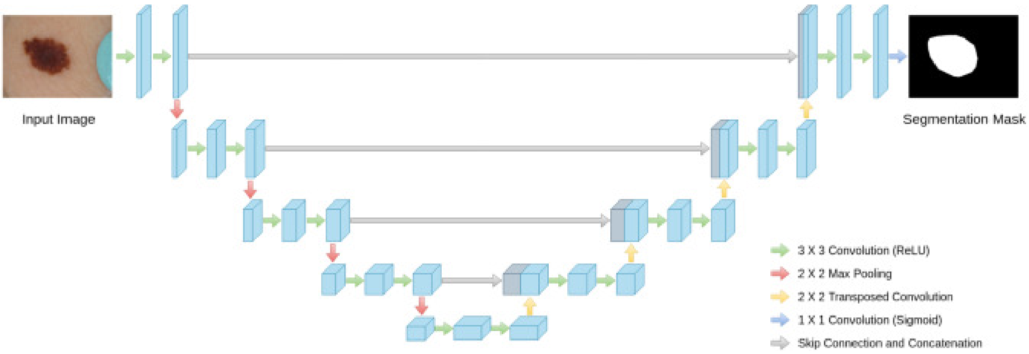
Segmentation of MILA Logo

Methodology - 1 (using modified U-Net)


Input Image
Segmentation Mask
3 X 3 Convolution (ReLU)
2 X 2 Max Pooling
2 X 2 Transposed Convolution
1 X 1 Convolution (Sigmoid)
Skip Connection and Concatenation

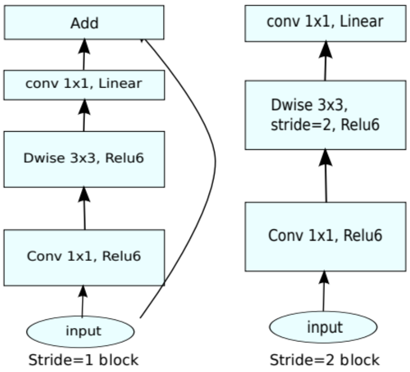
Source: https://www.sciencedirect.com/science/article/pii/S0893608019302503#fig1

In this problem, we chose a modified U-Net architecture. The UNet architecture mainly has two main parts, that are encoder and decoder. The encoder is all about the covenant layers followed by pooling operation. It is used to extract the factors/features in the image. The second part decoder uses transposed convolution to permit localization. For the encoder we use a standard backbone pretrained feature extractor, while for the decoder, we create a few upsample stacks of 2D convolutional networks. The architecture of a U-Net resembles like the above figure.

Objective function: The number of pixels representing MILA-logo is minimal compared to those in the background resulting in class imbalance. The binary cross-entropy loss assigns equal weight to minority and majority class due to which it performs poorly for imbalanced data-set. The focal loss is a better option as it adequately assigns weights to the two different classes. It is meant to help ease prediction when there's a lot of data in one class and sparse data in another (foreground or background). Focal loss is helpful when there's a ratio of [1:100] or more for the imbalanced classes.

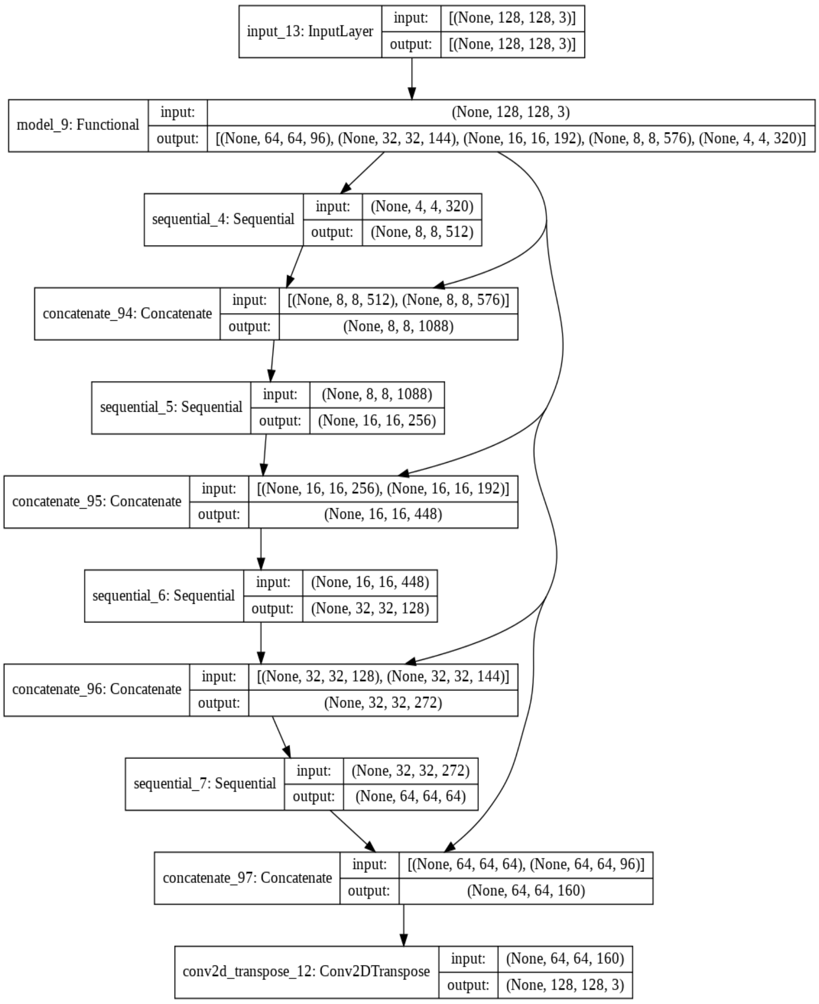
Model Architecture

In this problem statement, we have a set of natural images which have been inked with the MILA logo. To capture the natural image statistics, we use a backbone feature extraction network, which has been ImageNet pretrained. For this purpose, we choose the MobilenetV2 as the baseline architecture. The following figure shows the architecture of MobilenetV2.



Baseline architecture: MobileNetV2 is based on an inverted residual structure where the residual connections are between the bottleneck layers. It is introduced to dramatically reduce the complexity, cost and model size of the network, making it suitable for mobile devices, or any devices with low computational power. The intermediate expansion layer uses lightweight depthwise convolutions to filter features as a source of non-linearity. As a whole, the architecture of MobileNetV2 contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers.

Modified-U Net: Following this, the model being used here is a modified U-Net. A U-Net consists of an encoder (downsampler) and decoder (upsampler). In-order to learn robust features, and reduce the number of trainable parameters, a pretrained model can be used as the encoder. Thus, the encoder for this task will be a pre-trained MobileNetV2 model, whose intermediate outputs will be used, and the decoder will be the upsample block. We use two output channels in this task. One class being the background and one being the MILA logo mask. The reason to output two channels is because there are two possible labels for each pixel, making it a binary classification problem for the final segmentation, where each pixel is being classified into two classes. The model consists of certain skip connections. Skip connections are extra connections between nodes in different layers of a neural network that skip one or more layers of nonlinear processing. The final architecture of our modified U Net looks like the following:



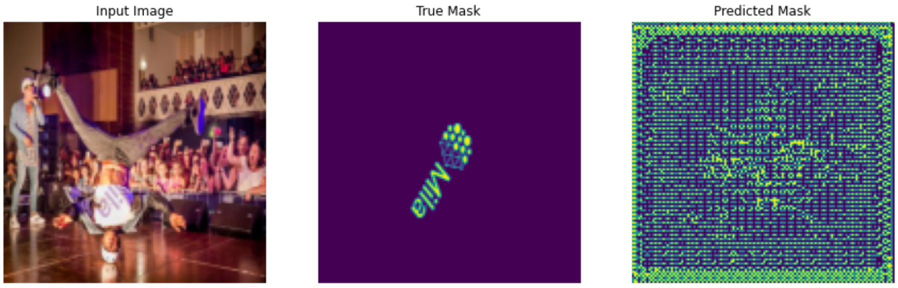
input_13: InputLayer
input: [(None, 128, 128, 3)]
output: [(None, 128, 128, 3)]
model_9: Functional
input: (None, 128, 128, 3)
output: [(None, 64, 64, 96), (None, 32, 32, 144), (None, 16, 16, 192), (None, 8, 8, 576), (None, 4, 4, 320)]
sequential_4: Sequential
input: (None, 4, 4, 320)
output: (None, 8, 8, 512)
concatenate_94: Concatenate
input: [(None, 8, 8, 512), (None, 8, 8, 576)]
output: (None, 8, 8, 1088)
sequential_5: Sequential
input: (None, 8, 8, 1088)
output: (None, 16, 16, 256)
concatenate_95: Concatenate
input: [(None, 16, 16, 256), (None, 16, 16, 192)]
output: (None, 16, 16, 448)
sequential_6: Sequential
input: (None, 16, 16, 448)
output: (None, 32, 32, 128)
concatenate_96: Concatenate
input: [(None, 32, 32, 128), (None, 32, 32, 144)]
output: (None, 32, 32, 272)
sequential_7: Sequential
input: (None, 32, 32, 272)
output: (None, 64, 64, 64)
concatenate_97: Concatenate
input: [(None, 64, 64, 64), (None, 64, 64, 96)]
output: (None, 64, 64, 160)
conv2d_transpose_12: Conv2DTranspose
input: (None, 64, 64, 160)
output: (None, 128, 128, 3)

Experimental Setting

In this section, we discuss in detail the training and the experimental protocols that have been used. We perform our experiments on the natural image data consisting of multiple MILA logos, which have been made to undergo certain translations. For our experiments, we choose randomly 90% of the image instances as the training data and the remaining 10% as the validation data. The masks corresponding to each image have 0/1pixel labels implying the presence or absence of a logo pixel. The original images and masks were resized to 128 x 128 to fit the pretrained network architectures.

To pre-train the network to get sufficiently discriminative initialization weights, we use a pretrained MobileNetV2 framework. The network was trained using an Adam optimizer, and a learning rate of 0.001 was chosen. Following this, the model being used here is a modified U-Net. A U-Net consists of an encoder (downsampler) and decoder (upsampler). The

Mobilenetv2 acts as the downsampler, while for the upsampler we use 512, 256, 128, and 64 channels with ReLU activation function and a kernel size of 3 x 3. 3 x 3 was chosen as the kernel size as the logo of MILA Is very small and a higher dimensional kernel might not be able to capture it well. Adam optimizer with a stochastic mini-batch gradient descent approach. We trained the model with a batch size of 16 for 50 epochs. Finally, the evaluation metric used here is 'Intersection of Union' (IoU) for reporting the performance. The following figure shows the prediction of the mask of a sample image. It can be noted that while the predicted mask is very noisy, the gradients eventually start focusing around the region where the logo is located. In order to train the model, the computational resources from Google Colab with driver version 460.32.03 and CUDA version: 11.2. The implementation was done using TensorFlow 2.0 and Keras.

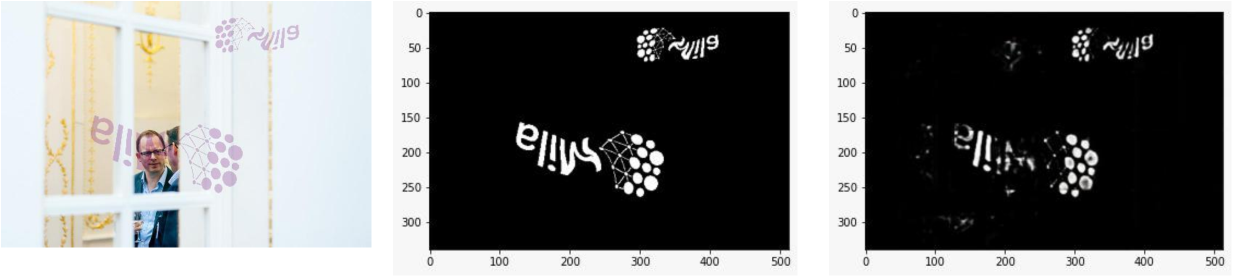


Improved Methodology - 2 (using attended-autoencoders)

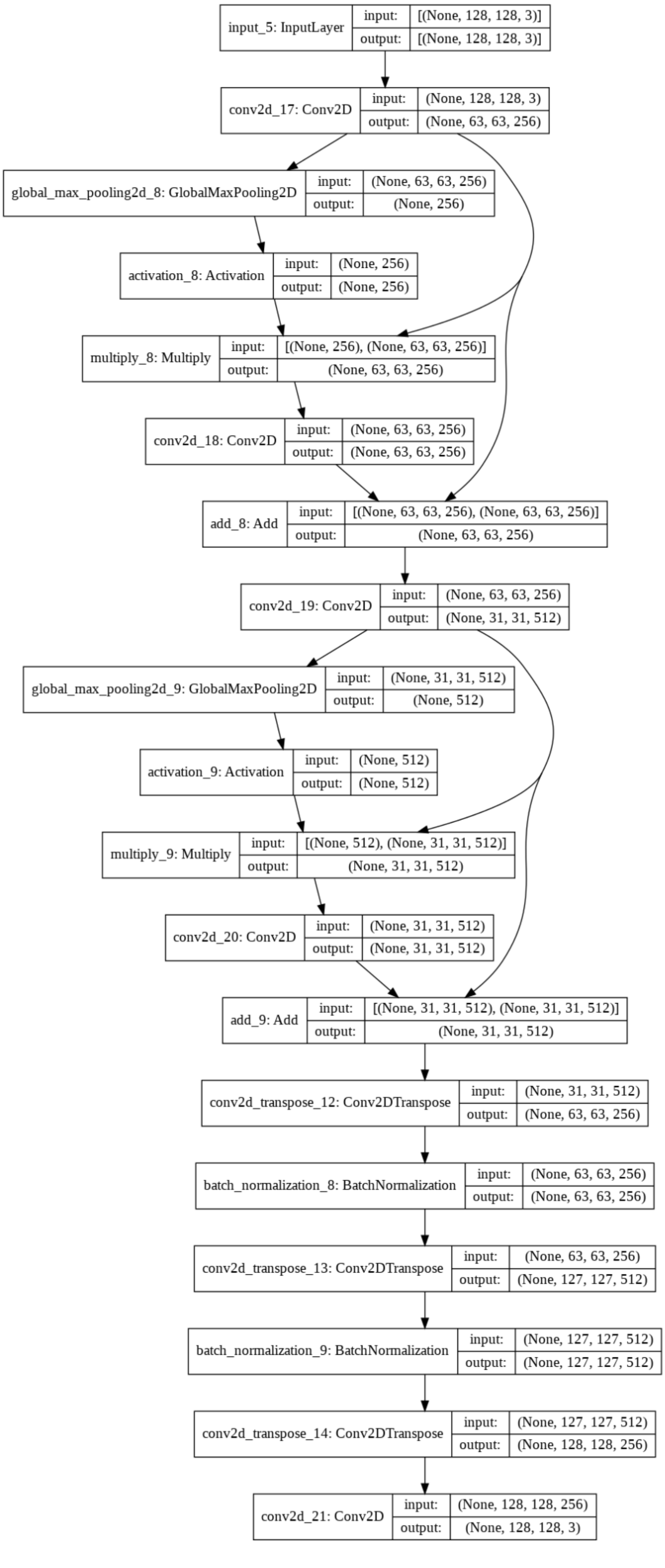
In this problem, we chose an attention based autoencoder type of architecture. This is a very simple architecture using auto-encoders. Each layer of autoencoder is modified with soft attention weights. It consists of 2-layers of 2D convolutions with sigmoid activations for soft attention weights. We also use batchnorm layers intermittently to make the learning. It reduces the dependence of gradients on the scale of the parameters or their initial values. It also regularizes the model and reduces the need for dropout, and other photometric distortions, local response normalization and other regularization techniques.

.

Objective function: In this case we are using the binary cross-entropy loss function for predicting the logo mask. The background pixels are labelled as 0 and the logo images are labelled as 1. The following images show the true input, true mask, and the predicted mask for a model which yields IoU value of about ~0.87. The fine horizontal and vertical lines occur due to stitching up the patches post inference phase. This can be removed by image post processing.



In addition, due to lack of time and lack of resources, we trained this model on only a subset of the entire dataset. We loaded **15** images and used just 12 images for training. The remaining 3 images were used for validation purposes. The test dataset was however pooled from the entire remaining image sample from the dataset. However, even with this small subset of the entire 22,470 image dataset, we were still able to achieve good results on the validation/test dataset. If trained on the entire dataset, this model would certainly yield much better performance. Presently, it can be treated as a few-shot segmentation task.



input_5: InputLayer | input: [(None, 128, 128, 3)] | output: [(None, 128, 128, 3)]
conv2d_17: Conv2D | input: (None, 128, 128, 3) | output: (None, 63, 63, 256)
global_max_pooling2d_8: GlobalMaxPooling2D | input: (None, 63, 63, 256) | output: (None, 256)
activation_8: Activation | input: (None, 256) | output: (None, 256)
multiply_8: Multiply | input: [(None, 256), (None, 63, 63, 256)] | output: (None, 63, 63, 256)
conv2d_18: Conv2D | input: (None, 63, 63, 256) | output: (None, 63, 63, 256)
add_8: Add | input: [(None, 63, 63, 256), (None, 63, 63, 256)] | output: (None, 63, 63, 256)
conv2d_19: Conv2D | input: (None, 63, 63, 256) | output: (None, 31, 31, 512)
global_max_pooling2d_9: GlobalMaxPooling2D | input: (None, 31, 31, 512) | output: (None, 512)
activation_9: Activation | input: (None, 512) | output: (None, 512)
multiply_9: Multiply | input: [(None, 512), (None, 31, 31, 512)] | output: (None, 31, 31, 512)
conv2d_20: Conv2D | input: (None, 31, 31, 512) | output: (None, 31, 31, 512)
add_9: Add | input: [(None, 31, 31, 512), (None, 31, 31, 512)] | output: (None, 31, 31, 512)
conv2d_transpose_12: Conv2DTranspose | input: (None, 31, 31, 512) | output: (None, 63, 63, 256)
batch_normalization_8: BatchNormalization | input: (None, 63, 63, 256) | output: (None, 63, 63, 256)
conv2d_transpose_13: Conv2DTranspose | input: (None, 63, 63, 256) | output: (None, 127, 127, 512)
batch_normalization_9: BatchNormalization | input: (None, 127, 127, 512) | output: (None, 127, 127, 512)
conv2d_transpose_14: Conv2DTranspose | input: (None, 127, 127, 512) | output: (None, 128, 128, 256)
conv2d_21: Conv2D | input: (None, 128, 128, 256) | output: (None, 128, 128, 3)

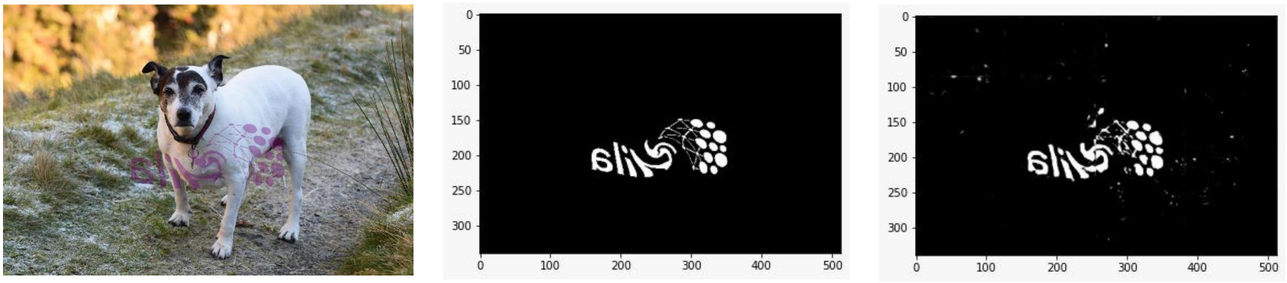
Experimental Setting

However, in order to account for the heavy imbalance in the labels of each image, we make 128 x 128 dimensional random crops of the original 512 x 340 dimensional images. Each of these patches can either contain the logo patch or not. By cropping and then training on these images helps in reducing the imbalance to quite an extent during training. Finally, we stitch up these 128 x 128 patches to form a 512 x 512 dimensional image and show the predicted mask on this image. The results obtained using this method is much better in comparison to the previous one.

The network was trained using an Adam optimizer, and a learning rate of 0.001 was chosen. We use a stochastic mini-batch gradient descent approach. We trained the model with a batch size of 8 for 50 epochs. Finally, the evaluation metric used here is ‘Intersection of Union’ (IoU) for reporting the performance. The IoU in the validation crops goes upto 0.90 (upper limit being 1.0). The

following figure shows the overall architecture details of the network.
Finally, to comment on how the network could have been improved. If we had more time than 48h, I would have gone for an image co-segmentation [1] based approach to segment the logos. Image co-segmentation is a fundamental problem in vision that aims to jointly identify and segment common objects from a set of images. Co-segmentation involves a group of similar input images to capture foreground segments of interest shared across them from. Such information aggregation across the group further helps to identify the common foreground when it is occluded
or visually ambiguous from the background due to clutter, transformations, rotations, scaling, swirled or flipped. However, it remains yet to be investigated how a cosegmentation based framework would perform with alpha-blends if blended too much with the image (higher alpha values).

Another instance of sample results obtained from methodology 2 (original photo, true mask, predicted mask):



References

[1] Banerjee, Sayan, S. Divakar Bhat, Subhasis Chaudhuri, and Rajbabu Velmurugan. "Directed Variational Cross-encoder Network for Few-shot Multi-image Co-segmentation." In 2020 25th International Conference on Pattern Recognition (ICPR), pp. 8431-8438. IEEE, 2021.