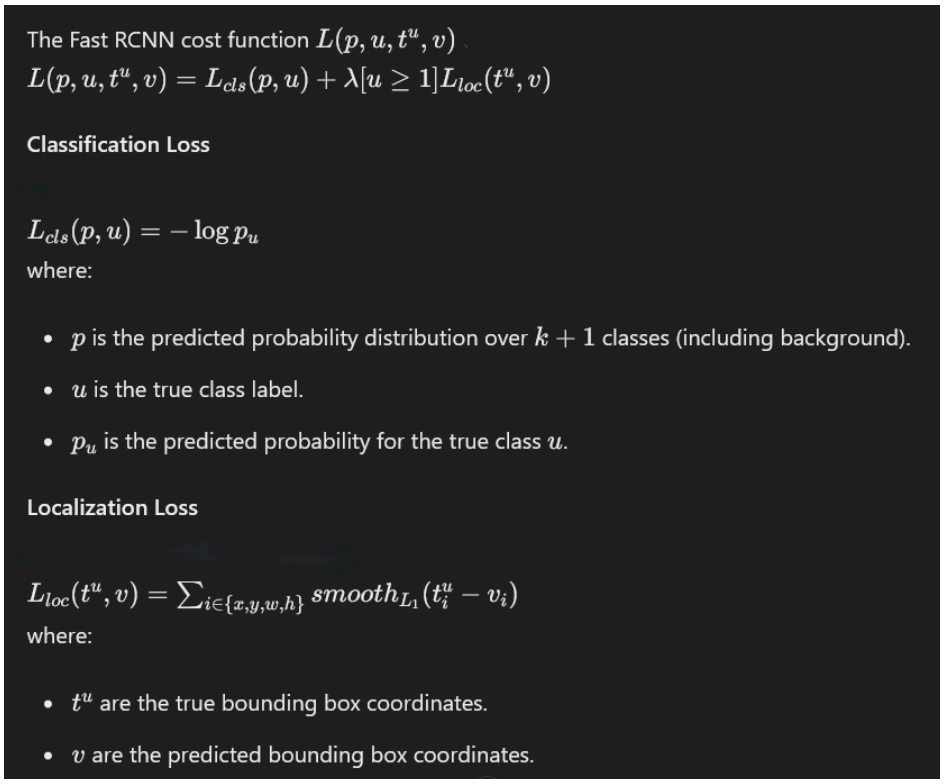
Farimah Rashidi(99222040) - HW3

Question 1)

Fast RCNN:

Look at these formulas and descriptions:



The Fast RCNN cost function $L(p, u, t^u, v)$

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

Classification Loss

$$L_{cls}(p, u) = -\log p_u$$

where:

- p is the predicted probability distribution over $k+1$ classes (including background).
- u is the true class label.
- p_u is the predicted probability for the true class u.

Localization Loss

$$L_{loc}(t^u, v) = \sum_{i \in \{x,y,w,h\}} smooth_{L_1}(t_i^u - v_i)$$

where:

- t^u are the true bounding box coordinates.
- v are the predicted bounding box coordinates.

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

https://towardsdatascience.com/fast-r-cnn-for-object-detection-a-technical-summary-a0ff94faa022

Fast RCNN suggests potential regions in an image by using an external method, such Selective Search. After that, these regions are processed in order to classify objects and improve their bounding boxes. One background class (k+1k+1) is handled by the model. Regression loss is used to improve bounding boxes, and classification loss is used to precisely identify objects in its cost function. Its decreased speed as a result of using a separate algorithm for region proposals is a disadvantage.

Faster RCNN:

By integrating region proposal generation into the network architecture through a Region Proposal Network (RPN), Faster R-CNN transforms object detection. It accomplishe end-to-end training in this way, greatly increasing speed and simplifying the entire process. The RPN receives independent training at first. Next, by using the learned RPN's insights, the classifier is trained, allowing its weights to be optimized. A symbiotic relationship is created between the two components when this trained classifier influences the RPN in turn. The outcome of this ongoing iterative process is an extremely precise and eficient object detection framework.

Faster RCNN works just as Fast RCNN does. It takes a feature map from an image and then tries to find regions of interest (ROI) from there, except that in faster RCNN we do not use selective search but a fully convolutional model called Region Proposal Network (RPN).

Relationship between RPN and photo(anchor boxes):

- **Anchor Boxes as Prior Boxes**: Densely placed over the image, anchor boxes function as prior boxes with different scales and aspect ratios. These boxes serve as possible object detection regions of interest.
- **The Function of RPNs in Proposal Development:** A tiny network is slid across the convolutional feature map by the RPN, a part of the Faster R-CNN architecture, which evaluates each spatial location. It determines whether or not each anchor box at these places has things by looking at it.
- **Prediction of Objectness Score:** The RPN determines an objectness score for every anchor box, which indicates the probability that the box will hold an object of interest instead of background clutter. The likelihood that an object will be found in an anchor box is indicated by this score.
- **Box Boundary Regression:** Furthermore, bounding box deltas—adjustments made to the anchor box coordinates—are predicted by the RPN. By using these deltas, the anchor boxes are improved so that they more closely match the bounding boxes of the actual objects in the picture.
- **Matching with Ground Truth Boxes**: Using the Intersection over Union (IoU) metric, anchor boxes are matched with ground truth bounding boxes during training. Positive samples, or anchors with high IoU values, are used to train the network to anticipate item presence and refine box coordinates.

There are some ***benefits*** of using anchor boxes:

Training Ease: By offering a structured framework for learning object detection tasks, anchor boxes make training easier. They facilitate more effective training convergence by allowing the model to focus on improving predictions within preset regions of interest.

Consistency Across Spatial Locations: By using anchor boxes, the image is consistently covered in a variety of spatial locations, making it possible for the model to capture objects at varied scales and positions consistently.

Managing object Occlusion: The model can more effectively manage intricate object arrangements by taking into account several anchor boxes for each spatial location. When objects are partially obscured or covered by other items, anchor boxes help with reliable object detection.

Question 2)

A)

The method used for optical flow estimation is where the Lucas-Kanade and Horn-Schunck algorithms differ most:

- **Lucas-Kanade algorithm:** Local picture gradient information serves as the foundation for the Lucas-Kanade method. It is predicated on the idea that, in the immediate area of a pixel, optical flow is continuous. It calculates the optical flow by using the gradients of the image and the template window to solve a system of linear equations.
- **Horn-Schunck approach:** On the other hand, this approach makes the assumption that the optical flow is uniformly distributed throughout the image. It formulates the problem of optical flow estimation as one of energy minimization, where the objective is to reduce the difference between the gradients of the estimated flow field and the gradients of the observed image, while simultaneously encouraging smoothness in the flow field.

In relation to noise resistance:

- **Lucas-Kanade:** Because of its local approach, this method is generally more noise-resistant than Horn-Schunck. By focussing on tiny local neighborhoods, noise has less of an impact on the estimating procedure.
- **Horn-Schunck:** It may be more sensitive to noise because it takes into account the full image and applies smoothness constraints, particularly in areas with poor texture or where the smoothness assumption is broken.

B)
The parameter α in the Horn-Schunck algorithm serves as a regularization parameter.It manages the trade-off between smoothness regularization—which promotes smoothness in the flow field—and data fidelity, which matches actual image gradients.

- Reducing α causes less regularization, which enables the flow field to conform more closely to the gradients of the observed image. This could result in flow estimation that is more accurate, but it could also make you more sensitive to noise and outliers.
- By increasing α, smoother flow fields are enforced by increasing the level of regularization. While this may assist reduce the impact of noise and outliers, it may also cause the estimated flow field to become overly smoothed and lose details.

For instance, a situation in which the α parameter is set extremely low. Accurate flow estimation can be achieved when the estimated flow field closely resembles the observed gradients in areas with significant gradients in the image and high texture. On the other hand, the flow field can be significantly affected by noise in areas with low texture or high noise levels, which could result in inaccurate estimations.

C)
When predicting motion direction using local image measurements, particularly in regions with uniform motion or poor texture, the aperture problem in optical flow appears. Accuracy suffers from this uncertainty, especially when motion is not adequately restrained by image gradients. Seeing motion is difficult when looking through a small window or aperture that is witnessing a moving object, like a textured surface or an edge. This leads, particularly at sharp edges or textures, to incomplete or underestimated motion estimates. This is addressed by methods that increase the resilience and precision of motion estimate, such as adding more constraints or taking use of temporal coherence in video sequences.

Question 3)

A)
FlownetS prioritizes simplicity and efficiency, while FlownetC prioritizes accuracy and complexity, reflecting a trade-off between performance and computational cost.

FlownetS:

- *Simpler Architecture:* The design of FlownetS is less complicated because of its ewer layers and parameters. Because of its simplicity, it is typically lighter and faster for inference and training.

- *Shared Weight Utilization*: This variation reduces the amount of memory and computation required for training and prediction by using shared weights across many execution processes.
- *High Performance at High Speed*: FlownetS usually works effectively in situations where speed is essential because of its simple construction.

FlownetC:

- *Deeper and More Complex architecture:* With more layers and parameters, FlownetC's architecture is more intricate and deeper. As a result, the model can learn and estimate increasingly complex optical flow patterns.
- *High Accuracy with Increasing Complexity:* FlownetC often provides higher optical flow estimation accuracy because to its deeper and more sophisticated architecture, especially in situations with more variations and complexity.
- *Skill to Learn complicated Patterns:* FlownetC's deeper architecture allows it to learn and estimate more complicated optical flow patterns, which may be required in complex settings.

FlownetS is better for applications where speed and efficiency are critical, while FlownetC is preferred for tasks where accuracy and complexity are paramount.

Now let's get deeper about their architecture differences:

- When compared to FlownetC, FlownetS usually has less convolutional layers. These layers frequently have fewer parameters and are shallower.
- Simpler or less complex network topologies, including lightweight versions of MobileNet or SqueezeNet or basic convolutional neural networks (CNNs), may be used by FlownetS.
- Because of its simplicity, FlownetS architecture can be applied to real-time applications and settings with limited computational resources, since it results in faster training and inference times.
- On the contrary, FlownetC utilizes a more profound and intricate network architecture, frequently incorporating numerous convolutional layers, residual connections, or other sophisticated architectural features.
- More computationally demanding network topologies, like densely connected networks like DenseNet or deeper CNNs like ResNet, might be used by FlownetC.
- Although FlownetC can capture more complex patterns and details in the optical flow due to its greater depth and complexity, this results in higher accuracy at the cost of increased computational resources and longer training times.

B)
Main Components of FlownetC Architecture:

- Encoder:
  Function: By downsampling the spatial resolution and deepening the feature maps, convolutional layers are used by the encoder to extract features from input images.
  Structure: To capture high-level semantic information, it consists of pooling and convolutional layers, with a ReLU activation coming after each convolutional layer.

- Decoder:
  Function: Using the encoded features, the decoder reconstructs the optical flow field.To restore the spatial resolution, it upsamples the feature maps gradually.
  Structure: To improve the spatial resolution, the decoder is made up of transposed convolutional (deconvolutional) layers. To mix high-level and low-level features, skip connections from equivalent encoder levels are frequently incorporated.
- Warping operations:
  Function: By using the current estimate of the optical flow, warping aligns the second image with the first image. By comparing the distorted second image to the original image, this technique aids in improving the flow estimation.
  Structure: Differentiable spatial transformers are typically used to apply the predicted flow field to the second image, so "warping" it so it matches the first image.
- Correlation layers:
  Function: The similarity between the feature maps of the two input images is calculated via correlation layers. This aids in determining pixel correspondences between the two images.
  Structure: The correlation procedure creates a cost volume that represents the matching costs by comparing patches from the feature maps of the two images. The process of estimating flow is guided by this cost volume.

Connection between components in FlownetC:
- Flow refinement: Using input from the warped features and correlation map, the decoder improves the flow estimate. The preservation of intricate spatial information from the encoder is guaranteed by skip connections.
- Iterative warping: By gradually aligning the features, warping procedures aid in the improvement of the flow estimations at various stages.

- Feature extraction: Each input frame is processed individually by the encoder, which then uses convolutional layers to extract features by downsampling the spatial resolution and deepening the feature maps. The correlation step subsequently makes advantage of these features.
- Correlation computation: By merging these feature maps, the correlation layer produces dense correspondences that the motion decoder can utilize to estimate.

Challenges in motion estimation with FlownetC:

- Large displacements:
  The challenge lies in precisely estimating large motions that occur between frames.
  Reason: The encoder's downsampling and limited receptive field cause the fine details necessary for accurate estimation to be lost.
- Complex motion patterns:
  Challenge: Complex and non-linear motion patterns are difficult for standard correlation and convolution operations to represent.
- Occluded regions:
  Challenge: In situations when some portions of the image are visible in one frame but not the next, estimating flow might be challenging.
  Reason: Missing correspondences cause ambiguity, which might result in inaccurate flow estimates or holes.
- Complexity of computation:
  Problem: Computational load is increased by correlation layers and subsequent operations, making real-time applications more difficult.
- Generalization:
  Challenge: Unseen scenarios or motion types not included in the training set may cause the model to perform poorly.

Question 4)

Self-Attention in Visual Transformers:

Through a process known as self-attention, each component of a sequence—in this case, picture patches—interacts with every other component to calculate a weighted sum of values, with the weights being dynamically changed according to how similar the components are to one another. Compared to localized operations in CNNs, self-attention in the context of visual transformers aids in capturing long-range dependencies and relationships between various regions of an image, allowing for more holistic understanding.

Advantages and limitations of visual transformers compared to CNNs:

Advantages:

- Global context: Compared to CNNs, the model captures global context more well since self-attention enables it to take into account the relationships between every component of the image.
- Transformers can be parallelized more effectively than the sequential operations in CNNs, which could result in quicker training periods when the right hardware is used.

Limitations:

- Data efficiency: Transformers are often less data-efficient than CNNs since they need more data to function at a level that is comparable to CNNs.
- computing difficulty: Higher computing needs result from self-attention's quadratic difficulty with respect to sequence length (number of patches), particularly for high-resolution images.
- Memory usage: Compared to CNNs, visual transformers might use a lot more memory because of the high number of parameters and the nature of self-attention.

number of trainable parameters and the number of patches generated from the input image:

The input image with 224 x 224 pixels is divided into 16 x 16 pixels patches
Number of patches: $224/16 \times 224/16 = 14 \times 14 = 196$

Each patch is flattened and projected to the embedding dimension (512).

Trainable parameters for linear projection: $16 \times 16 \times 3 \times 512 = 12288 \times 512 = 393{,}216$

Each of the 8 heads has its own projection matrices for queries, keys, and values, each of size $512/8 \times 512$
Trainable parameters for each head: $3 \times (512/8 \times 512) = 3 \times 64 \times 512 = 98{,}304$
For 8 heads: $8 \times 98{,}304 = 786{,}432$

Projection of concatenated outputs of attention heads back to the embedding dimension (512).
Trainable parameters: $512 \times 512 = 262{,}144$

Feed-Forward network:
Consists of two linear layers with an intermediate dimension(for example 2048).
Parameters: $512 \times 2048 + 2048 \times 512 = 1,048,576 + 1,048,576 = 2,097,152$

Layer norms:
Parameters are negligible compared to other components.

All parametres of one layer:
Sum of parameters for multi-head attention, output linear layer, and feed-forward network: $786,432 + 262,144 + 2,097,152 = 3,145,728$

Total Parameters for 12 Layers = $12 \times 3,145,728 = 37,748,736$

Now we add the parameters from the patch embedding step:

Total number of trainable parameters: $393,216 + 37,748,736 = 37,141,952$