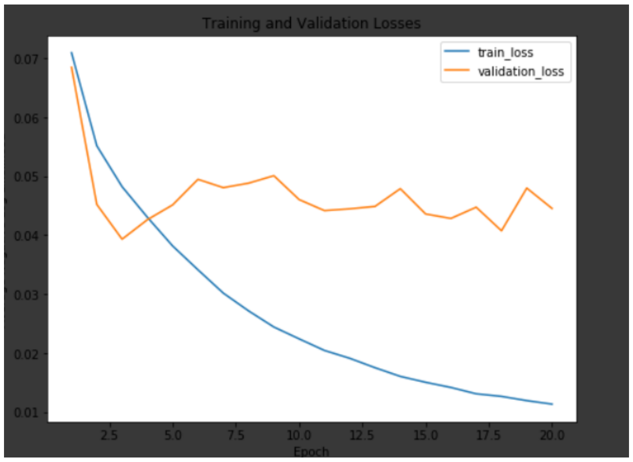


Figure 7: Loss Graph

This graph shows up to train results.Train with batch size 1 .This model have run very fast because there isn't so much layer. This graph shows while network training validation loss in decreasing in some part but not informaly. After 3-4 epoch network starts overfitting that means train mode must stop now but in this experiment i don't implement early stopping because of this graph. Some part valid loss decreasing and increasing so we can't stop immediately at any point. After the 20 epoch i choose best network checkpoint and save them if network getting overfitted it doesn't matter.

After a couple of test i decide best learning rate as 0.0001. We see that in my model while 1 batch size is better than 2 batch size. In contrast to this running time parameter is better than other one.

Implementation Details:

In this part there is some explanation about implementation steps:

- Collect all images in dataloader.
- Create new model and edit it.
- Initiliaze training parameters optimizer(ADAM),scheduler,criterion(MSEloss)...
- Call train model functions.
- Train network with using training and validation images.
- After training is done evaluate model on test images and calculate accuracy.

4 Testing and Evaluation

In this part we evaluate loss and bounding box for all test image in dataset. For each test video initialize the bounding box of the first frame from the ground-truth.Then,network will predict the bounding boxes of the rest.

EXPERIMENT RESULTS:

Table 1: Part 3 results

Part		
Batch Size	Learning Rate	Average Loss
1	0.0001	%0.0469

As we see model training was succesfull but there is some problem in this chapter which i can't figure it out. After finding coordinates from evaluation when i print it on the relatively image. There is totally different from what it must to be. I think it cause about normalize coordinates and images.

This GIF's created after getting result coordinates. After that this coordinates scale by resizing process and get coordinates for bigger image. **You can get my GIF result from this link:** Results I think my GIF results are succesfull because it track the object and keep steady related location. Bounding boxes generally don't appears far away from object.

5 Functions:

NormalizeVector(): Normalize given vector between 0 and 1.

extractFeature(): Extract pool5 features from given pre-trained model and phase.

cropImage(): Crop image with 2 times enlarged bounding box.

evalModel() Evaluate loss about test images on training network.Return coordinates.

trainModel() Train on new created model.

For other process built-in functions used.

6 Conclusion:

In conclusion we understand that learning rate and batch size values are important in transfer learning. We test two batch size and both of them give different result. In situation depends one which one of must be used. When i create firstly this experiment i used MSE as optimizer.Then i used learning rate 0.001 value. But when convert them to the ADAM optimizer this value is too big for it.Then i change learning rate 0.0001 and this is the best option for this optimizer in my experiment.

In this assignment there is a couple important steps about training and evaluation. If i didn't update bounding box coordinates after resize or crop image , this training couldn't be that succesfully.

Other thing that normalization coordinates when i train image, i normalized everything.Images and coordinates after when i get predicted coordinates apply this coordinates on denormalized images and get better result.