Computer vision
Assignment 3
Hossein Yahyaei
99222119

1.

This is the cost function of the Fast RCNN :

$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1]L_{\text{loc}}(t^u, v)$$

In this cost function first loss function is cross entropy which in its paper is simplified as $L_{\text{cls}}(p, u) = -\log p_u$ which is exactly the same as cross entropy and is used for maximizing the likelihood of detected objects by the network in RCNN and the second loss function is smooth L1 loss to define how bad was the regression loss or how off was the bounding boxes from the truth but using smooth version of L1 loss is because small errors that are in between 0 and 0.5 becomes smaller and gets ignored as they were no mistakes in networks predictions, not to forget about the coefficient of the smooth L1 loss it is used for showing that there is no need to update the network when there is no true bounding box (object) in that specific predicted bounding box for dealing with imbalanced data.
And this is the cost function of the region proposal part of Faster RCNN it :

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*).$$

Faster RCNN also has the cost function of fast RCNN as well as this cost function; these two costs are then added up together as a final loss. It is similar to the first cost function introduced in the Fast RCNN but instead of n classes there exist two classes that are defining whether there is an object in that specific anchor box or not . this is shown using P* Thus if there was any object the regression head would be updated .
RPN tries to predict the difference needed to be applied on each and every anchor box to reach the true bounding box Hence it predicts the difference between the true bounding box and current anchor box As it is being shown in the provided picture this differences are defined as

the deltas in height width center position of X and Y . RPN and anchor boxes are proposed to overcome the low speed of the selective search in RCNN and Fast RCNN so instead of searching for the highly probable bounding boxes in the image it tries to predict the difference between defined anchors and true bounding boxes .This RPN improves the speed of the network because high time complexity in RCNN and Fast RCNN was according to the selective search they had .

2.

A. The most important difference between Lucas Kanade and Horn Schunck is in the extra term that Horn Schunck has that is smoothness term. This smoothness term defines that flow in the whole image must be smooth which means that the difference in flow in the neighboring pixels is small. Lucas Kanade is more robust to noise in the image because Horn Schunck has this extra term of flow smoothness; it is more susceptible to noises because if its neighbors have small noise they will affect the flow of the current pixel to differ .In terms of their assumptions both of them assume constancy in brightness over time also they both assume motion in two consecutive frames are small they also assume motion of the neighboring pixels are constant .But Horn Schunck has extra assumption which is flow field in the whole image must be smooth (in other words difference in the optical flow of the neighboring pixels must be small)
B. This regularization parameter actually indicates how strict the smoothness constraint must be. In other words if this parameter increases it forces the flow of the whole image to be in a specific region and does not change much.
   For example if it's too large the optical flow field is a plane because too much smoothness in the velocities leads to the constant velocity for the whole image and if it is small it leads to a wavy form of the optical flow field because it lets the velocities be diverse.
C. The aperture problem occurs when the local image features, such as edges or textures, are oriented in a way that does not provide enough information about the true motion direction , because when the direction of the motion is the same as the direction of the edge(*or other features that has small gradient info*) in an image exact motion is not observable . Thus an edge or a single point is not enough info to calculate the true direction of the motion at least (*other than its magnitude*). The aperture problem affects the accuracy and reliability of motion estimation algorithms, particularly in scenarios where the motion is predominantly along edges or textures.The estimated motion tends to be biased towards the direction perpendicular to the edge, rather than the true motion direction.

3.

A. Simple Flownet has only an advantage of speed over its complex version but if comparison is based on the performance on their accuracy it is desirable to conclude that complex one had better result because of the correlation layer it has but in fact they both get the same result and the difference in their performance on accuracy is not significant.anyway the Simple flownet used concatenation of two frames in video and then passes it thru a fully convolutional network like Unet (Unet hasn't been proposed at the time thus it's just an example) to get the optical flow as its target But its complex

version has two input stream each gets a single frame of two consecutive frames and then passes them thru their corresponding convolution layers afterwards they are going thru a proposed correlation layer , output of this layer goes to a network similar to the simple version to get the optical flow , in fact there is a useful skip connection from convolution feature map of the first frame to the refinement net of the Network that tends to induce information about how the first frame was to the optical flow .

B. A two stream convolutional layer gets two different frames and then encode them to feature maps to achieve reach perception of the context and then applies a correlation layer on these two feature maps that are given by the two convolutional encoder, this correlation layer is used to detect movement of specific features in the whole image (*not a big fan*) output of this correlation layer is given to the convolutional encoder to get even more information out of this difference in the movement finally output of encoder is given to the refinement network to get the optical flow but the main idea that is important to me is the fact that it uses upsampled optical flow as an extra feature map for next up convolved layer for passing multi scale optical flow information to the next layer most papers used the top down approach to get the multiscale optical flow information to the next layers but as it is being shown in this architecture it uses bottom up approach to get this which result in better understanding of optical flow in large motions .

   Traditional methods have been more effective in dealing with small movements in the image data. FlowNetC, like its predecessor FlowNet, struggled to accurately estimate flow when the displacements were minimal.The performance of FlowNetC on datasets that closely represent real-world scenarios was not on par with the state-of-the-art traditional methods. This is due to the fact that real-world data often contains complex motion patterns and varying illumination conditions that are challenging for deep learning models trained primarily on synthetic datasets.

4.

Self attention mechanism is a mechanism that extracts expected information of different patches of the image in every single part of the image related to a single patch . For each patch similarity between itself and every other patches is being calculated to define how much information of other patches needed to be integrated to the current information of the current patch, this defined amount then will be multiplied by its corresponding information to get the final context extracted from a single patch and these context corresponding to a single patch summed up and finally a vector that defines the context of this current query patch in the whole image but this information in a single self attention head is seen from a single aspect which is defined by the embedding layer at the begging of the self attention layer.(this is why multihead attention was proposed).

The number of learnable parameters are as follow :

- Linear layer for key query value embedding in each head = 512*64 + 64
  Because we have three of these in each head thus its = 3*(512*64 + 64)=98496 parameter
- Multi head attention = 98496*8 + (512*512 + 512) = 361152 parameter
  (*(512*512 + 512) this is the number of parameters in linear layer at the end of the multi head attention)

- Each encoder block = 361152 +2*(512*512 + 512) + 4 = 886468 parameter
  (4 is used for number of parameters in the layer normalization used in encoder block twice)
- 12 blocks of encoder = 12*886468 = 10637616
- Positional encoder = 14*14*512 = 100352 parameter

Thus the final number of parameter despite the fact it's a classification task and needs a classification head to define the final number of parameters (cause it hasn't been mentioned I ignored it) anyway it's **10737968** parameter in total

Output of the vision transformer has the same number of patches as input plus a cls token that has information of the whole image.Thus by the given number of patches that are 14by 14 we would have 196 patches plus a cls token at the end.

Pros and cons compared to CNNs : ViT has the advantage of global vision which means it can gather global context associated to a single patch but as we all know attention is pointwise which means it can only compare a point to another which has the problem of temporal information lost but CNNs are better at capturing temporal information . Another con is attention suffer from quadratic computational cost they have compared to CNNs.