**Need for Video compression**

Due to the evolution of internet technology, popularity of video streaming applications is rapidly growing. Such as 4K videos, need large storage space and network bandwidth. Various efforts are currently in force to compress the videos with various algorithms such as H263, MPEG4, H264 currently entailing significant improvements in coding efficiency, latency, complexity.

**Motivation**

All the video compression algorithms currently used are stated based on the motion estimation or on empirical experimentations. But using them for a modern neural networks wont suit because those algorithms wont work proper with the current application usage. Thus, we need to study much further on those algorithms and see what can be made to improve the efficiency.

**Previous Works**

The previous developments were the usage of Reconstruction CNN (RecCNN) and Compact Representation CNN’s(ComCNN). The ComCNN is used to generate a compact representation of the input image for the encoding, which preserves structural information of the image and therefore facilitates the accurate reconstruction of high-quality images. The RecCNN is used to enhance the quality of the decoded image. These two CNNs collaborate with each other and are optimized simultaneously to achieve high-quality image compression at low bit rates.

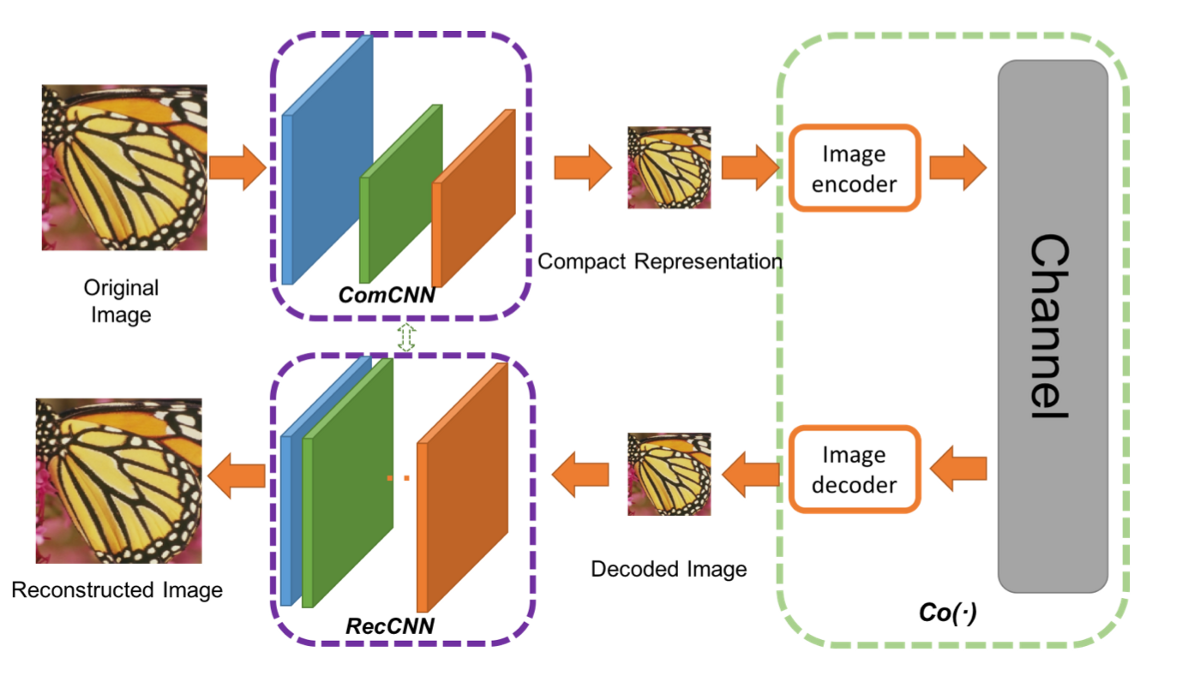
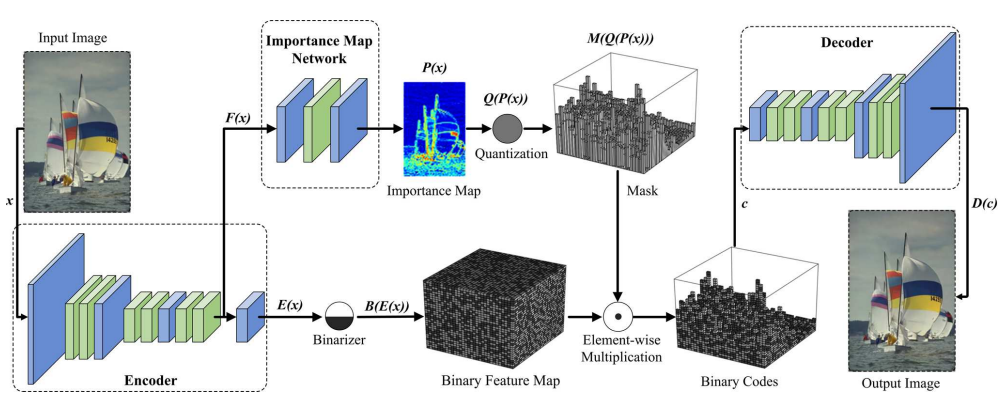


Figure1 : RecCNN and ComCNN Architecture

**Content Weighted Image Compression**

Many of the functions used in the above algorithms are non-differentiable, mainly towards the quantization part and thus makes it challenging to develop a CNN based image compression system. With this motivation, the Content weighted image compression proposed that the bit rate of the different parts of the image should be adapted to local content. And the content aware bit rate is allocated under the guidance of a content weighted importance map. Thus, the sum of the importance map can serve as a continuous alternative of discrete entropy estimation to control compression rate. And binarizer is adopted to quantize the output of encoder due to the binarization scheme is also directly defined by the importance map. Furthermore, a proxy function is introduced for binary operation in backward propagation to make it differentiable. Therefore, the encoder, decoder, binarizer and importance map can be jointly optimized in an end-toend manner by using a subset of the ImageNet database



**Our Contribution**

With the above structure, we did various experimentations on the encoder architecture. We have used three encoder-based architectures. The main purpose of this experimentation was to see which architecture might give better reconstructed image and improve the quality of video compression.

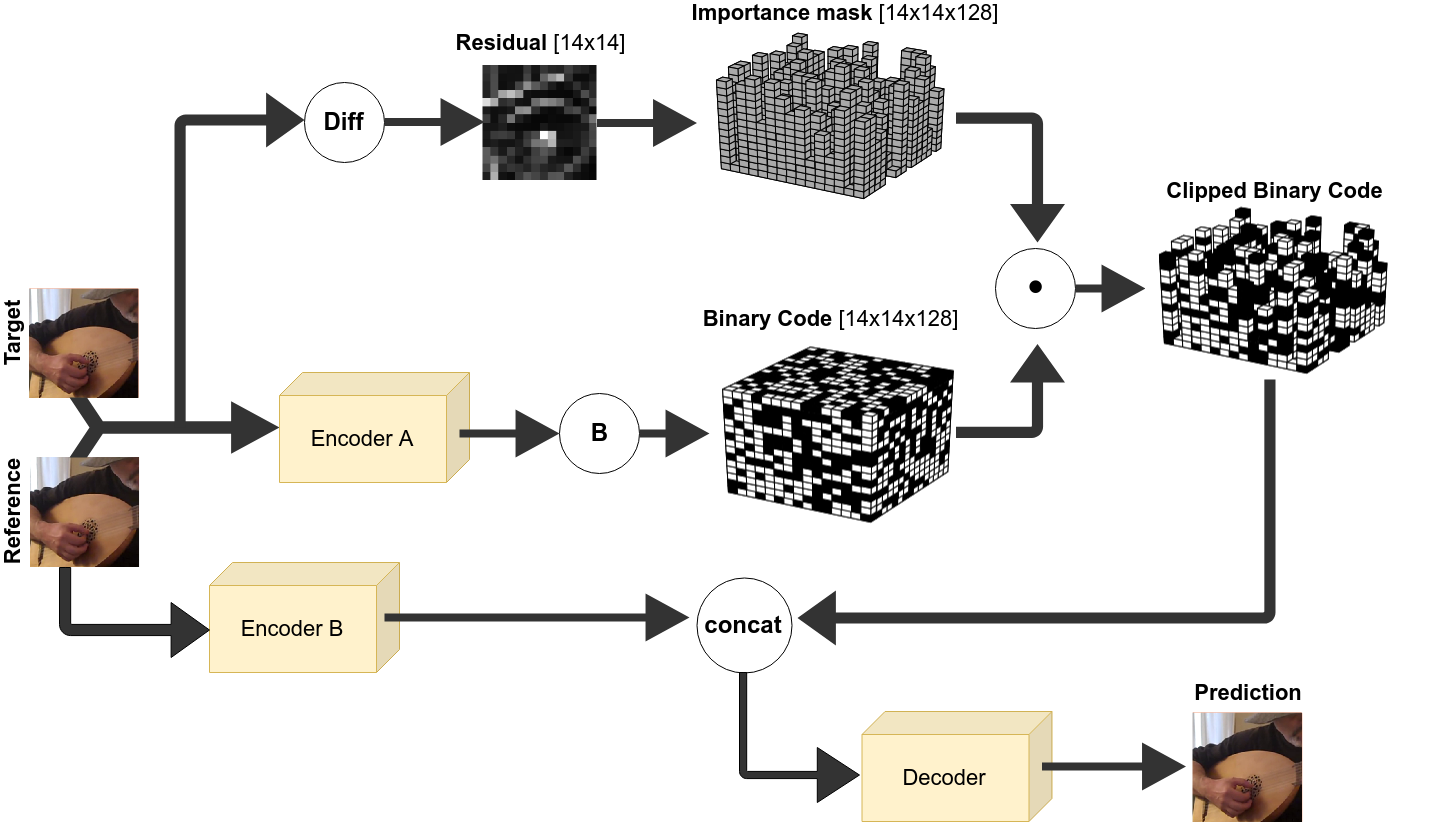
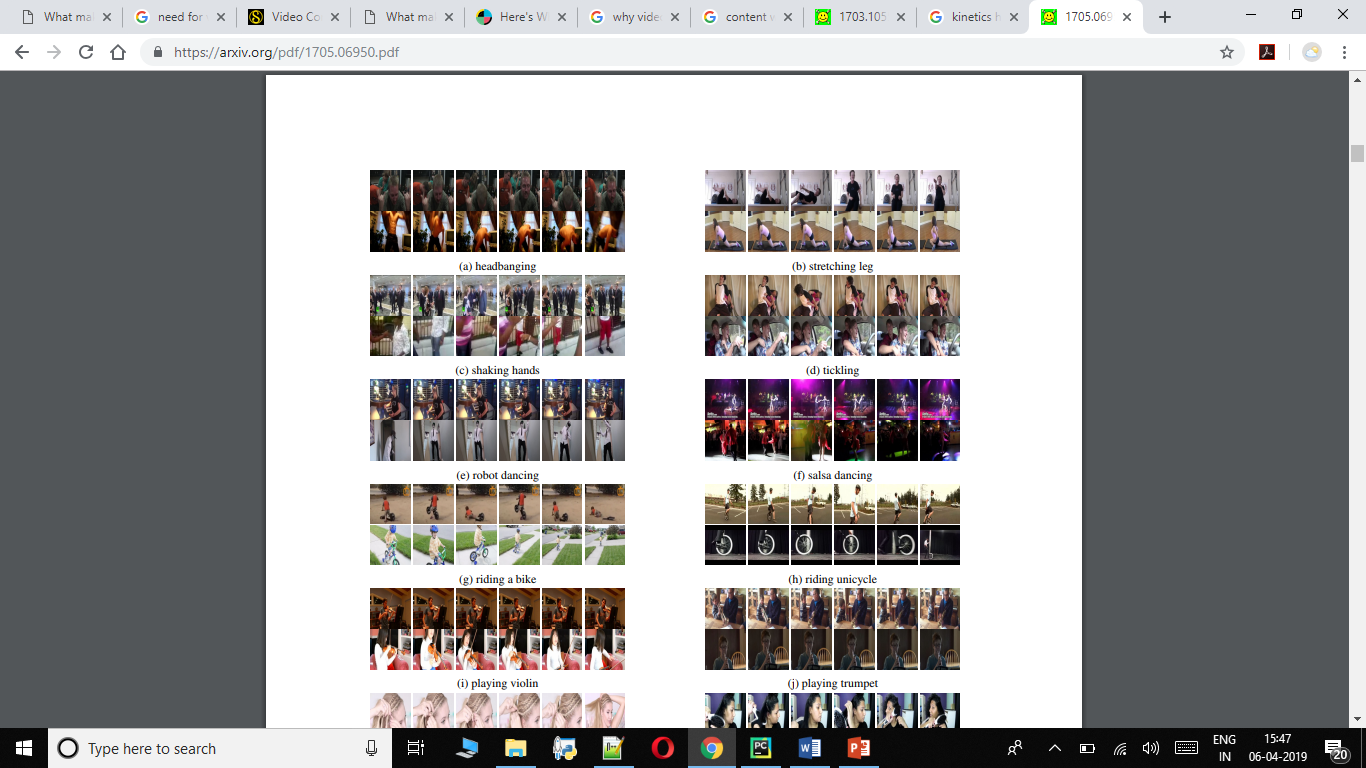


Figure 2: - Architecture of our proposed algorithm in the encoder region.

**Dataset: -** We have used the DeepMind Kinetics Human Action Video Dataset. It contains 500,000 videos taken from YouTube containing high human motions in scene containing 400 human action classes, with atleast 400 video clips for each action. Each clip lasts 10s the motivation to take this dataset was to see the action are human focussed with high interactions such as playing instruments, shaking hands etc.

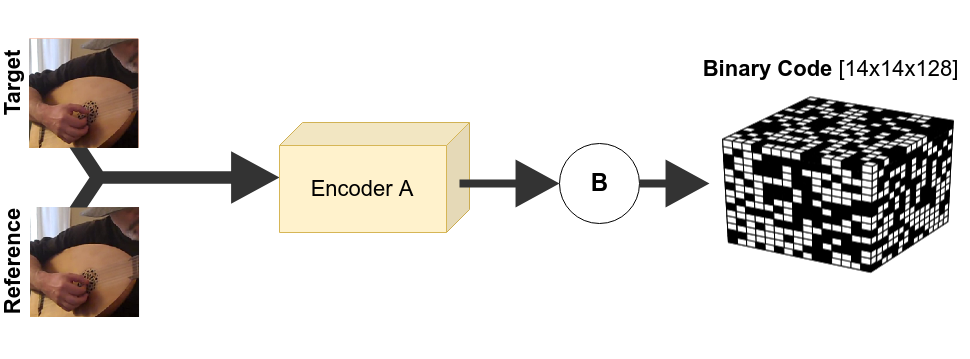
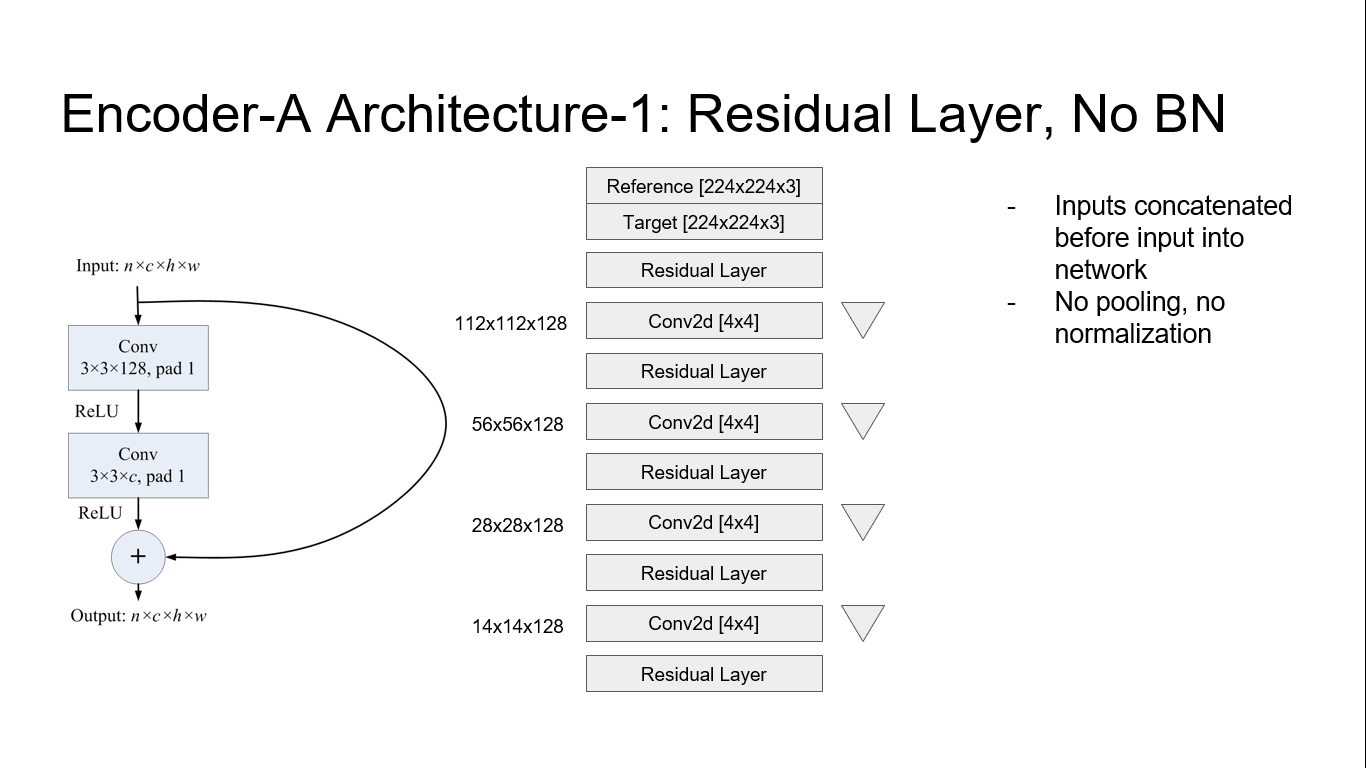
Figure 3: - Sample of the Kinetics Human Action Video Dataset.



Our strategy is to download the video from the dataset and randomly select 5 target frames in the video. To ensure a mix of large and small translations in the dataset we used, we take frames anywhere between 1 to 5. Each of the training and testing sets have 10k samples of target/reference pairs.

1. **Encoder A & Binarizer Architecture**

The purpose of the encoder is to recover the target from the reference through learning a model that generates a 14\*14\*128 feature map. The binarizer converts the feature map into a binary code of shape 14\*14\*128. We have contacted both the inputs before feeding into the network.

 Figure 4,5: - Diagram showing the encoder A and binarizer architecture and the mode used

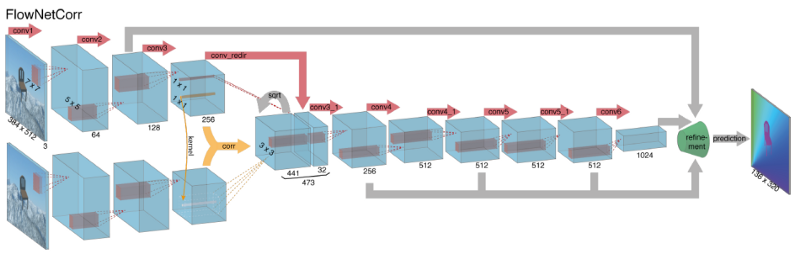
In the network model, we forward the concatenated input with multiple down sampling. We attempted to feed the model as it is with no pooling or batch normalization layers. The output of the models gives the expected 14\*14\*128 dimension.

**Encoder-A Architecture-2: - Using Resnet 50**

The beauty of Resnet 50 is that it is highly successful in Image classification. So, we wanted to use Resnet 50 model and see the improvement in reconstructing the target image. The input for this experiment remains the same (where we previously concatenated target and reference frames), but applied Batch Normalization and a rectifier ReLU.

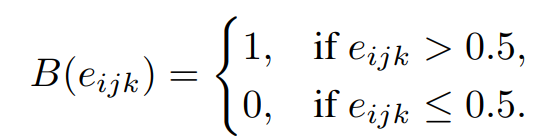
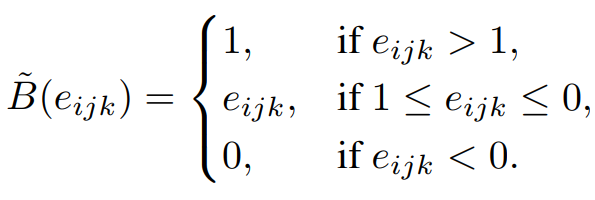
Encoder-A Architecture 3: - FlowNet Correlation

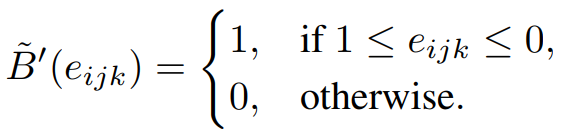
Figure 6:- Diagram of the FlowNet Architecture.



**Binarized Networks: The Differentiation Prob**

The main idea behind using the Binary code is to learn the pixels which potentially need more importance and compress the un important regions to achieve better compression. We used the binarizer which reads the motion between the target and reference frames and identify the regions that can potentially be compressed. However, we face this differentiable problem. It can be solved using the Straight-Through Estimators.





Bits Per Pixel = (14x14x128) / (224x224) = 0.5 bpp

**Importance Mask**

Not all the pixels in the target frame needs attention. So, we introduce an importance mask which gives attentions to only the pixels that were in motion between the target and the reference frame.

We represent a 16\*16\*3 patch in the original 14\*14 image and get the residual of the target and the reference. This residual clearly restricts the pixels that needs more attention and the pixels that doesn’t. With this we achieve a better compression.

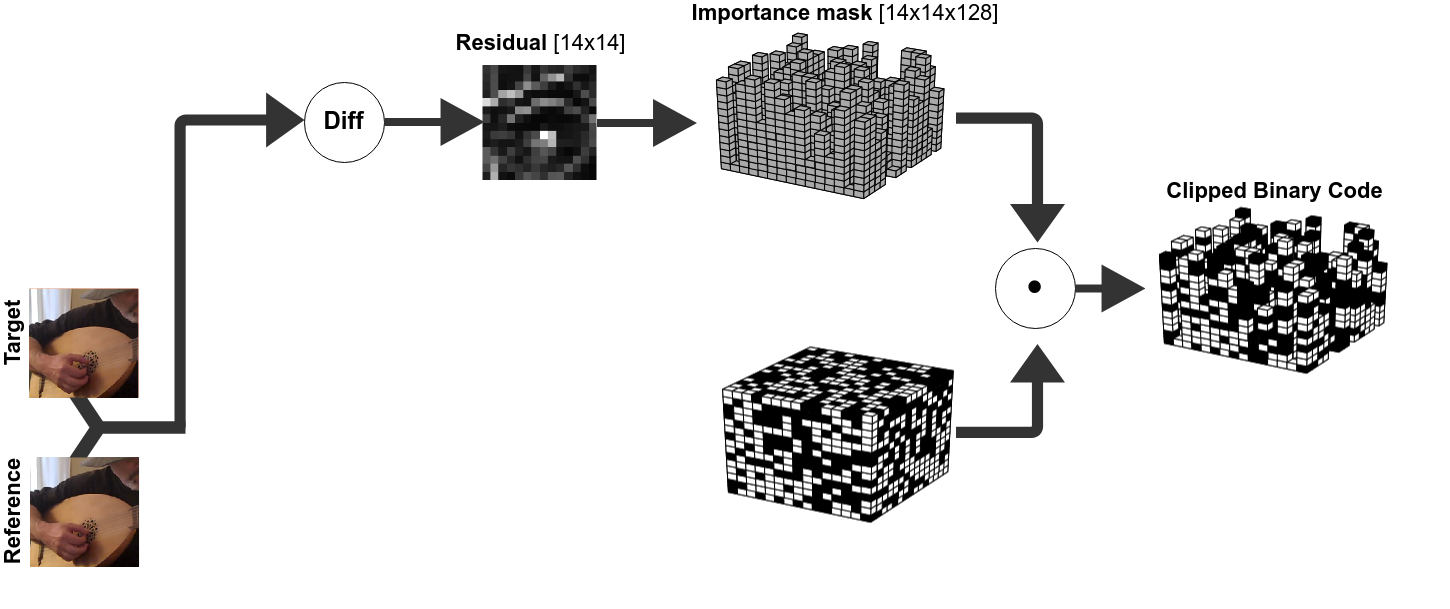
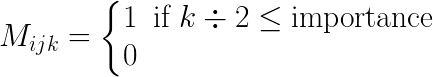


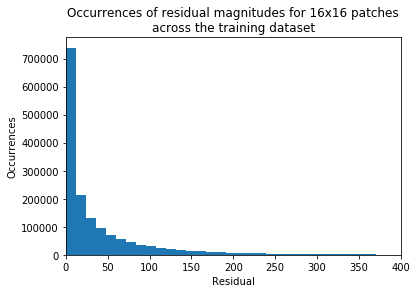
Figure 6:- Representation of the Residual Mask

To sum up precisely, the importance mask is calculated by summing the residual difference between the target and the reference frame.





To view the importance levels on a scale, we arbitrarily assign 64 distinct importance levels. Each importance level represents 2 extra bits of representation. We chose an arbitrary cut-off line at 128.

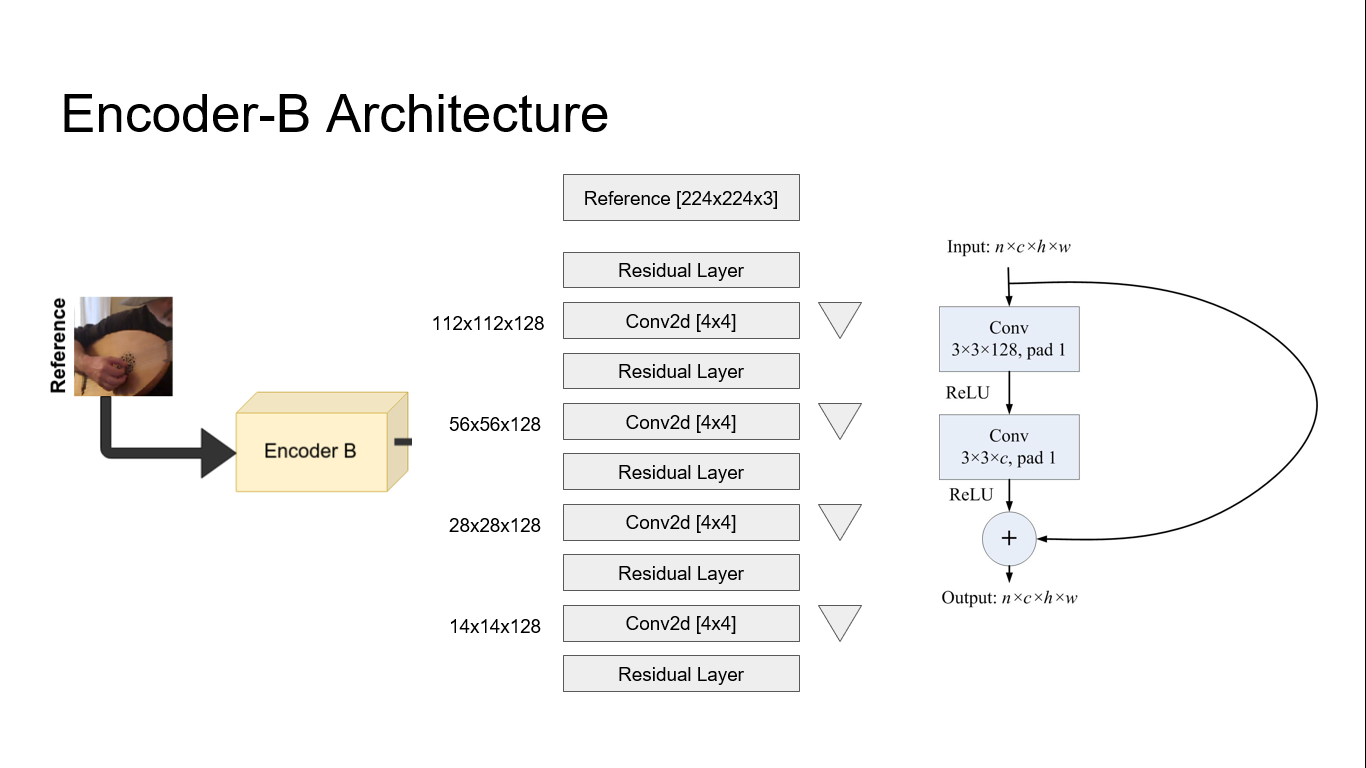
Figure 7: - Histogram of the importance level

**Pairwise Multiplication.**

We now perform the pairwise multiplication between Importance mask and Binary Code. The decoder has the benefit of knowing the important pixels and has only the necessary information for performing a compression.

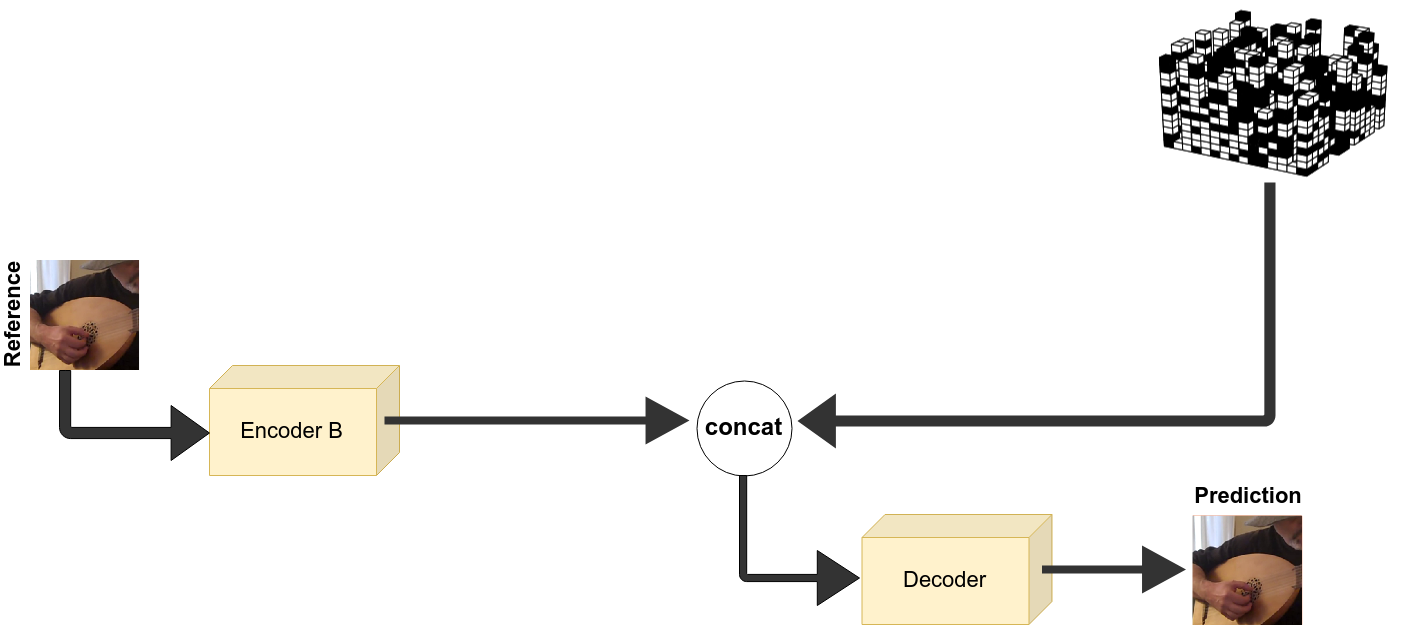
**Experiment 3 – Encoder B and the Decoder**

**Encoder B**

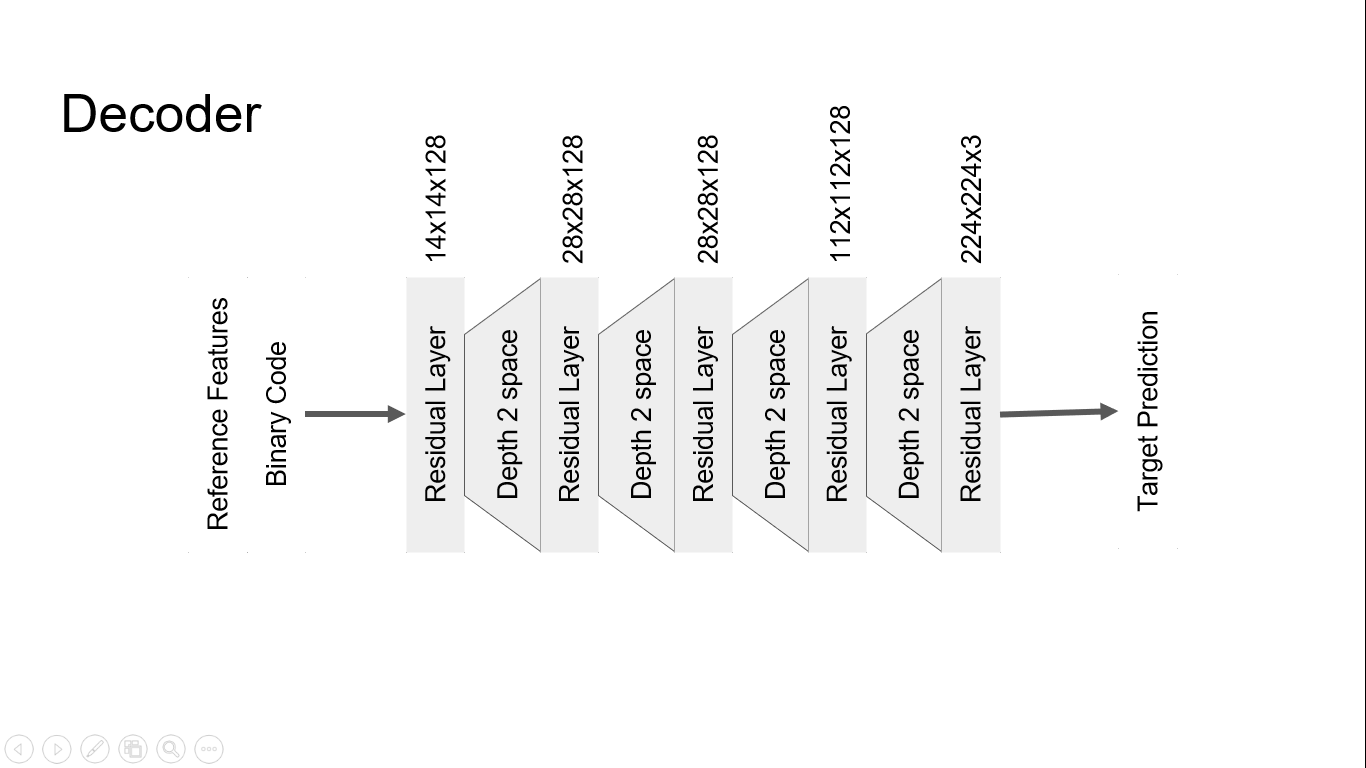
We now try to reconstruct the target image through the reference image. Before we send it to the decoder, we convert the reference image to a feature space of 14\*14\*128 dimension. Note that we send only the reference frame of size 224\*224\*3 to the network.

**The Decoder**

The binary code received from the Encoder A is now concatenated along with the Encoder B.



The decoder up samples the 14\*14\*128 feature map into the expected 224\*224\*3 by passing through different layers.

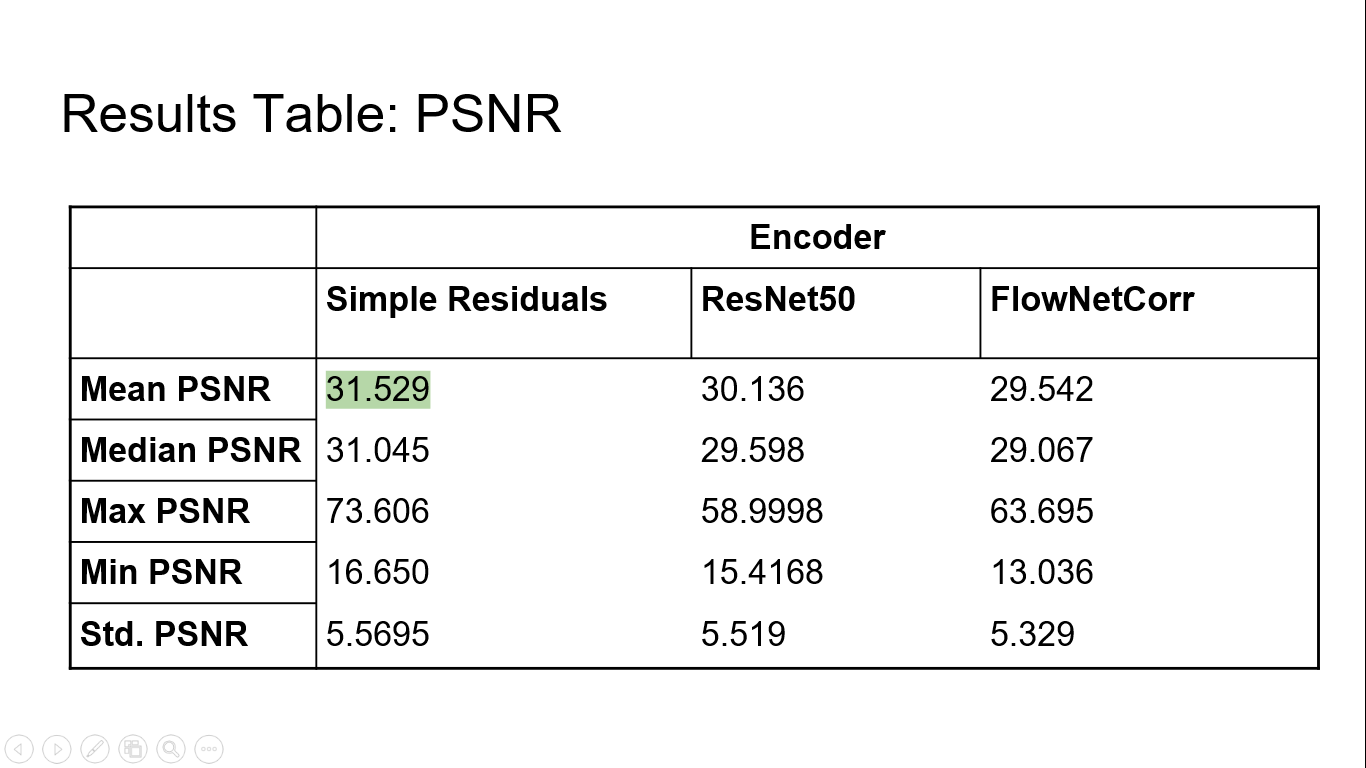


**Loss Function**

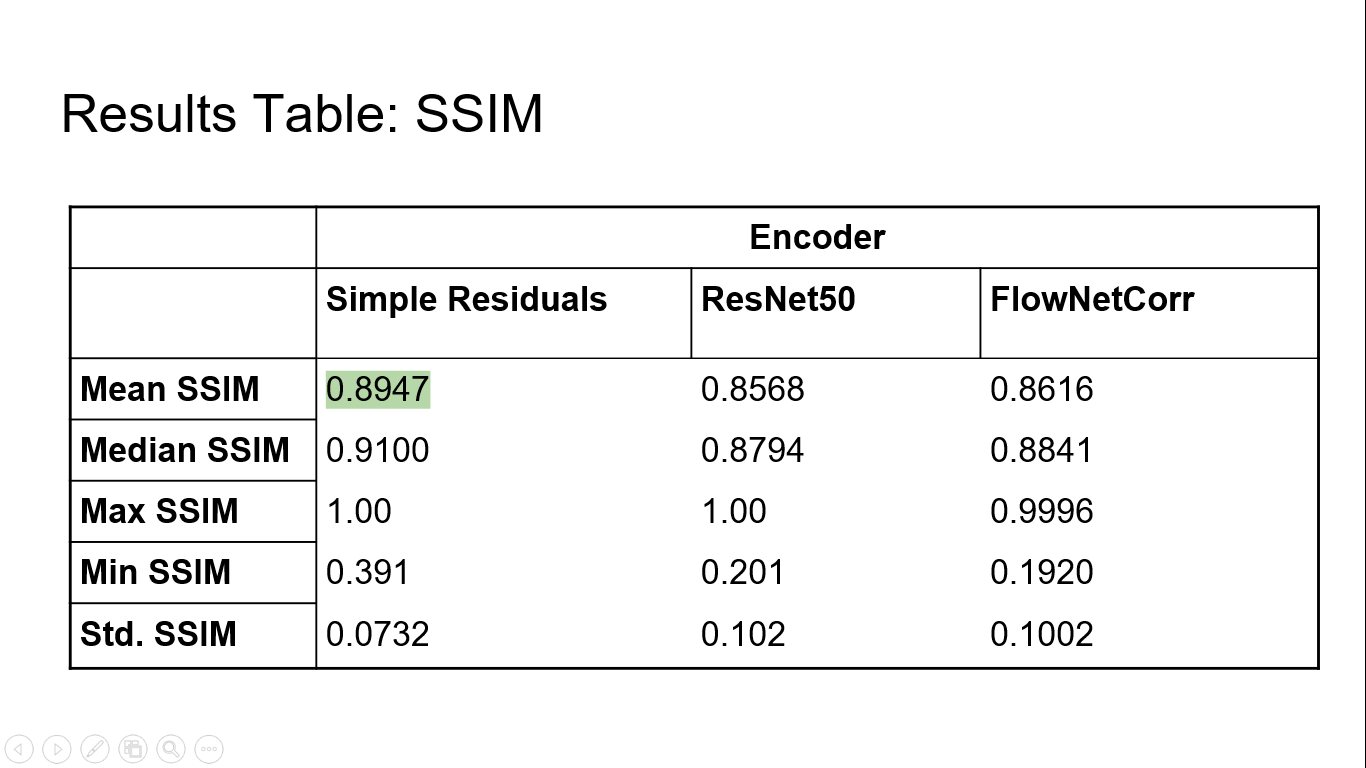
Using Mean Squared Error function for reconstructing the images results in blurry, low quality images. However, an alternative solution can be using GAN’s.

**RESULTS**

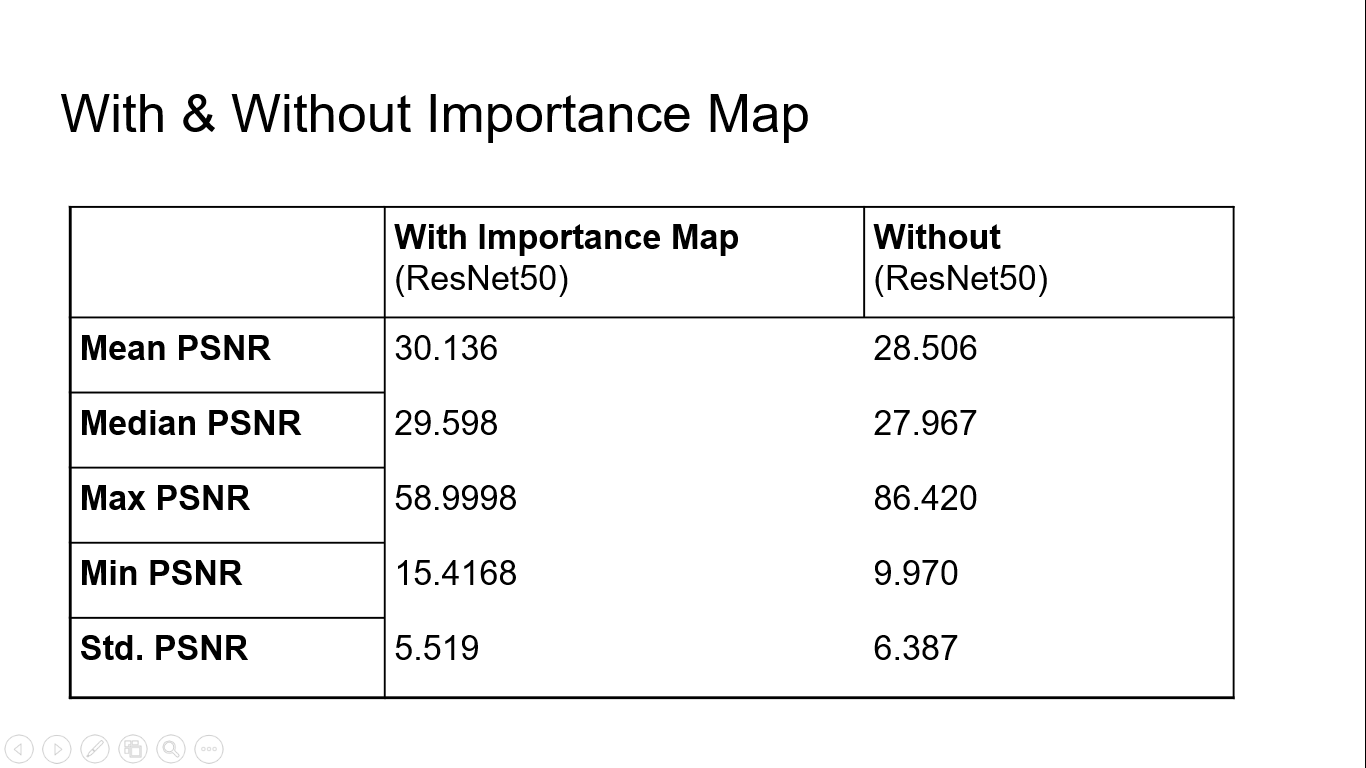
**PSNR**



**SSIM**



**With and Without Importance Map**



**Experiment Results**

**Future Works**

**Reference**

1. Fischer, Philipp, et al. "Flownet: Learning optical flow with convolutional networks." arXiv preprint arXiv:1504.06852 (2015).
2. Kay, Will, et al. "The kinetics human action video dataset." arXiv preprint arXiv:1705.06950 (2017).