

# IE -643 Take-Home Quiz

Roll No.- 19I190008

**Q1 (a) Explain with appropriate reasons if a typical deep convolutional network (e.g. VGGNet, ResNet) trained on data sets like Pascal VOC and ImageNet can or cannot be used for the detection and recognition tasks illustrated in Figures 3.1 and 3.2.**

- Regular deep convolutional network architectures are trained on ImageNet and are suitable for image classification tasks, and can also be used for other computer vision tasks like detection, segmentation using transfer learning techniques. But these alone cannot be used.
- These networks essentially act as **feature extractors**- convolutions are basically filters applied on the image to extract features. The extracted feature maps can include vertical and horizontal edges, shapes, textures, colours etc. in the image.

-These feature maps are then fed into the detection algorithms (R-CNN, YOLO etc.) to facilitate **multiple object recognition and localization**.

Hence, typical networks **cannot be directly used** for the given tasks. Also, for the given tasks, straightaway using the above approach would not be helpful because:

-The images are not scene luminant. They are **hazy and have reduced visibility** which can be because of pollution, temperature, etc. These networks may not work efficiently, since they have been trained on clear and well-lit images, and such images have confusing reflections and glares.

-The objects to be detected in the image can be of different sizes, large or small. It is possible that the regular networks along with detection algorithms are able to detect the objects which are larger (or closer to imaging device in real-time) and **fail to detect objects that look smaller** in the receptive field (are at a distance)

**Q1 (b) If a pre-trained network cannot be used for the detection and recognition tasks in Figures 3.1 and 3.2, can you come up with a deep network which can effectively be used for these tasks? (Note that you can refer to one or more related papers which solve similar tasks and come up with a modified deep network based on existing works.) Explain clearly why your deep network would be helpful for the tasks and explain the salient features of your network. Give details about the loss functions that can be used.**

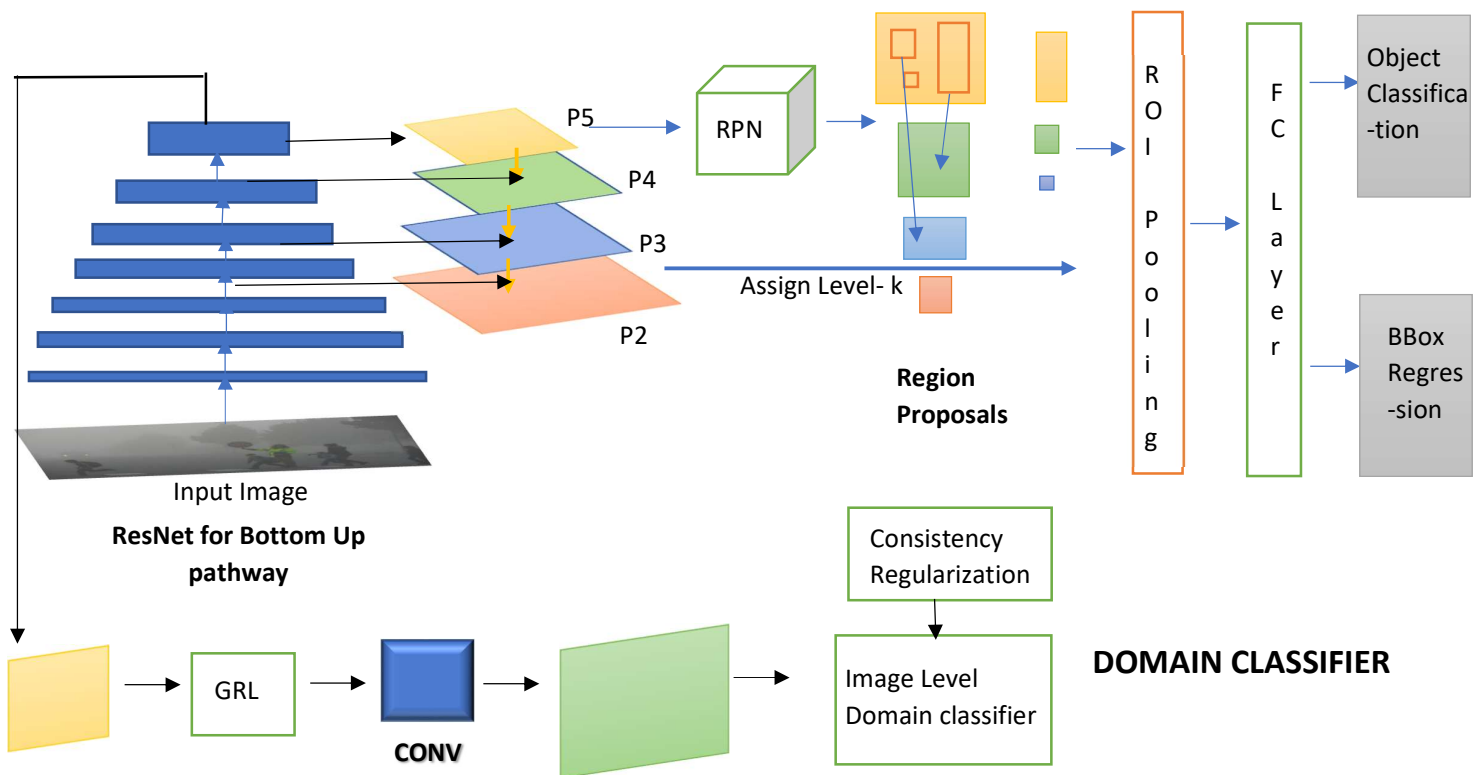
## **Haze removal**

- Although conventional techniques such as histogram equalization help in haze removal and enhancing visibility for scenes at a small distance from the imaging device, but are not that effective for the scenes which are at greater distance. Eg:-



**Instead of using a dehaze-detection cascade, an adaptive detection model can be used for this task, which will also deal with class imbalance problem in this task and can be trained end to end:**

## FPN + Faster-RCNN



FPN- <https://arxiv.org/abs/1612.03144>) Domain Adaptation-(<https://arxiv.org/pdf/1803.03243.pdf>)

**Domain Adaptation** – Domain adaptation is a sub field of transfer learning. It is basically the ability to apply a model trained on a source domain to a different (but related) target domain. Using labelled data in source domain, the model is used to solve the task in target domain. The task space is same and difference is only in source and target domain. Here, we use divergence-based domain adaptation, which helps in achieving **domain invariant feature representation**.

If such a representation is found, the model would be able to perform equally well in both the domains. The detection system would perform faithfully in different weather conditions.

## Baseline Model: FPN + Faster RCNN

### FPN (Feature Pyramid Network) based convolutional neural network

Featurized Image pyramid is a well-known image processing technique useful in detecting edges etc. and recognition tasks. **FPN leverages pyramidal feature hierarchy** (which considers high level semantics at different scales) within a deep convolutional neural network. Feature pyramids are built inside the convolutional neural network using a single input image scale, by exploiting the fact that ConvNet computes a **feature hierarchy** layer by layer, and with **subsampling** layers the feature hierarchy has an **inherent** multiscale, pyramidal shape. It combines low-resolution, semantically strong features with high-resolution, semantically weak features via a **top-down pathway** and **lateral connections**. This helps in **detection of small sized objects as well** (which is a limitation of Faster-RCNN).

### RPN

RPN generates the proposal for the objects. FPN extracts feature maps and feeds into RPN. RPN applies a sliding window over the feature maps to make predictions on the objectness (has an object or not) and the object boundary box at each location. (Now, since we have multiple scale feature

maps, the anchor boxes would be of only one size for each feature map, as opposed to original Faster RCNN.)

- After the generation of ROIs from the RPN, based on the size of the ROI, feature map layer in the most proper scale to extract the feature patches is selected, using the formula:

$$k = \lfloor k_0 + \log_2(\sqrt{wh}/224) \rfloor$$

where

$$k_0 = 4$$

$k$  is the  $P_k$  layer in the FPN used to generate the feature patch.

Then, after ROI pooling of selected feature map and ROIs, the output is fed to the subsequent layers and prediction is made.

## Domain Classifier- Unsupervised Domain Adaptation

The primary goal of using the domain classifier is to make the extracted features as much **domain invariant** as possible. Suppose the domain of the training data be **source domain (S)** and the domain of test data be **target domain (T)**. Here, training data refers to clean or de-fogged image (well lit) data and test data is **hazy image data** as given in the question.

- **Image level adaptation-** The technique used to achieve this is image level adaptation. We want the detector to be consistent between the two domains. Given an image, the detection results should be the **same regardless of which domain the image belongs**. Let  $I$  be the image representation, or the feature map outputs of the base convolutional layers (Here, ResNet). Therefore, to handle the domain shift problem, the distribution of image representation from two domains should be same (i.e.,  $P_S(I) = P_T(I)$ ), which is referred to as **image-level adaptation**.

Hence, to eliminate the domain distribution mismatch, this **patch-based** domain classifier is employed. The domain classifier is trained on each activation from the feature map. Since the receptive field of each activation corresponds to an image patch of the input image, the domain classifier predicts the domain label  $y_i$  for each image patch- 0 for source domain and 1 for target domain.

Let  $I_i$  be the  $i$ -th training image. Let  $\phi_{u,v}(I_i)$  be the activation at position  $(u,v)$  of the feature maps from the base convolutional layers. Let  $p_i(u,v)$  be the output of the domain classifier. Then the image level adaptation loss can be found using cross entropy loss:

$$- \sum_{i,u,v} y_i \log p_i(u,v) + (1-y_i) \log(1-p_i(u,v))$$

Using **H-divergence**, designed to measure the divergence between two sets of samples with different distributions, (or the distance between the two domains), to **enforce the base convolutional network to output feature vectors** that minimize the domain distance, we need to simultaneously optimize the parameters of domain classifier to **minimize** this loss, and optimize the parameters of the base network to **maximize this loss**. known as **Domain Confusion loss**, i.e.,

$$\max_f \min - \sum_{i,u,v} y_i \log p_i(u,v) + (1-y_i) \log(1-p_i(u,v))$$

where  $f$  is the base ConvNet. This can be optimized in an adversarial training manner. A consistency regularizer is also incorporated within the classifier to learn a domain-invariant RPN for the Faster RCNN model. Thus, the overall loss for the network can be written as :

$$L = L_{\text{det}}(\theta_{\text{base}}, \theta_{\text{FPN+RPN+RCNN}}) + \lambda L_{\text{dom}}(\theta_{\text{dom}}) - \lambda L_{\text{dom}}(\theta_{\text{base}})$$

where,  $\theta$  represents weights corresponding to each network component

$L_{\text{det}}(\theta_{\text{base}}, \theta_{\text{FPN+RPN+RCNN}})$  is the loss for FPN + Faster RCNN detection of object class and bounding boxes, only wrt source domain

$L_{\text{dom}}(\theta_{\text{dom}})$  is domain classifier loss wrt. source domain and target domain

$L_{\text{dom}}(\theta_{\text{base}})$  is the back-propagated loss from domain classifier wrt source domain and target domain

**GRL** - For the implementation of adversarial training, the gradient reverse layer (GRL) is used. The negative gradient of the domain classifier loss needs to be propagated back to base network(ResNet). The sign of the gradient is reversed when passing through the GRL layer to optimize the ResNet. The GRL is added after the feature maps generated by the ResNet, and its output is further fed to the domain classifier. The GRL has a hyper-parameter  $\lambda$ , which, during forward propagation, acts as an identity transform and during back propagation, it takes the gradient from the upper level and multiplies it by  $-\lambda$  before passing it to the preceding layers(base network).

**Thus, domain adaptation makes the model learn a shared space to match the distributions of source and the target domain**

*c) Give details of public data sets available in web, which can be used for training your proposed network for the detection and recognition tasks in Figures 3.1 and 3.2. If you could not public find data sets directly useful for the tasks, explain a procedure to construct new data sets from existing data sets, that can be used for these tasks. Would you need any pre-processing to make the images suitable for training? Explain. Also, illustrate the associated training procedure.*

**Dataset** : To exploit the domain shift because of different weather conditions, source domain would be clean images(obtained in clear weather). The choice for **source domain** can be Cityscapes dataset or the MS COCO dataset, to train the detection model.

First, the FPN + Faster RCNN model should be trained independently on the MSCOCO dataset

For the target domain , RESIDE dataset can be used. RESIDE is the first largescale dataset for benchmarking single image dehazing algorithms and includes both indoor and outdoor hazy images. We can use 4322 real world annotated hazy images from **RESIDE RTTS** dataset for the target domain. The images have five categories with bounding box annotations: pedestrian, car, bus, bicycle, motorcycle. But since we are in unsupervised learning setting, we can use unannotated images. Further to this, we can use Foggy Cityscapes dataset for the target domain.

Now, the pretrained detection model along with the domain classifier component is trained. For training, we can use the batch size of 2 with one image from source domain and another from target domain. Minimum Average Precision (**mAP**) metric should be used.

Optimizer can be SGD/ Adam etc., and a decaying learning rate may improve performance.

Hyperparameters can be tuned depending on the performance. The domain adaptive detector is proven to perform better for this task, than typical dehaze-detection cascade

( <https://arxiv.org/pdf/1904.04474.pdf>) We can further dehaze hazy images using a dehazing method ( AOD Net/ MSCNN/ DCP) and use this model for detection to further improve performance.



**Q2. (a) Suppose you wish to extract the frames from the video clips which contain both human and animated characters. Construct an end-to-end deep neural network architecture which can achieve this task. Explain about the components in your deep network and their functionalities, the loss function used and provide proper justification why your network will solve the task**

Videos have an added property of temporal features in addition to the spatial features present in 2D images. However, for the given task, temporal features are not relevant since we just have to classify and then extract the frames with BOTH human and animated characters. There is no action recognition involved. We just have to extract a set of images from given set of video frames(2D images). Hence, a deep convolutional neural network for multi-label image classification would suffice.

**For classification of frames:** The problem can be interpreted as two binary classification problems with 1 as label if character is present in the image and 0 otherwise. Thus, we can define an architecture which gives 2 dimensional outputs. For eg., the model outputs (x,y) where  $x \in \{0,1\}$  depending on whether the image has animated character or not. Similarly  $y \in \{0,1\}$  depending on whether the image has human character or not.

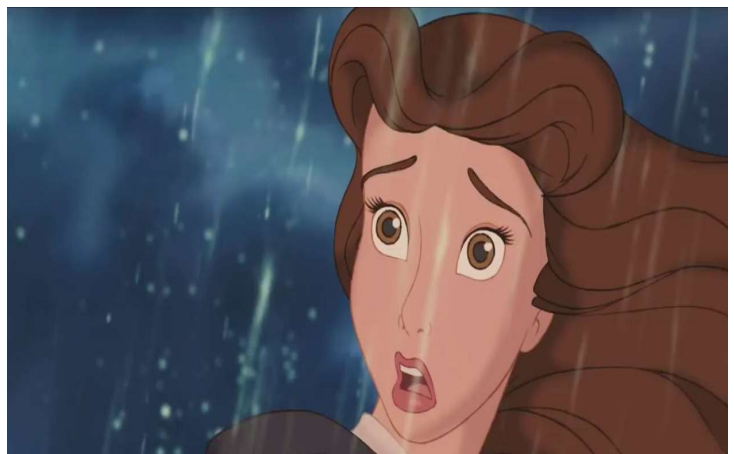
**Frames with both human and animated characters**

**True Label- (1,1)**



**Frames with only animated characters**

**True Label – (1,0)**



**Frames with only human characters**

**True Label- (0,1)**

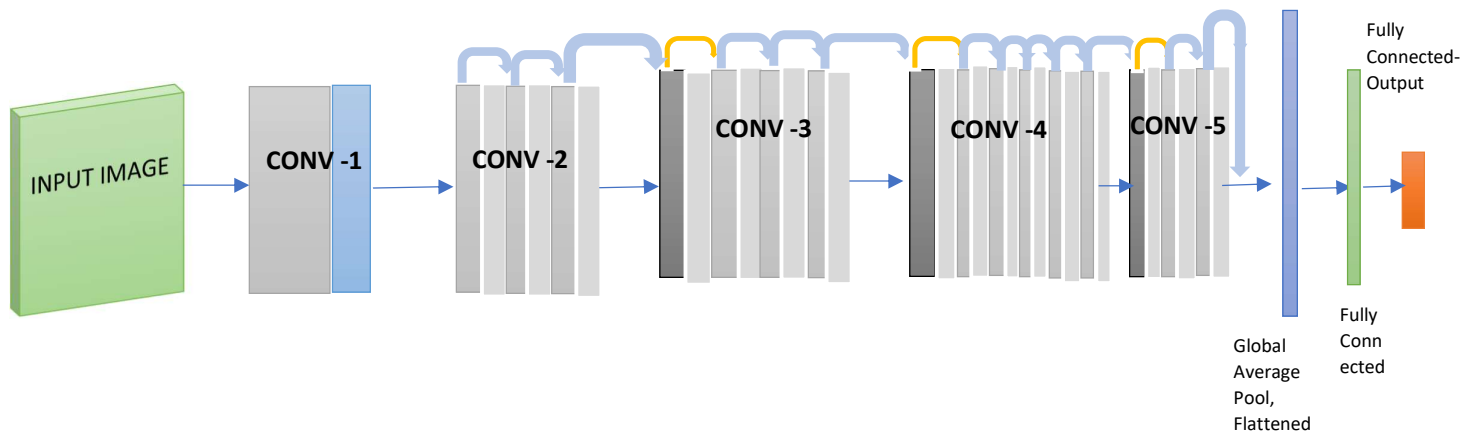


**Frames with neither human nor animated characters**

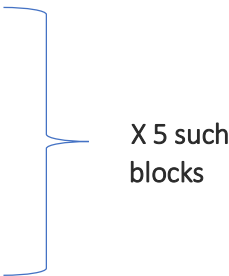
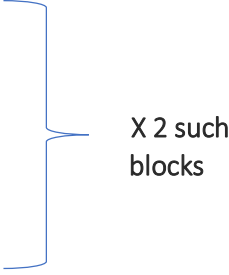
**True Label – (0,0)**



The proposed model (inspired by ResNet-34) is as follows:



Stage Name	Input Image	Operations	Output Image
Conv-1	Size – (384X384,3)  Padding – 3	<ul style="list-style-type: none"> <li>• <b>Convolution</b> 7X7 ,no. of filters = 64, [Each filter has dim 7X7X3]</li> <li>stride-2 Dim- (112X112X64)</li> <li>-Batch Normalization</li> <li>-Swish</li> <li>-Padding = 2</li> <li>-Max Pooling 3X3 stride-2 Dim- (56X56X64)</li> </ul>	Size – (96X96,64)
Conv-2	Size- (96X96X64) Padding =1 Before all convolutions	<ul style="list-style-type: none"> <li>• <b>Convolution</b> 3X3, filters = 64, stride 1</li> <li>-Batch Normalization</li> <li>-Swish</li> <li>• <b>Convolution</b> 3X3, filters = 64, stride 1</li> <li>-Batch Normalization</li> </ul> <p>X3 such blocks</p>	(96X96X64)
Conv-3	Padding =1  Before all convolution operations	<ul style="list-style-type: none"> <li>• <b>Convolution</b> 3X3, filters = 128, stride 2 → This would down-sample feature map from 96 to 48</li> <li>-Batch Normalization</li> <li>-Swish</li> <li>• <b>Convolution</b> 3X3, filters = 128, stride 1</li> <li>-Batch Normalization</li> <li>• <b>Convolution</b> 3X3, filters = 128, stride 1</li> <li>-Batch Normalization</li> <li>-Swish</li> <li>• <b>Convolution</b> 3X3, filters = 128, stride 1</li> <li>-Batch Normalization</li> </ul> <p>X3 such blocks</p>	(48X48X128)
Conv-4	(48X48X128)  Before all convolution operations	<ul style="list-style-type: none"> <li>• <b>Convolution</b> 3X3, filters = 256, stride 2 (downsample to 24)</li> <li>-Batch Normalization</li> <li>-Swish</li> <li>• <b>Convolution</b> 3X3, filters = 256, stride 1</li> </ul>	(24X24X 256)

		-Batch Normalization  <div> <ul style="list-style-type: none"> <li>• Convolution 3X3, filters = 256, stride 1</li> </ul> </div> -Batch Normalization <div> <ul style="list-style-type: none"> <li>• Convolution 3X3, filters = 256, stride 1</li> </ul> </div> -Batch Normalization <div>  </div>	
Conv- 5	(24X24X256)	<div> <ul style="list-style-type: none"> <li>• Convolution 3X3, filters = 512, stride 2 (downsample to 12)</li> </ul> </div> -Batch Normalization -Swish <div> <ul style="list-style-type: none"> <li>• Convolution 3X3, filters = 64, stride 1</li> </ul> </div> -Batch Normalization  <div> <ul style="list-style-type: none"> <li>• Convolution 3X3, filters = 512, stride 1</li> </ul> </div> -Batch Normalization -Swish <div> <ul style="list-style-type: none"> <li>• Convolution 3X3, filters = 512, stride 1</li> </ul> </div> -Batch Normalization <div>  </div>	(12X12X512)
Fully Connected Block	(12X12X512)	<ul style="list-style-type: none"> <li>• Global Average Pooling – (1X1X512)</li> <li>• Flatten – (512X1)</li> <li>• Linear( in – 512, out – 256)</li> <li>• Fully Connected- (in – 256, out -2) activation- sigmoid</li> </ul>	(2X1)

The purpose of using deeper network is to extract more and more critical features from the high resolution (384X384) images to make good predictions.

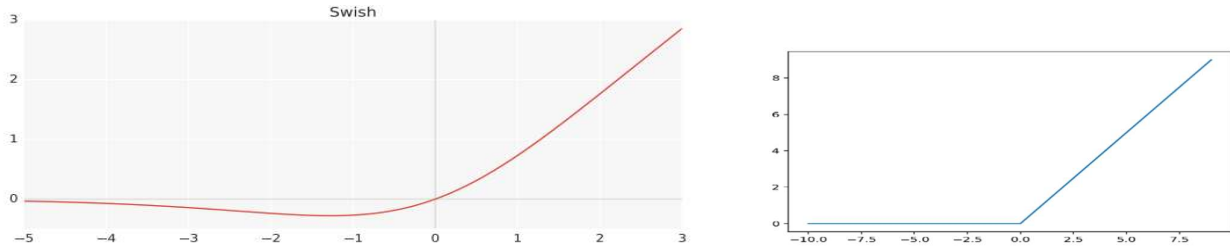
The objective of various components are :

1. **Convolution Operation**- The convolution operation between the filters and the image facilitate the extraction the high-level features such as edges, patterns etc. from the input image. As we go deeper in the network, the learned filter weights extract much finer details with more semantic value (high level features).
2. **Padding** – Padding is used before every convolution layer to preserve the dimension of the feature map. It also helps to process information from the corners and edges of the feature map.
3. **Batch Normalization** - The model heavily depends on batch normalization, used to normalize the outputs from previous layers activations. It has a regularization effect to reduce overfitting.
4. **Skip Connections** – Deeper networks give rise to vanishing gradients problem,as a result the performance gets saturated after certain depth. To address this, skip (residual) connections have been added so that gradients can flow back to initial layers easily. The blue coloured

connections are identity connections, and the yellow coloured arise because of change in dimension due to downsampling ( which is done by increasing stride, instead of pooling), known as projection shortcut.

The projection shortcut performs a convolution operation to ensure the volumes at concatenation are of same size.

5. **Swish** – Swish activation has been used throughout(except linear in fully connected layer).



Swish activation function is known to match/outperform ReLU activation. It is unbounded above and bounded below similar to ReLU. ReLU simply zeroes out activations with negative values, whereas Swish only zeroes out very large negative values, since negative values can be equally important in capturing the information. Moreover output landscape is smooth for Swish as compared to ReLU

6. **Fully Connected Block** – The main reason for the network to work for the given task is the fully connected block. The Fully connected layer uses sigmoid activation rather than softmax, since the two dimensional output components are independent to each other as opposed to typical multi-class classification. The sigmoid activation predicts independent probability of human or animated characters being present in the image.
7. Hence, the choice for loss function is **Binary Cross Entropy**. Predictions are made by using thresholding on the output.

If the final output score for a class is more than 0.5, the image is classified into that class. It is possible that both the output components have score more than 0.5, which means both human and animated characters are present in image

**(b) Give examples of data sets (available in public) that can be used for training your neural network. If existing data sets can be used for training your neural network, explain about the pre-processing tasks involved, if any.**

According to me, no datasets are readily available to be used for this task. We need to construct a dataset. If suppose existing datasets were available, the pre-processing involved would involve:

1. Rescaling the images to the input size of the network (384X384)
2. Data augmentation – in case there is a class imbalance in the dataset. For eg. If there are 900 human frames, 30 human + animation, 30 animation frames, 40 Blank, the network would mis-classify most images as human.  
For class with less data in the dataset, data augmentation techniques such as Horizontal or Vertical Flip, rotating, taking Centre Crop, Four corner crop, changing hue of images are useful for increasing the number of images of that particular class
3. Normalize each input image for gradient propagation

Note- Data augmentation won't be applied on the test data.

**(c) If no data sets can be directly used, can you construct new data sets by adapting the existing data sets, so that you can use the newly constructed data sets for training? Explain the construction procedure.**



We can easily construct a dataset of 15000+ images. The construction procedure for a dataset is as follows:

1. Collecting raw data- Videos with both human and animated characters can easily be found on YouTube.
2. Extracting Video Frames - using OpenCV, and storing them locally.

For eg. The videos referred to in the question had frame rate 23.976 fps, 30 fps and 23.976 fps respectively. The duration of videos would also determine the number of frames that can be extracted from videos. 1800 frames can be extracted from videos with rate 30fps and duration 1 minute. Hence, we can find approximately 10-12 videos with similar frame rate and similar duration to avoid any dataset imbalance.

3. Some frames extracted from videos would have low brightness and contrast because of transitions between scenes. This might lead to poor performance of the network. Hence, a corrective image processing algorithm- CLAHE( Contrast limited Adaptive Histogram Equalization) can be applied on ALL images/frames. This would correct brightness and contrast of all the frames- ( This can be done for color images by converting them to HSV space and applying the CLAHE to value using openCV)

**Applying CLAHE on low contrast frames and high contrast frames:**



4. The extracted images would be of varied sizes, so they need to be resized using appropriate interpolation method (eg. Bicubic) to 384X384 (or to higher resolution if information loss is not desired. In that case, since the resolution would increase, the network would need scaling up as well, using compound scaling for better performance)

Eg:                      720 X1280



384X384



5. Lastly we need to manually label the data using 4 labels- [1,1], [0,0], [0,1],[1,0] indicating the presence or absence of human or animated character in each frame, and save the labels in a csv file.

6. We need to use data augmentation techniques to avoid class imbalance. For eg. The frames with no human or animated characters( text frames, blank frames, frames of places) would be very less. This needs to be taken care of.

Note- Data augmentation won't be applied on the test data.

***(d) Suppose you have suitable data sets (either public or constructed) available for training. Explain how you will use them for training your network.***

The training procedure is as follows:

1. We first split the frames dataset into Train- Validation – Test folders using ratio 75:5:20.
2. After data pre-processing, we divide use data loaders to load the data using mini batch of different sizes, to improve performance.
3. SGD optimizer would be used along with a learning rate scheduler with initial learning rate – 0.01 which reduces by factor 0.1 every 10 epochs and momentum 0.9.
4. These hyperparameters can be tuned accordingly using grid search or manually, depending on the performance of the network.
5. To evaluate the performance, accuracy metric is used. Training epochs = 1000 (can be tuned later). After each epoch, the training accuracy and validation accuracy along with training loss and validation loss is noted. Using cross validation method, I would also ensure the network doesn't overfit. The best validation accuracy is noted down among all the epochs and that particular model would be saved.

To use the trained model for the given task, the probe video would be downloaded and processed using the **same procedure** followed for dataset construction (except augmentation), and fed to the model. From the output, predictions would be made for each particular frame and the frames predicted with (1,1) would be the desired frames.

***(e) Can you modify your network and data sets so that they can be also used for localizing the animated characters in each frame? Explain.***

Dataset Modification: Apart from adding the labels, we would now require annotating each frame with a bounding box coordinates. Also, now the labels added would be scalar – 0 if the frame doesn't have any animated character and 1 if it has.

Since single object localization is desired, this can be achieved by modifying the fully connected block of the network, and the loss function. Modifying the very last layer as fully connected with sigmoid activation, and adding a regression layer for bounding box predictions, the output would be 5 dimensional. The first component would be 0 or 1 depending on whether image has animation or not, and other 4 components as bounding box coordinates.

Also, the MSE loss would be added to the loss function along with binary cross entropy

Blog referred for this question - <https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035>