Computer Vision

FCN
Mask R-CNN
RNN for video understanding
Spatio-Temporal models
Action/Activity Recognition

Unit 5

Deep Learning Techniques for Computer Vision

Introduction to CNNs ; Visualization of Kernels ; Backprop to image / Deconvolution Methods ; CNNs for Detection : Background of Object Detection - R-CNN - Fast R-CNN - Faster R-CNN ; CNNs for Segmentation: FCN - SegNet - U-Net - Mask-RCNN ; Recurrent Neural Networks (RNNs) : CNN and RNN models for video Understanding : Spatio-Temporal Model - Action / Activity Recognition

* Introduction to CNNs

→ Convolutional neural networks or CNNs, are a specialized kind of neural network for processing data that has a known grid-like topology (time series (1D grid), images (2D-grids)

→ The network employs a mathematical operation called convolution

Convolution

→ Convolution is an operation on two functions of a real-valued argument

→ It is an integral that expresses the amount of overlap of function g as it is shifted over another function f.

$$[f*g](t) = \int_0^t f(\gamma)\, g(t-\gamma)\, d\gamma$$

→ In a convolutional network, the operation is written as

$$s(t) = (x * w)(t)$$

→ This is a 1D kernel

input

kernel

(also called feature map)

→ For a 2D network, a two dimensional kernel can be used.

This would be of the form

$$S(i,j) = (K*I)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n)$$

Basic working

1. A CNN consists of a sequence of layers, where each layer transforms the activations or outputs of the previous layer through another differentiable function.

2. The most common building blocks are the:

   (i) convolution layer

   (ii) pooling layer

   (iii) Fully connected layer

Convolution Layer

→ During a forward pass, a filter slides across the input and computes the activation map of the filter at that point. This filter is implemented with convolution.

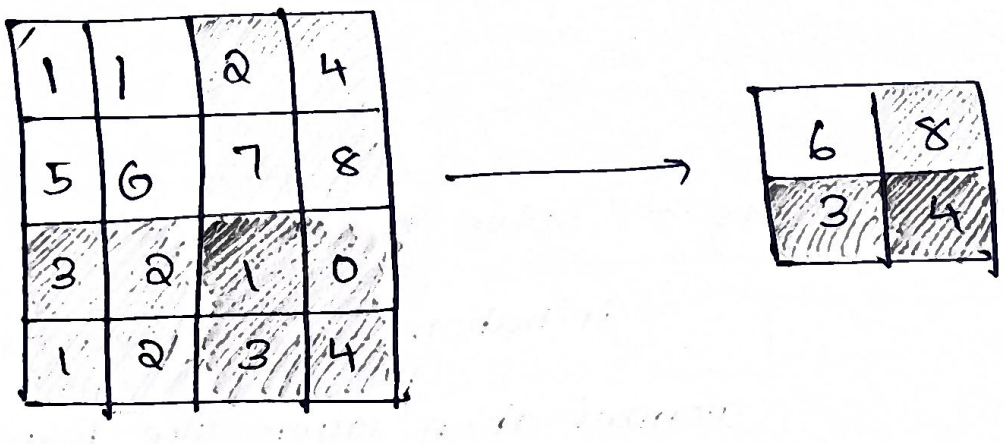
Pooling Layers

→ Pooling layers are placed between convolution layers - they reduce the size of the image across layers by samping.

→ In max pooling - the max value in a window is chosen

In average pooling - average over the window.

→ Pooling acts as a regularization technique to avoid overfitting



Fully connected Layers

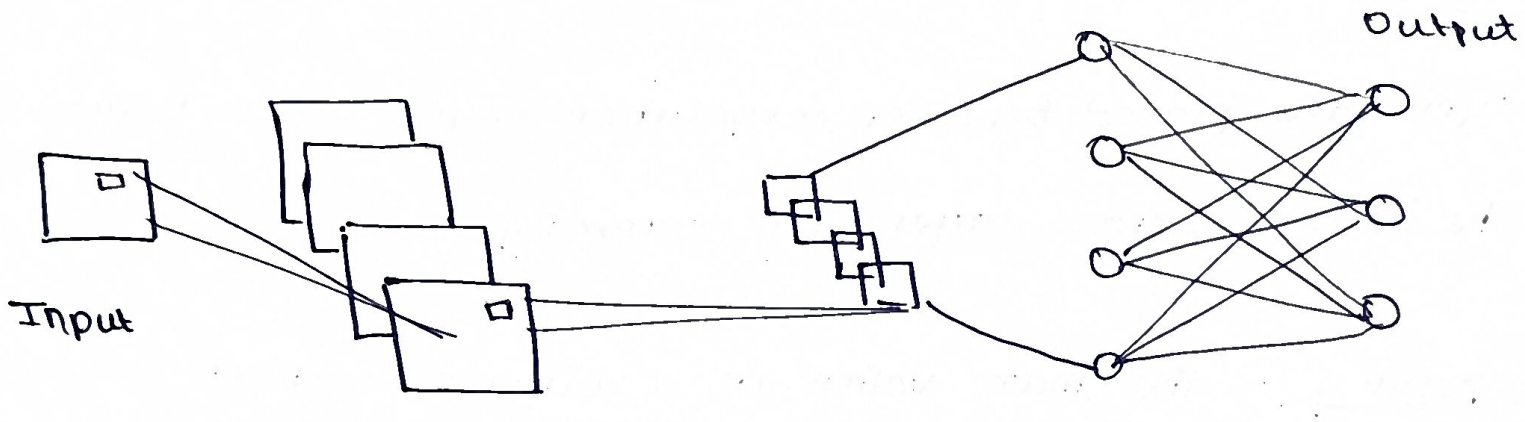
→ A fully connected layer with n input dimensions and m output dimensions is defined as:

W = a weight matrix w/ m rows & n columns
b = bias vector

→ Given an input vector x, the output of a FC layer with an activation function f is:

$$FC(x) = f(Wx+b)$$

→ FC layers are used as the final layers in classification problems.



* Challenges with CNNs

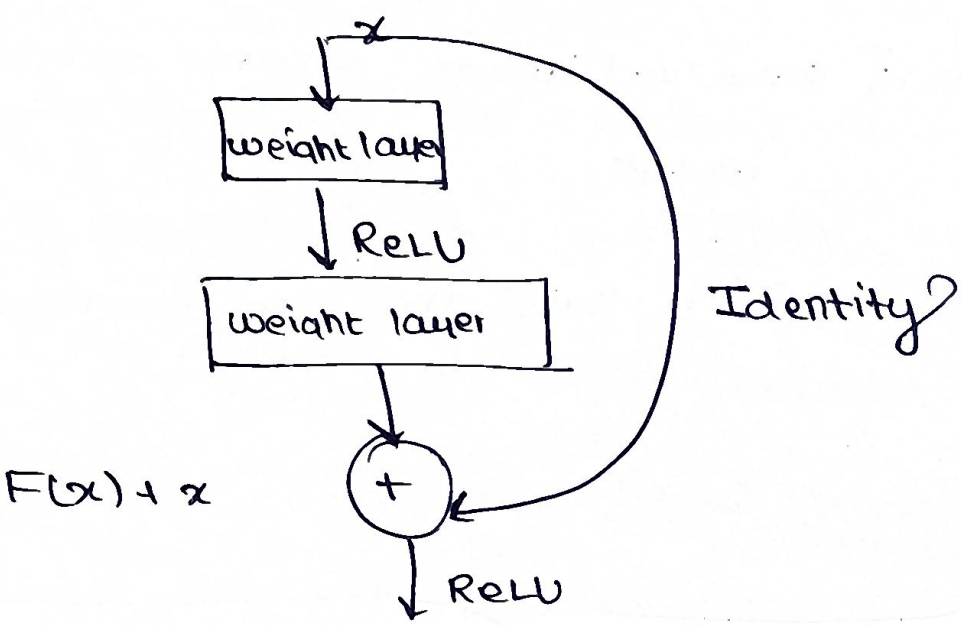
1. Exploding / Vanishing gradients - solve by normalized weight initiation and intermediate normalization layers like Batch Normalization

2. Performance Degradation with Depth: as the depth of a FFN increases, both testing & training accuracies get saturated and degrade afterwards.

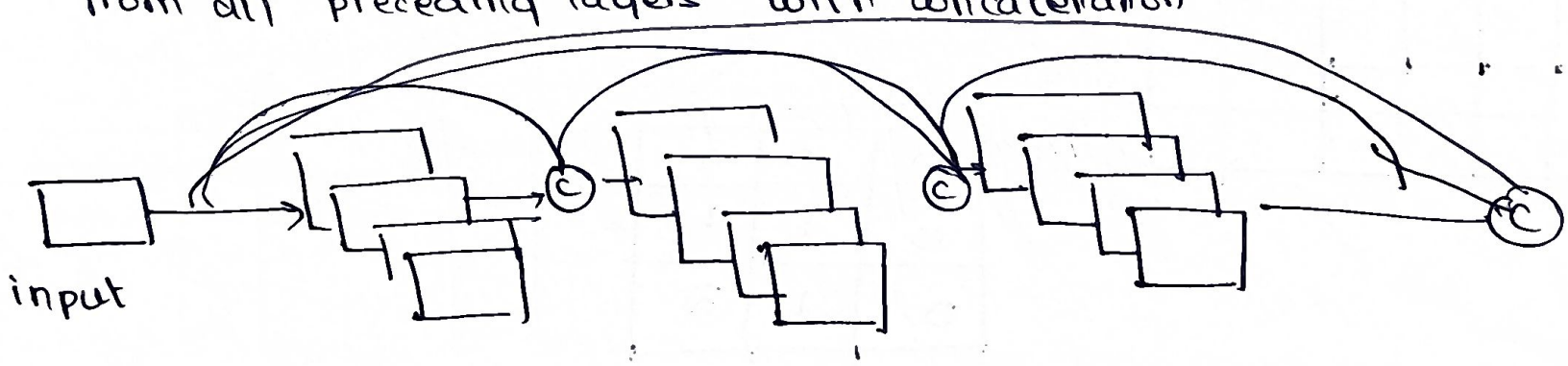
→ This is solved by using skip connections - which carry information directly from earlier layers into later layer without passing through intermediate convolution layers.

→ This helps in preventing information from washing out.

For eg. (i) **ResNet** deals with this by introducing residual blocks



(ii) **DenseNet** - In DenseNet, each layer obtains additional inputs from all preceding layers with concatenation



C = channel-wise concatenation

* Visualization of Kernels

→ Kernels are the parameter convolution layer used to convolve their image.

Kernels have 2 parameters :

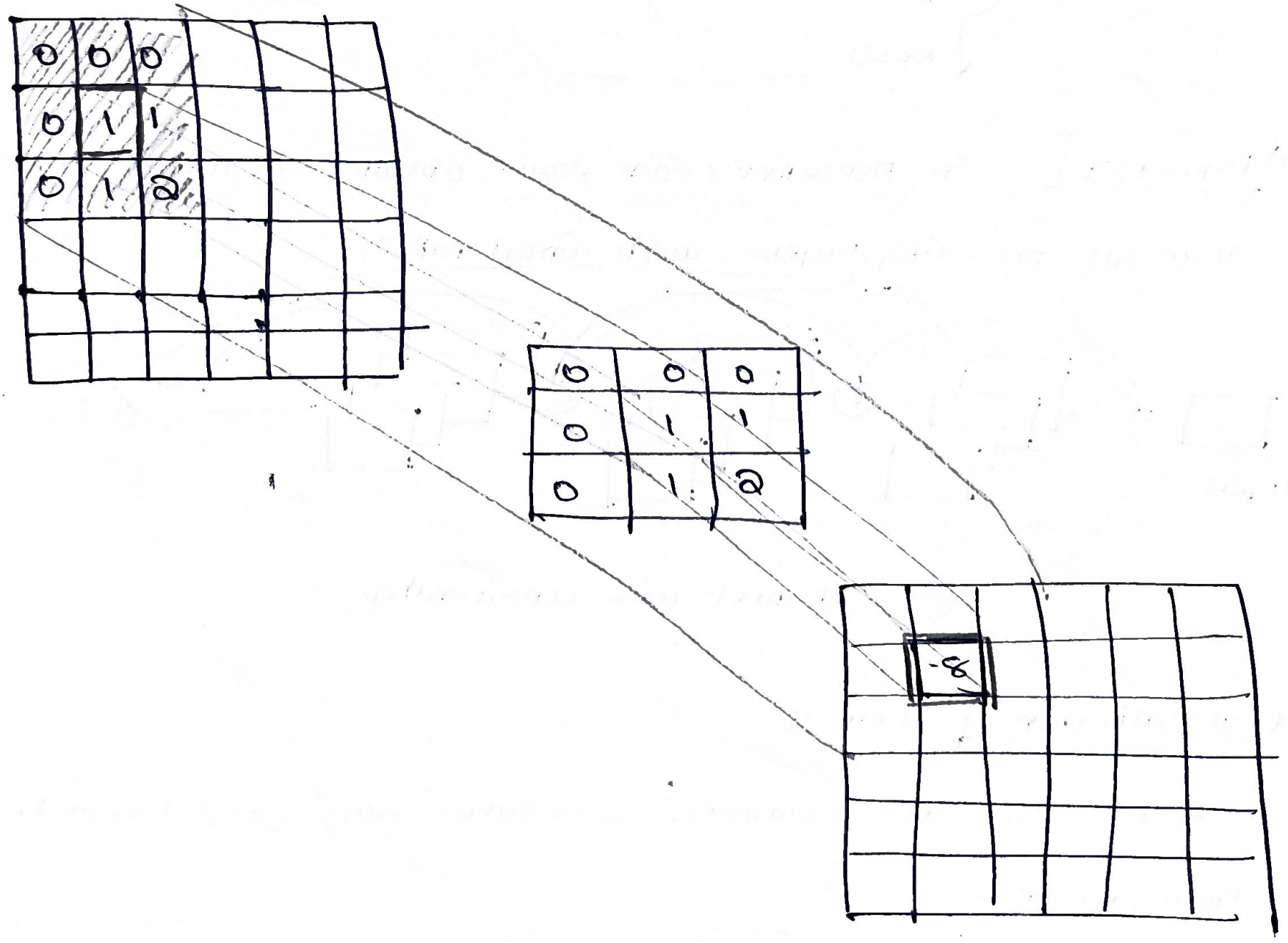
(i) Size = rectangle dimensions

(ii) Stride = amount by which filter slides over an image

Stride = 1 ⇒ move filter 1 pixel horizontally & vertically

⇒ size of convolved image = same as original image

Stride = 2 ⇒ convolution layer is half the input dimension



* CNNs for Object Detection

A. R-CNNs

→ CNNs with their fully connected layer are not capable of dealing with the frequency of occurrence of objects (i.e multiple objects)

→ Region-based CNN (R-CNN) can help deal with this challenge of object detection.

→ The major steps in RCNN are:

(i) Generate Region Proposals

(ii) Feature Extraction with CNN

(iii) Classification with SVM

(iv) Localization with Bounding Box Regressor

Step1: Generate Region Proposals

→ Region proposals = regions in an image that could belong to a particular object.

→ This is done using the selective-search algorithm.

→ The selective search algorithm works by generating sub-segmentations of the image that could belong to one object - based on color, texture, size and shape - and iteratively combining similar

regions to form objects

→ The original algorithm generates 2000 region proposals, with bounding boxes.

Steps : 1. Generate initial sub-segmentations of the image
2. Combine similar bounding boxes into larger ones recursively
3. Use the larger boxes for region proposals

Step 2: Feature Extraction with CNNs

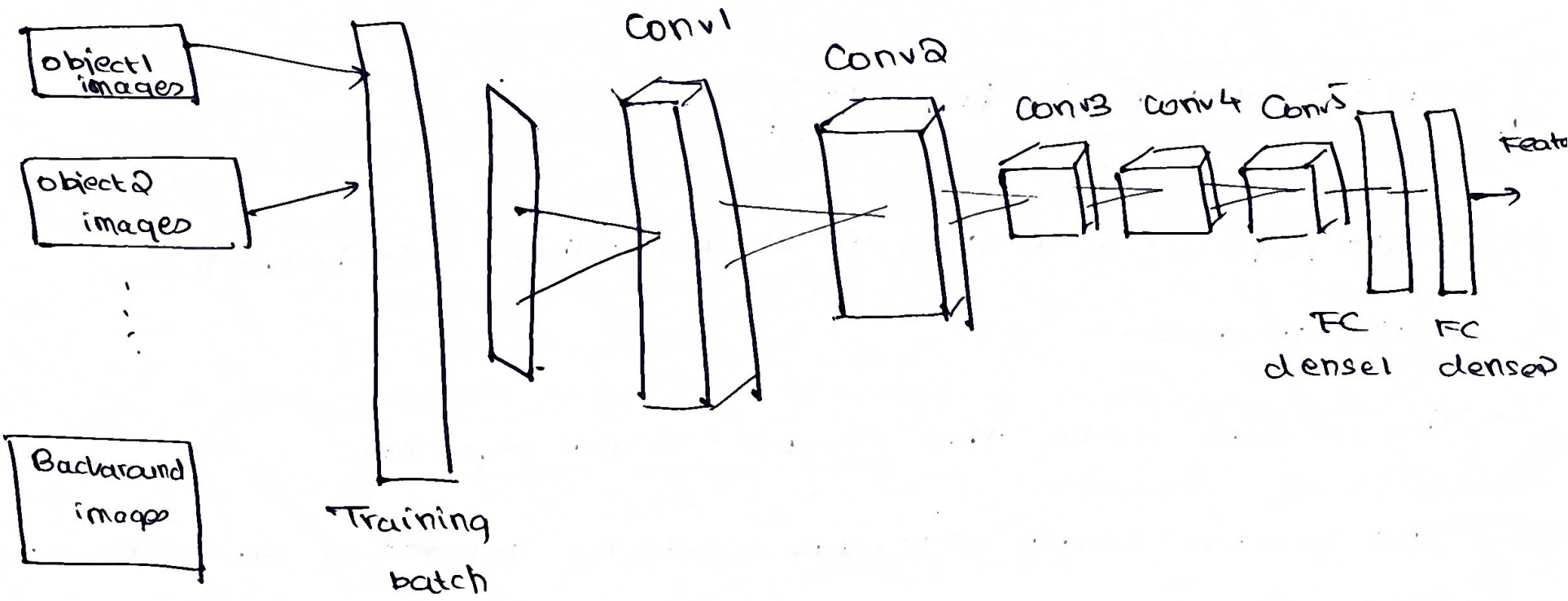
→ A feature vector is generated from each of the 2000 region proposals using a CNN.

→ The CNN used is AlexNet with 5 convolutional layers and 2 fully connected layers.

→ The network is fine-tuned to:

(a) learn the visual features within the region proposals

(b) learn specific target classes for the detection task.



Step 3: classification with SVM

→ The feature-vector for each region proposal is fed into an SVM, which is trained for ~~the~~ each class independently.

→ It is a binary SVM - where features of all region proposals that have an IoU overlap of less than 0.3 w/ the ground truth are considered negatives for that class.

Issue: cannot train AlexNet & SVM independently in a parallel manner

Step 4: Bounding Box Regressor

→ In order to precisely locate the bounding box in the image, a scale-invariant linear regression model called a bounding box regressor is used

→ The predicted & ground truth values are taken in tuples of 4 dimensions (x, y, w, h)

- x, y: pixel coordinates of the center of bounding box
- w, h: width & height of bounding box

→ This helps increase the Mean Average Precision (MAP)

B. Fast R-CNN

→ Developed to handle challenges posed by R-CNN namely,

(i) Each part of R-CNN training (CNN, SVM, bounding box) requires separate training and cannot be parallelled.

(ii) Every region proposal needs to passed through a DCNN - very time consuming

Architecture

1. The Fast R-CNN consists of a CNN (pretrained on ImageNet) with its final pooling layer replaced by an RoI pooling layer and the final FC layer replaced by (i) a softmax layer (ii) a bounding box regression box.

2. The entire image is fed into the backbone CNN and the features from the last convolution layer are obtained. The output feature maps are much smaller than the original image size.

3. Meanwhile, object proposal windows are obtained from the image using a region proposal algorithm - selective search.

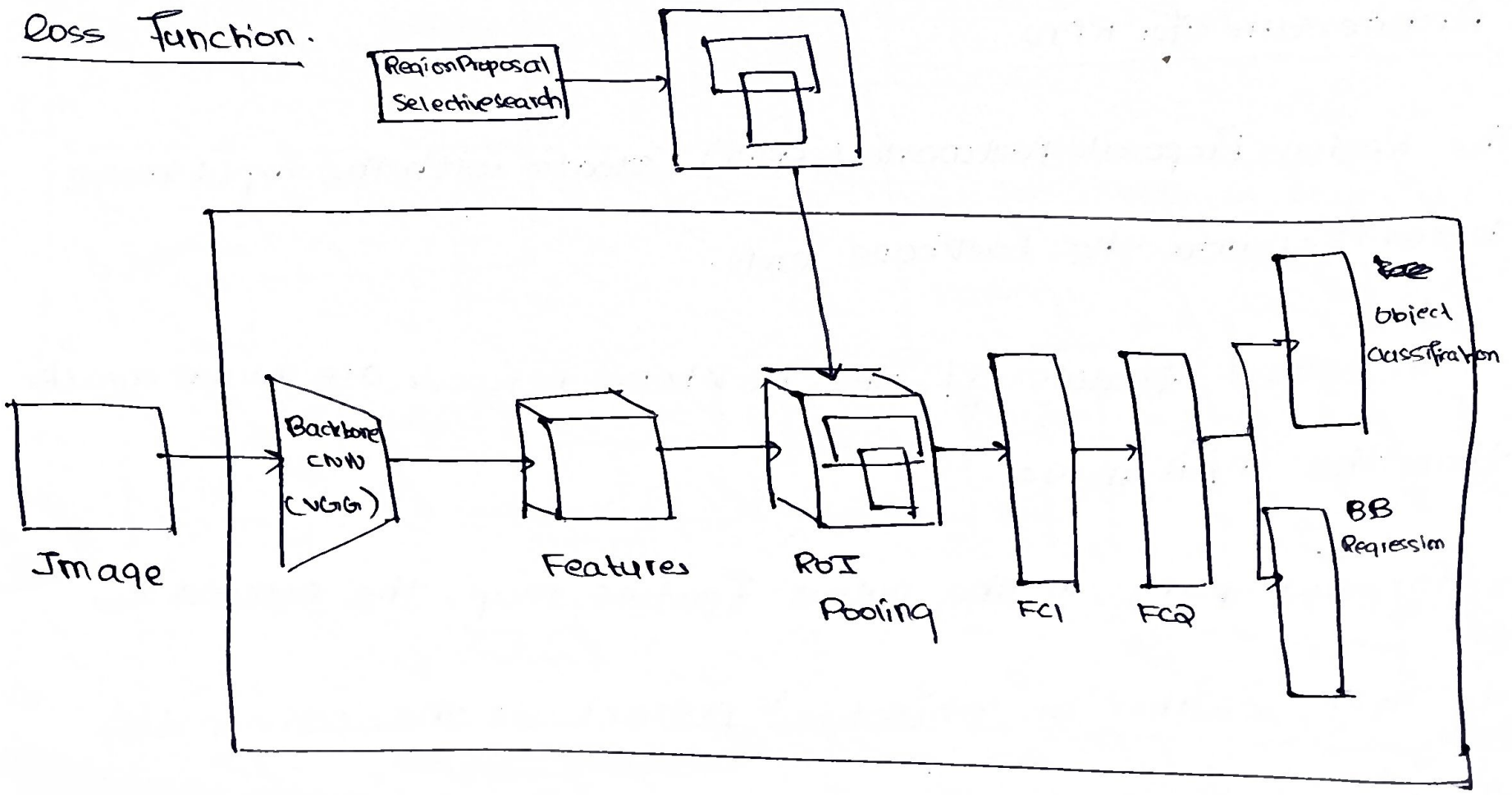
4. The portion of the backbone feature map that belongs to the window is fed into the RoI Pooling Layer.

→ The RoI pooling layer is a special case of the Spatial Pyramid Pooling (SPP) layer with just one layer. The layer divides features from the selected proposal windows (from the region proposal algorithm) into sub-windows of size h/H and w/W and performs a pooling operation in each of these sub-windows.

→ The output features are of dimension H×W are fed into successive FC layers, and then the Softmax and BB-regression branches.

→ The softmax produces probability values of each RoI belonging to K categories

→ The loss is calculated from the softmax probabilities with a log-loss function.



c. **Faster R-CNN**

→ The major changed introduced in Faster R-CNN is the change of the region proposal algorithm.

→ Both R-CNN and Fast R-CNN use the selective search algorithm.

→ The Faster R-CNN model addresses this by using another CNN called the RPN to generate the region proposals.

→ The benefits of this are:

(i) Reduces Region Proposal Time

(ii) Allows the region proposal stage to share layers with the following detection stages, causing an overall improvement in feature representation.

Architecture for RPN

→ The Region Proposal Network (RPN) starts with the input image being fed into the backbone CNN.

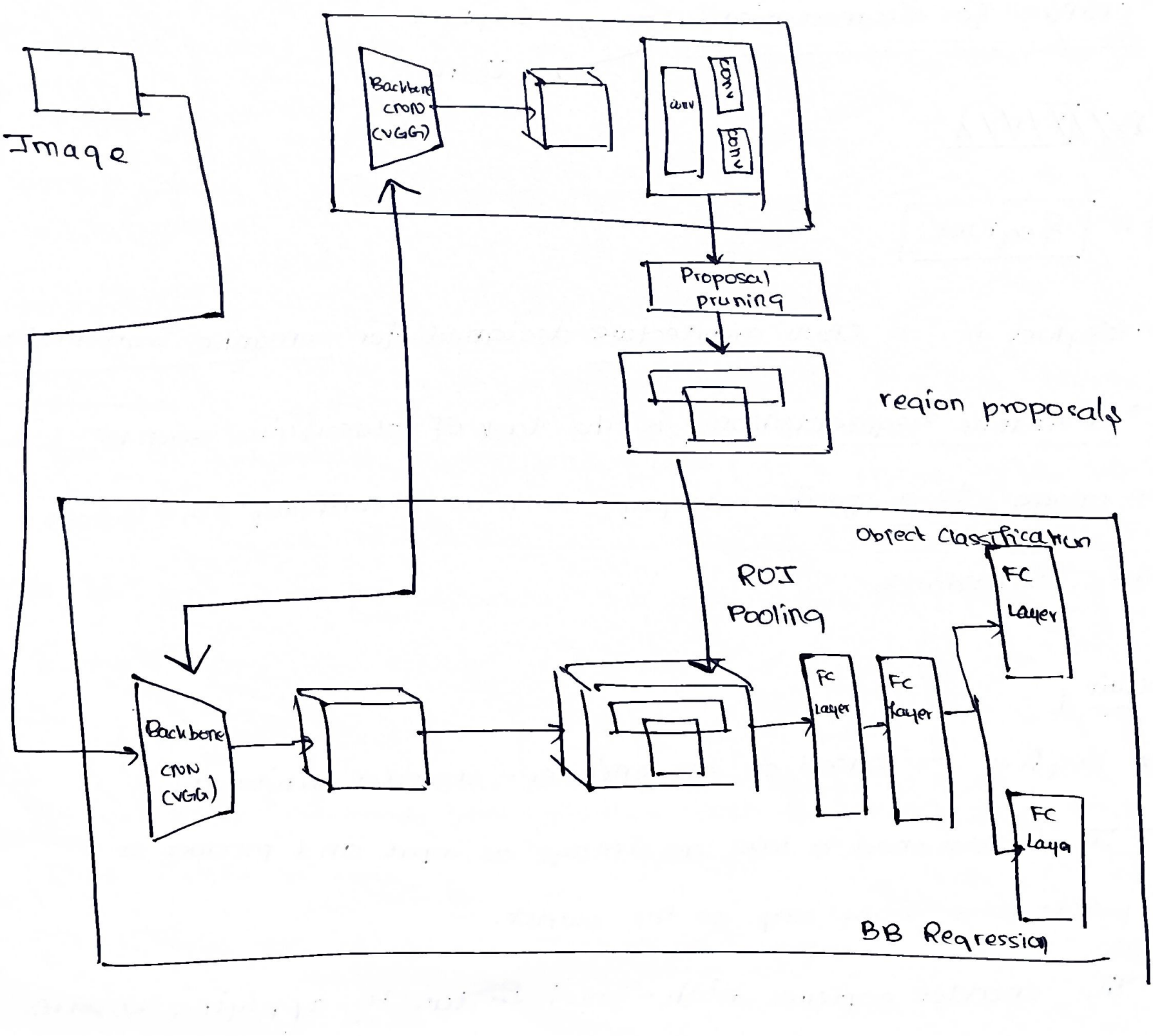
→ The output features of the backbone network are much smaller than the input image.

→ For each point in the output feature map, the network has to learn whether an object is present at the corresponding locations.

→ This is done by placing anchors on the input image. These anchors indicate possible objects in various sizes & aspect ratios.

→ For each pixel, the network checks if k corresponding anchors actually contains object, and then iteratively refines the anchor's coordinates to give bounding boxes as object proposals.

Object Detection Architecture



→ In order to force the networks to share weights, the following approach is used:

① Train the RPN independently

② Train the Fast-RCNN independently. Fix RPN weights & train Faster-RCNN

③ Initialize RPN w/ weights from the Faster R-CNN, and fine-tune the Faster-RCNN again w/ the new RPN weights

* CNNs for Segmentation → FCN, SegNet, U-Net

A. SegNet

→ SegNet is a CNN architecture designed for semantic segmentation

→ Semantic Segmentation is the task of classifying each pixel in an image to a specific category, such as identifying objects & their boundaries

Working

→ SegNet is based on an encoder-decoder architecture

→ It is designed to take an image as input and produce a pixel-wise label map as the output.

→ The encoder captures high-level features by applying a series

of convolutional and pooling layers.

→ The key importance of SegNet is in its decoder network, which uses the spatial pooling indices generated during the max-pooling in the encoder phase ~~to~~ up. — This is memory efficient and helps preserve fine-grained details.

Advantages of SegNet

→ computationally efficient due to the re-use of pooling indices

→ can be trained with limited labelled data using pre-trained models

→ produces high-resolution segmentation maps

Architecture

Encoder

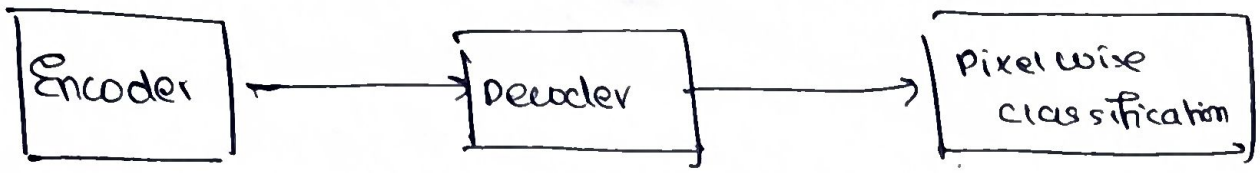
→ has 13 convolutional layers which correspond to those in the VGG16 network.

→ The FC layers are discarded so as to retain higher resolution feature maps.

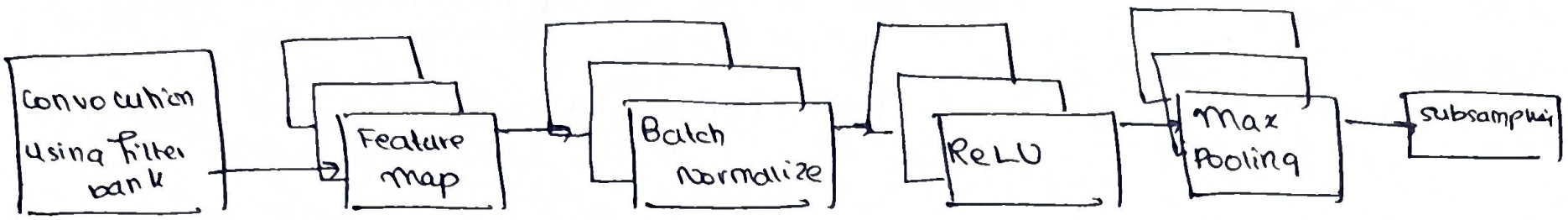
Decoder

→ uses memorized max-pooling indices from the corresponding feature maps during encoder.

→ convolve with a decoder filter bank to produce a dense feature map



Encoder



B. U-Net

→ U-Net is a fully convolutional neural network that does semantic segmentation, using fewer training samples

→ It is a U-shaped encoder-decoder architecture, that consists of four encoder blocks and four decoder blocks that are connected via a bridge

Encoder

→ The encoder network acts as the feature extractor and learns an abstract representation through each encoder block

→ Each encoder block consists of two 3X3 convolutions, where each convolution is followed by a ReLU (for non-linearity)

→ The output of the ReLU acts as a skip connection for the corresponding decoder block

→ Then there is a 2x2 max-pooling layer, where the spatial dimensions are reduced by half. (helps reduces computational cost by decreasing trainable parameters)

Skip-Connections

→ provides additional info to decoder

→ helps in the better flow of gradient during backpropagation, which helps the network learn better representation

Bridge

→ connects the encoder & decoder network and completes the flow of info.

→ has 2 3x3 convolutions followed by a ReLU activation fn

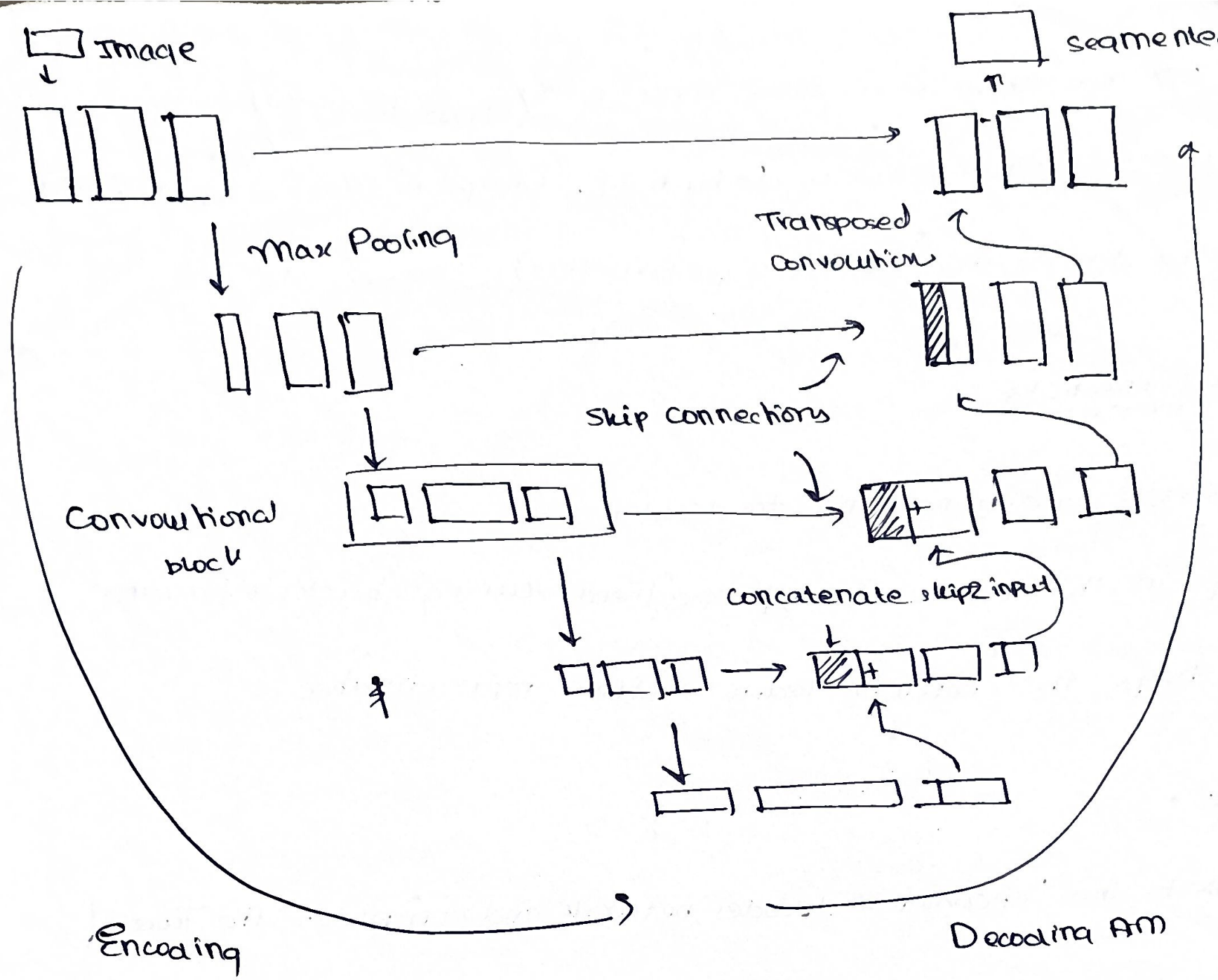
Decoder Network

→ Takes the abstract representation and generates a semantic segmentation task.

→ It first uses a 2x2 transpose convolution, and is then concatenated w/ the skip connection from the encoder block.

→ Then there are 2 3x3 convolutions and the ReLU activation.

→ The output of the decoder passes through a 1x1 convolution w/ sigmoid activation for pixel-wise classification



Am

* RNNs — from NLP

* Segmentation using FCN → used for semantic segmentation

Input — an image of arbitrary size is provided to the network as input

Convolutional layers — The input image passes through a series of convolutional layers. They extract hierarchical features - and capture spatial & contextual information

Pooling layers downsample the feature maps, reducing the spatial dimensions while preserving imp features.

Fully convolutional Transformation

→ Unlike traditional CNNs, FCNs replace the fully connected (dense) layers with convolutional layers. This allows the network to handle input images of varying sizes

Upsampling → The downsampled feature maps are upsampled using transpose convolution (deconvolution)

→ This restores the spatial resolution to match the input image size.

Skip-connections

Pixel-wise classification → The final upsampled feature map passes through a 1×1 convolution layer to reduce the number of chanels to the no. of classes (classify w/ softmax)

Output Segmentation → dense classification map

* ~~Ext~~ Mask R-CNN

→ Built on top of faster R-CNN to allow for instance/ semantic segmentation.

→ all remains the same as the faster R-CNN, but there is an additional ConvNet that takes an RoI as input & outputs the m × m mask representation

→ This mask is upscaled for inference on the input image